An investigation of online multiplayer game development within the context of novice development team with the help of a generic multiplayer framework

A thesis submitted for the degree of Masters by Research (MbR)

by

Animesh Sharma
School of Design and Informatics
Abertay University

supervised by

Dr. Robin Sloan

Mr. Martin Lynagh

August 2022

Declaration

Candidate’s declarations:

I, Animesh Sharma, hereby certify that this thesis submitted in partial fulfilment of the requirements for the award of Masters by Research (MbR), Abertay University, is my own work unless otherwise referenced or acknowledged. This work has not been submitted for any other qualification at any other academic institution.

Signed …Animesh Sharma…………………………………………………….

Date……08/08/2022………………………………………………………….

Supervisor’s declaration:

We, Robin Sloan, and Martin Lynagh hereby certify that the candidate has fulfilled the conditions of the Resolution and Regulations appropriate for the degree of Masters by Research (MbR) in Abertay University and that the candidate is qualified to submit this thesis in application for that degree.

Signed ……Robin Sloan…………………..……………..………….………

Date………08/08/2022…………………………………………………………..

Certificate of Approval

I certify that this is a true and accurate version of the thesis approved by the examiners, and that all relevant ordinance regulations have been fulfilled.

Supervisor……………………………………………………………….…….

Date………………………………………………………………………….

Table of Contents

Abstract

The multiplayer games have constantly evolved over the years to incorporate numerous elements such as competitiveness, synchronised gameplay, ranking system, leader boards, etc, to propel the multiplayer games and eSports industry. Online multiplayer games are a popular form of entertainment, and their development can be a complex and challenging task.

This study investigates the process of developing an online multiplayer game within the context of a novice development team, with the help of a generic multiplayer framework. The research aims to identify the key challenges and issues faced by novice developers when creating an online multiplayer game, and to explore the ways in which a generic multiplayer framework can support the development process and team involved in the development of the multiplayer game. The context here is the novice development team. With the increasing difficulty in the development of video games for novice developers with new mechanics, features and trends in game development, there is a need to understand the challenges faced by novice developers and their possible solutions during the development of a multiplayer game. Unreal Engine was chosen as the engine to develop the framework and the game in based on the papers and sources comparing the most popular game engines in the market. Rychkova et. al's (2020) Orbital Battleship and Buckley's (2021) Choosing a game engine suggested that Unreal Engine is more suitable for multiplayer games development. Cedric Neukirchen's (2021) Unreal Engine 4 network compendium was utilised for the purpose of understanding the architecture of the multiplayer API and the development of the multiplayer framework. The study employed a mixed-methods approach, including both qualitative and quantitative data collection methods. The framework was deployed and utilised by the development team to build a fully functional multiplayer game.

The findings suggest that the use of a generic multiplayer framework can facilitate the development process for novice game developers, by providing a ready-made foundational structure for implementing common multiplayer game features and functions. However, the study also identified several key challenges faced by novice developers, including the technical issues encountered and the need for specialised technical skills and the importance of effective teamwork and communication. Overall, the results of this research highlight the potential value of using a generic multiplayer framework for supporting the development of an online multiplayer game by a novice team.

Acknowledgements

First of all, I would like to thank my supervisors, Dr. Robin Sloan, and Mr. Martin Lynagh, for all the help and impeccable guidance they have provided me throughout the year. They have made my time for this program quite enjoyable. They have also recognized my interest in eSports and kept my mental intact with extra motivation throughout the program. I’ve been able to receive the Abertay Futures Scholarship with the help from them. My principal supervisor, Robin has provided extremely useful feedback and very helpful advice with subtle and minute details at each small step. I’ve learned so much about the strategies and planning in this research. He has guided me in writing the thesis and carrying out the research in an elegant way with detailed feedback and comments.

My other supervisor, Martin has guided me throughout the MbR and previously in MProf with in-depth emphasis on general game development and multiplayer development along with genres of multiplayer game to develop. With his help, I’ve been able to learn and apply the elegant networking practices to develop a multiplayer framework in Unreal Engine. Martin was also responsible for assigning me a fantastic undergraduate game development team to work with. Both Martin and Robin have provided me with extremely useful sources and references to be utilised in the process of thesis writing and other relevant purposes. They have guided me in coping and mitigating with delays in this research project as well. Words are not enough to describe the support both my supervisors have given me. I would also like to thank the Graduate School Staff for providing me help and support throughout the year.

I would like to thank all the participants and my team members with whom I worked to develop the turn-based multiplayer game called Land of Morphie and the testers who tested the game and provided useful feedback and response for the game. The team members maintained excellent team harmony and fun environment to work in, with occasional joking and motivational moments to brighten up the mood.

I would like to thank Dr. Dayna Galloway and my colleague Alexander Tarvet (Sandy) who have deployed me as a lab assistant for the module DES205 which in-turn has given me first-hand experience in teaching. With the help of Sandy, I’ve been able to engage in extensive engagement with second year undergraduate students for their Level Design and Scripting module in Unreal Engine. The students have been extremely fun to work with, for their insightful projects in the module.

Lastly and most importantly, I would like to thank my family and close friends who have thoroughly supported me to pursue this program. I would not have been able to

do this course without my parents in the first place. They supported me to register for this course in a very tight situation in order to continue my studies in the UK. They have talked to me and guided me on a day-to-day basis to keep my motivation up top throughout my time here.

List of Figures and Tables

Figures

Tables

1. Introduction

Online video games engagement is increasing (Clement, 2022). Individuals with or without real interest in outdoor games are devoting their time to playing video games. The increasing popularity of online games and their eSports business models have made the games even more engaging with more online players (Figure 1), more professional players (Khromov et. al., 2018), and viewers (Gough, 2022) along with the rise in revenue generated (PwC, 2021). In these complex video games, the players themselves play either as individuals against their opponents in a 1v1 match or as teams to compete in order to climb the rank ladder. Additionally, the players play for their pleasure in quick matches or unranked matches, making those games even more playable and enjoyable. The features of voice chat and text chat have allowed these games to be played with friends and family. Online games have become a ground of social interaction and fun for gaming enthusiasts. Most of the online games have these common aspects which highlight their uniqueness compared to other game genres.

More recently, the COVID-19 pandemic has further impacted on the number of video games players due to availability of time devoted to only home stay. The industry has taken a new turn with the community engaging with online games in a new and different style. Interestingly, some professionals of sports such as Sergio Aguero, retired and former player of Manchester City FC (Lawless, 2020), Diogo Jota of Liverpool FC themselves were found engaging in live video games activities during this time (Jones, 2022). As a result of the COVID-19 pandemic, many people turned to video gaming as a form of entertainment during long periods of being locked down at home. During a global survey in June 2020, around 60 percent of respondents stated that they were playing more multiplayer games during the pandemic (Figure 1), no doubt in part to replace the face-to-face interaction that was severely restricted, or even banned entirely, during parts of the crisis (Clement, 2021).

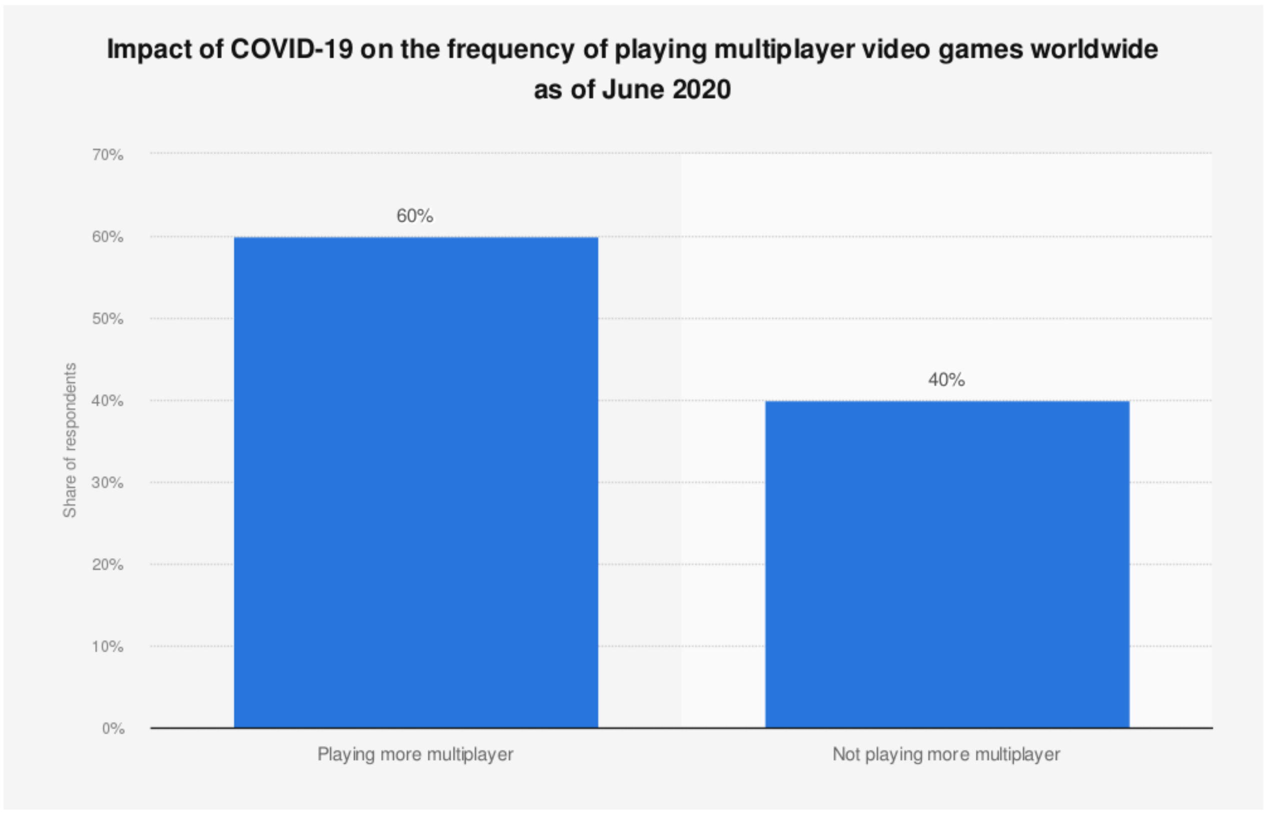


Figure 1. Impact of COVID-19 on the frequency of playing multiplayer games worldwide as of June 2020 (Simon-Kucher and Partners, 2020)

Online remote tournaments have been organized consisting of teams of professional players and professional athletes rather than on-site tournaments to spread the publicity of these competitive games and even further promoting the engagement from players around the globe with large fan followings (Hawkins, 2020). According to IDC data (International Data Corporation), global videogame revenue was expected to surge by 20% to $179.7 billion in 2020, making the videogame industry a bigger revenue generator than the global movie and North American sports industries combined (Witkowski, 2020). The current global video game market revenue is $208.6 billion as of July 2022 (Statista, 2022a) and the current global eSports market revenue is $1.384 billion (Newzoo, 2022, graph).

With the increasing popularity of online video games, the general interest in thoroughly studying and understanding video games with the help of research has also increased (von der Heiden et. al., 2019). Much of the attention to video game research has been negative while some research has also focused on the positive sides of video games along with technical research projects (American Psychological Association, 2010).

Some crucial areas of video game research are yet to be focused on despite the increasing focus on games as a subject of study in academic research. There is a need for more research to focus on production studies with a support for development of games as it is increasing in complexity in terms of the skills, knowledge. For example, the difficulties faced by developers freshly starting out on making games. All experienced developers in the industry were all beginners when they started to learn how to develop video games. Video games are really complex to develop, especially for a team starting out i.e., a novice team. Furthermore, multiplayer games are even more difficult to work on (Morgan, 2009). Every multiplayer game is required to have a common multiplayer pipeline which governs the traversing of players in lobbies and matches. Some bugs and issues in online games can be tremendously difficult to even reproduce, let alone fix them. That is because online games don't run on a single instance but rather each connected player/client has its own instance of the game running with the governance from a remote server (Chen, 2015). To implement functionalities, programmers are required to write remote function calls which are basically called from another machine in each different scenario of gameplay. Aspects such as slow internet or lower system specifications can also cause anomalies in the gameplay. Even in published multiplayer games on the market, there remain bugs which are yet to be fixed by the developer or take longer than expected to be fixed. For example, in *Valorant*, Riot's tactical FPS Shooter, there was a bug (now fixed) when an observer watching the agent/character 'Chamber' pop his ultimate ability, 'Tour De Force' which is basically a fast-shooting sniper rifle, the gun laterally rotates giving the observer a view of incorrect gun orientation (Riot Games, 2020). This brings us to the question of what these issues are exactly and how they can be fixed, especially when a novice team is working on a multiplayer game, since a novice team is more likely to run into more issues than a fully experienced team of developers. There is a need to understand these complex technical issues faced by a novice team and also developing a multiplayer pipeline to aid the team for better technical knowledge of the online games and the solutions to the problems encountered during development. This study proposes to explore how an online multiplayer game is developed by a novice team with the help of a generic multiplayer pipeline keeping the aspects such as resources, team-size, market, players response, technologies in mind. The intended application of this research is to inform future support for novice teams or developers entering game development from education into industry.

1.1 - Research Hypothesis

The foundation and the development of a generic multiplayer framework will have positive impact on the development of a multiplayer game by a novice team.

1.2 - Research Problem

The project aims to investigate and understand the design and development of competitive online multiplayer games. The problem at hand lies in finding the problems encountered both technically and conceptually by small-sized novice team (with about 2-12 members) and their solutions during a development of a multiplayer game. The research then proposes to investigate, to what extent, a team with limited experience, knowledge, and time can develop a fully functional multiplayer game prototype when provided with targeted support and resources.

1.3 - Research Questions

To address the research problem, test the research hypothesis and investigate the technicalities of multiplayer games including the challenges faced during development and the technologies available to support a team of novice developers, this research leads to the formation of the following research questions -

RQ1: What challenges, knowledge gaps, and risks do novice development teams face when working on concepts for competitive multiplayer games?

RQ2: What multiplayer development pipelines, workflows, and tools exist that can support novice teams, and how can a generic multiplayer pipeline be developed for use by teams working on a self-directed multiplayer game concept?

RQ3: How will a generic multiplayer pipeline impact on the design process and knowledge development of a novice team, what are its limitations, and how will the end product be received by players?

The research question 2 and research question 3 are directly linked to the research hypothesis testing the impact of a generic multiplayer framework on the development process.

1.4 - Objectives

To address the above formulated research questions, this research project established the following objectives.

1. Study the background of multiplayer games and analyse the market for multiplayer and eSports games.
2. Study the literature around multiplayer games, technical challenges, and novice developers along with game production studies and teaching development to novice developers.
3. Develop a generic multiplayer framework for a novice team which they can use order to develop their own multiplayer game prototype.
4. Work with a novice team to study the live development of a multiplayer game prototype considering factors such as team knowledge, skills, technical challenges, and issues faced.
5. Evaluate the framework with the help of feedback provided by testers playing the multiplayer game.

2. Background

The purpose of the following sections is to review the academic literature, industry, and market sources pertinent to multiplayer games development, inclusive of the history of multiplayer games, the market trends in the industry, along with the revenue generated by online and esports games. This chapter provides an underpinning rationale for the need for focus on multiplayer development in the context of games education and novice teams.

2.1 – History of Multiplayer Games and Genres

Multiplayer games have come a long way along with eSports through the development of technology in terms of both hardware and software (Semanova, 2020). The evolution of multiplayer games is attributed to the new functionalities and features available in the networking systems (Chikhani, 2015).

The origin of multiplayer games dates back to the early games, the first commercially released video game, *Computer Space* in 1971. Next year, the first commercially successful video game was released, Pong. Pong is popular even to this day (Pearce and Artemesia, 2009). The inspiration for multiplayer games were drawn from these games. Later, we witnessed the golden age in the late ’70s with the release of *Space Invaders, Asteroids*, and *Pac-Man* (CDW, 2021).

In the year 1984, the game Islands of Kesmai was released which supported up to one hundred players via the CompuServe online service (Stephenson, 2008, p. 115). It is considered a predecessor of modern Massive Multiplayer Online RPGs (Swain, 2013, p. 156). The very first version of a LAN game supporting up to 4 players came in 1993, with the name, *Doom*, one of the earliest first-person shooters (Sabadello, 2006). The *Doom* series has been developed along with the advancement of graphics and gameplay with multiple successfully released titles (Sabadello, 2006).

It is worth noting that these MMORPG and FPS titles laid the foundation of the world of multiplayer games, i.e., they were one of the first established digital multiplayer games responsible for the onset of the further development of multiplayer games. Furthermore, the release of an RTS title (Real Time Strategy), *Warcraft: Orcs & Humans* in 1994 set the facilities for future RTS games (Sabadello, 2006). *Warcraft* allowed only two players to play in a multiplayer battle by Lan/modem or local networks (CDW, 2021). Similarly, in the same era, *The Lost Vikings* laid the idea of cooperative multiplayer because it had the player having the ability to control different characters, though only one at a time to solve puzzles for game progression (Weiss, 2014). In 1997, *Goldeneye 007* became one of the first split-screen multiplayer games allowing up to four players to play against each other (Karhulahti and Grabarczyk, 2021).

In the ’90s, several other titles were released for up to 4 players simultaneously, such as *Mario Kart*, *Counterstrike*, *Marvel against Capcom*, *Contra*, *Street Fighter II*, etc (Yuzyk and Seidner, 2022).

The last two decades of the 20th century were the beginning of the multiplayer era of video games along with the foundation of the eSports industry. Although, in 1972, Stanford University organized the first video game competition for *Spacewar* (Yuzyk and Seidner, 2022). Later in 1980, Atari’s *Space Invaders* tournament was held which is known as one of the most popular first-recorded competitive gaming events (Yuzyk and Seidner, 2022). Since multiplayer games began to evolve later in the ‘1990s and ‘2000s with more titles and genres such as *FIFA*, *Half-Life*, *Mario Kart*, etc, the eSports industry adjusted along with the newly coming online games with the networking framework, i.e., the framework or the API supporting networked games (Yuzyk and Seidner, 2022).

With the help of the two multiplayer architectures, i.e., the multiplayer modules to support networking (Spurling, 2005) directly applied to single player games to develop them into online multiplayer games, there has been several genres divided further into several types of multiplayer games such as seamless multiplayer, winless multiplayer, goalless multiplayer, always online, asynchronous multiplayer, and asymmetrical gameplay multiplayer games (Forehand et al., 2014).

Seamless multiplayer is a type of co-operative multiplayer where players can join each other in their respective worlds such as *Genshin Impact*, *Watch Dogs*, etc (Forehand et al., 2014). Winless multiplayer does not have any winner, the players fight together to survive and the game ends when all players are dead (Chahat, 2020). Examples of such games or modes are *Call of Duty: Zombies Mode*, *Halo 3*: *ODST – Firefight Mode*, etc (Chahat, 2020). Goalless multiplayer games are games with no goals or objectives, the players make their own goals such as *Minecraft*, *Animal Crossing*, etc (Jesper, 2007).

Some games or game modes such as *FIFA Ultimate Team* or *Genshin Impact* cannot be played without an internet connection also fall under the category of always online games, this feature benefits social interactions between the players (Mirowski and Harper, 2019). This has also led to a new idea of a game genre named asynchronous multiplayer where players do not have to play at the same moment to play against each other, turn-based multiplayer games fall under this category (Kelly, 2011).

Finally, the game genre of asymmetrical gameplay multiplayer game has attracted a lot of audiences from all over the world. This type of game is a multiplayer game in which several players can play the game together in diverse ways like different maps, goals/objectives, abilities, etc (Bycer, 2019). Games such as *Dota*, *Overwatch*, *Valorant*, *League of Legends* fall under this category (Bycer, 2019). This category is currently shaping the eSports world and drawing audience in terms of both players and viewers.

Currently, multiplayer games and video games in general with their current potential and rising interest, the streaming services such as Twitch, YouTube, etc. will continue to facilitate the development of emergent multiplayer games and other video games (Wodarczyk and von Mammen, 2020). These streaming platform services have led to increase in engagement and popularity of the multiplayer games along with the rise of eSports (Wodarczyk and von Mammen, 2020).

In summary, multiplayer games have evolved in terms of the features, genres, and graphics with the smallest and simplest forms of video games to highly complex and competitive video games along with different categories and their wide application in the eSports industry. Additionally, the multiplayer games continue to grow in popularity and engagement with the help of streaming platforms such as Twitch, YouTube and through eSports events.

2.2 – eSports Market and Titles

The aim of this section is to review the international market for competitive online games, and the roles of the top eSports countries in the market. The top eSports countries in the market are the countries with the most engagement in eSports in terms of viewership, revenue, and professional players, etc. They have the highest consumer activity around eSports.

The market of the eSports games (games with an eSports model or eSports ready games) and online games industry in top eSports countries is widespread in terms of various game genres, which means the eSports market is highly segmented between the various game genres (Research and Market, 2021). Various consoles and devices have generated distinct revenues in the recent years. The eSports market by platforms is divided into consoles, PC, mobile, and others (Research and Market, 2021). With the ongoing digitalization and the overtaking of the games industry compared to the other media industries such as the music and the film industry in terms of revenue (Rama, Fernandez, and Camacho, 2020), eSports has attracted more even more professional players and spectators from the regions around the world (Geyser, 2022).

According to Statista (2022b), China has generated the most revenue among all countries in 2021 with a total of $ 5.87 billion for online games alone (Statista, 2022b, table 1). At second place, the US stands with $ 4.84 billion and Japan, South Korea and UK follow with a total generated revenue of $ 3.7, 1.34, 1.33 billion respectively . The total global revenue generated by online games alone in 2021 is $ 23.71 billion (Statista, 2022b, bar chart 1).

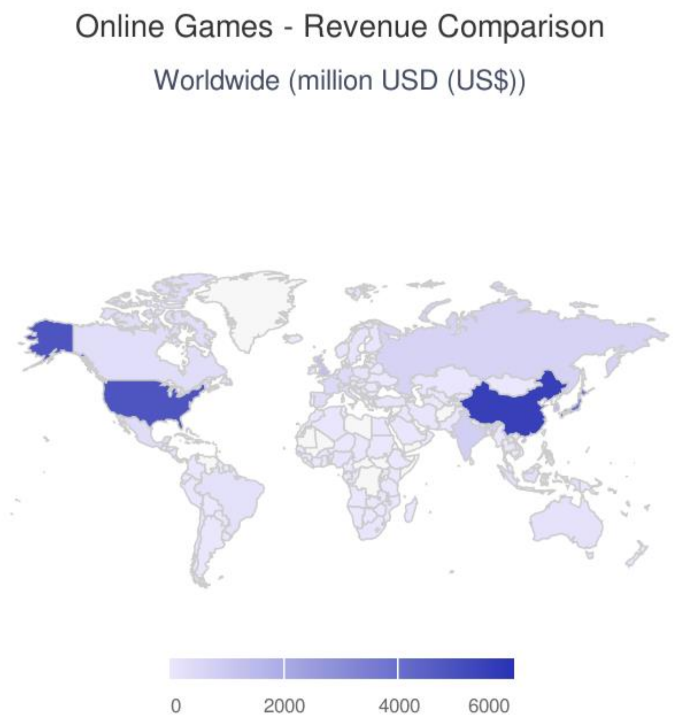

Online Games - Revenue Comparison
Worldwide (million USD (US$))
0
2000
4000
6000

Figure 2. Global Online Games Revenue (Statista, 2022b, table 1)

It is important to note that, the population, GDP and popularity of online games play a role in generating revenues in the countries around the world. For example, as mentioned above, China, USA, Japan, etc are among the top countries in generating revenues for video games and online games. Although, there is an exception as well, this is because of the value of currencies in different countries, GDP, their respective market, lack of sponsorships by brands and recognition of eSports as a sporting discipline by their respective government (Esports Insider, 2022). For example, India has only generated 0.79 billion dollars of revenue in online games in 2021 despite having 1.366 billion population (Statista, 2022b, table 1), as the market in India is largely occupied by mobile games (Basuroy, 2022). The same platform, mobile phone has led to increase in revenue and users in China (Thomala, 2022). This is because the revenues generated is dependent on the GDP of a country, gaming market, ARPU or the average revenue per user (Gupta, 2022). Recently, popular games' genres like battle royale have also attracted more users (Clement, 2022).

On the other hand, the eSports market revenue generated worldwide has top 3 regions (Figure 3), China with $ 360.1 million, United States with $ 243 million USD and the entire of Western Europe with $ 205.8. This means that eSports has grown the most in these 3 regions with widespread engagement in the eSports industry (PocketGamer.biz, 2021).

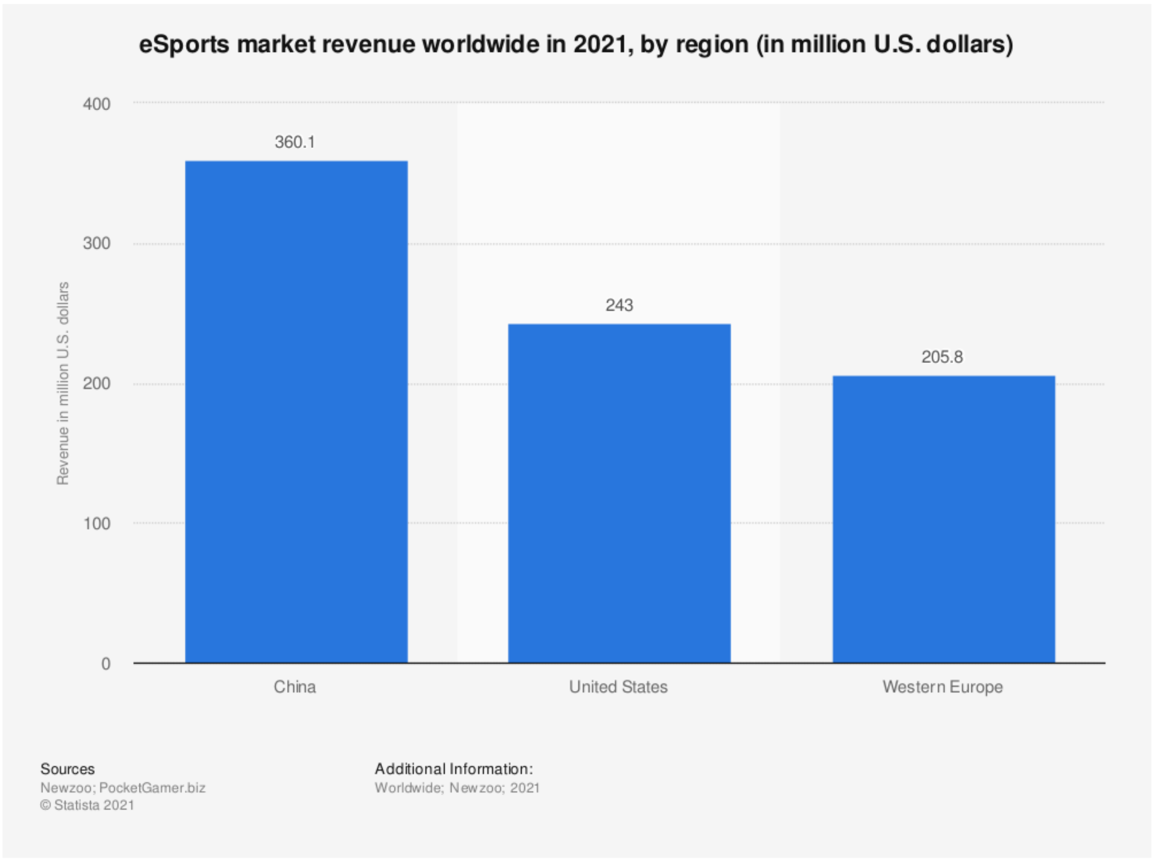


Figure 3. eSports Market Revenue Worldwide in 2021 (PocketGamer.biz, 2021)

The total number of registered players in online games in 2021 are reported to be 1.1 billion players while the user penetration rate is 14.6 percent (Clement, 2022). Mentioning user penetration rate is more important because players registered per targeted audience is the key to understanding the revenue generated as well as the number of players registered. Although 1.1 billion players doesn't mean 1.1 billion people, it actually means 1.1 billion accounts created. Unsurprisingly, China has also registered most players of online games in 2021 with 382.8 million in number with user penetration rate of 26.4 percent. South Korea, Japan, US, and Australia follow China with user penetration rate of 23.7, 23.0, 22.6 and 21.2 percent, respectively (Clement, 2022). Additionally, the projected users and revenue in all countries is projected to increase till 2025 and further on (Research and Markets, 2021). Although, the projected revenue growth is projected to decrease after 2020 (Statista, 2022b, graph 1). Because of the global pandemic of COVID-19, the year 2020 has registered a large number of online players (Statista, 2022b, graph 3) with vast amount of revenue all over the world (Statista, 2022b, graph 2).

“Thanks to high smartphone penetration across the globe, gaming is increasingly happening on the go. Mobile games of all kinds are becoming more and more popular and attract players with additional premium contents or functionalities. Thus, traditional online games slowly lose attractiveness to its usual audience. Only recent phenomena like the Battle Royale hits Fortnite and PUBG are still driving the market’s growth and shape online gaming in general.” (Clement, 2022)

Coming to the popular online games in the world, there are several titles with distinct genres. Some of these genres include shooters, fighting, sports, MOBA, battle royale, etc (Research and Markets, 2021). Reports also suggest that multiplayer online battle arena (MOBA) was the biggest segment of eSports market with a total of 40.9 % of the total segment in 2020 (Research and Markets, 2021). Topmost popular games in terms of awarding prize money include Dota 2, Counterstrike: Global Offensive, Fortnite, League of Legends, Arena of Valor (Esports Earnings, 2022).

In-terms of most watched eSports titles, the top games in 2021 (Figure 4) were League of Legends, Counterstrike : Global Offensive, Mobile Legends : Bang Bang, Dota 2, PUBG Mobile (Markov, 2022).

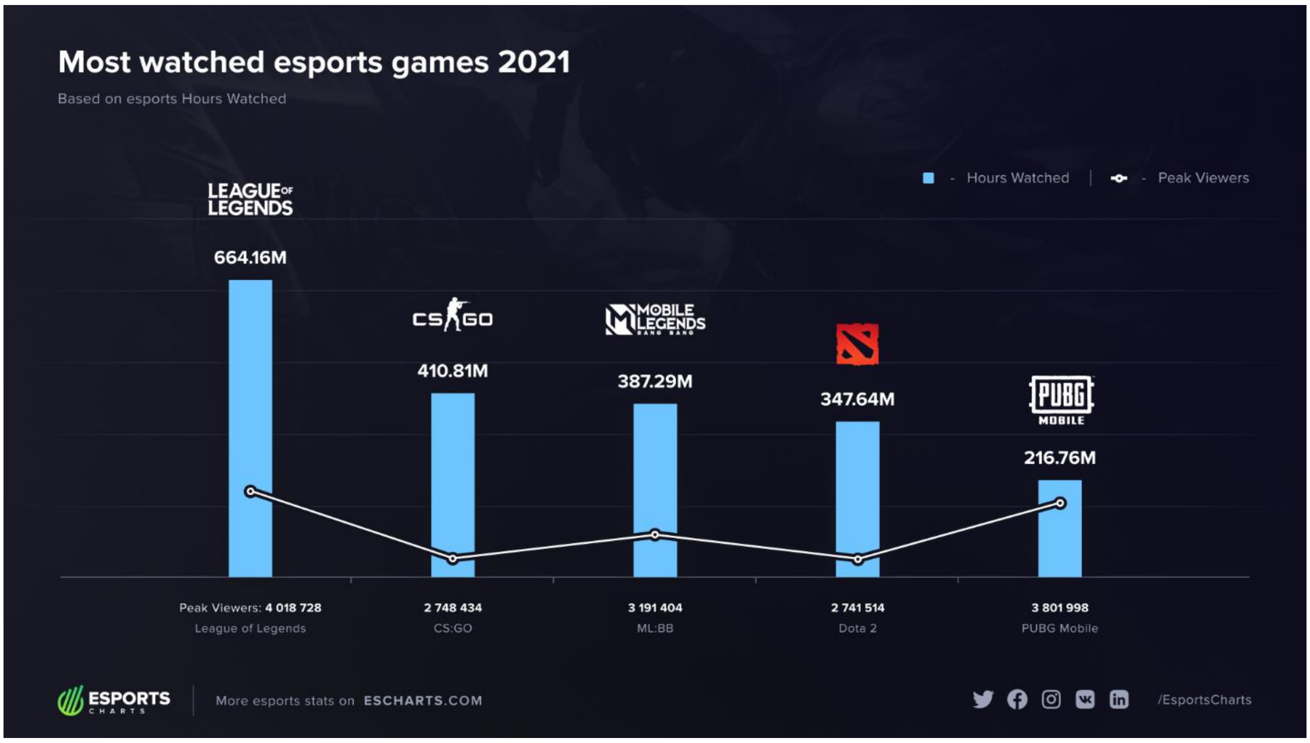


Figure 4. Most Watched eSports Games in 2021

Lastly, the top 10 most played (total hours played) eSports games are (Syakah, 2021) –

1. Counterstrike: Global Offensive
2. League of Legends
3. Dota 2
4. Fortnite
5. Overwatch
6. Arena of Valor
7. Apex Legends
8. FIFA
9. Player Unknown's Battlegrounds (PUBG)
10. Valorant

Summarising the data and analysis above, we’ve seen the market trends, top revenue generating regions and countries by eSports and online games. We’ve seen the top eSports multiplayer games in terms of watch hours and playtime. We’ve understood the genres of games that are the most popular in the eSports industry which include shooters, MOBA, sports, battle royale, fighting, etc. This provides us useful information for the basis of the genre of the game to be chosen for this research and that means games which have team vs team or player vs player feature with goals/objectives, i.e., asymmetrical gameplay online games (section 2.1) are the most popular genres of multiplayer games in the market. These are dissimilar to co-operative multiplayer games or goalless multiplayer games.

3. Contextual Review

This chapter builds up the foundation of the research and reviews the literature accumulated to help in answering the research questions or possibly understanding the limitations in the literature failing to answer all the research questions. It assists in gathering the non-proprietary materials, assets, and tools available in order to support a novice team in developing a multiplayer game.

Firstly, section 3.1.1 helps in reviewing the technology required and involved in the development of online multiplayer games which also include the game engines and the middleware solutions available for integration in these game engines. This addresses research question 2. Secondly, section 3.1.2 helps in understanding the common technical problems of multiplayer development and their possible solutions along with the issues faced by novice developers or programmers which further addresses research question 1.

3.1 – Technology and Challenges for Multiplayer Game Development

3.1.1 – Technology/Game Engine Review

The first stage of technology review for online multiplayer game development concerned an evaluation of available game engines, their features, and capabilities. Technology selection in terms of reviewing an engine best available to develop an online multiplayer game was extensive research in itself. Additionally, there was a need to search for the tools and plugins available to readily develop multiplayer games. Not only that, but the game engine available should also be capable of creating a generic framework for the development of multiplayer games. Initially, the best way to start selecting from options was to look out for the best and most popular engines in the market with a wide variety of functionalities and modularity. This means that the strategy to review the engines for the development of multiplayer games was to look out for aspects such as –

1. Readily available multiplayer modules/functionalities
2. Free to use
3. Wide documentation, support, and community

4. Wide range of application in multiplayer games

Currently, there are some game engines that allow the development of games to be applied for wide variety of browsers and platforms from single development instance in some cases (Marín-Vega et al., 2016). The most common options found were Unity, Unreal Engine, Godot, Game Maker Studio, etc (Vohera et al., 2021). Given the wide functionalities of Unreal Engine and Unity and their tremendous application in the industry (Tyler, 2017), it was best to first investigate the features of the remaining engines. The table below differentiates all the popular engines in terms of different functionalities provided by them.

ENGINE	**PRICE**	**PLATFORMS**	**TARGETS**	**3D**	**LANGUAGES**	**MULTIPLAYER FEATURES**
Unity	Free	Windows, MacOS, Linux	Windows, Mobile, MacOS, Linux, Consoles	Yes	C#, Bolt	Yes (obsolete) Requires third party plugins
Unreal Engine	Free	Windows, MacOS, Linux	Windows, Mobile, MacOS, Linux, Consoles	Yes	C++, BPs	Yes
Godot	Free	Windows, MacOS, Linux	Windows, MacOS, Mobile	Yes	GDScript, C++, C#	Yes
Game Maker Studio	Paid	Windows, MacOS	Windows, MacOS, Mobile, Consoles	Yes	Game Maker scripting language	Yes
RPG Maker	Free	Windows	Windows, Consoles	No	Drag and Drop - DnD	Yes (requires extra plugin)

Table 1. Comparison between Game Engines

Starting out with RPG Maker, as the name implies RPG Maker was primarily designed to develop RPG Games quickly with less effort (Azmi, 2016). RPG Maker's functionality is only limited to 2D Games (Azmi, 2016), which limits the overall potential of this research. Developers can develop multiplayer games with an extra plugin, Alpha Netz, but with limited functionality, genre. An MMO (massive multiplayer online) game cannot be developed with this plugin (Kagedesu, 2018), it would not have been best to choose this engine. Likewise, Game Maker Studio has similar features as the RPG Maker. It is one of the most recommended engines for beginners, mainly for the development of 2D games but is not limited to it (Eugene Public Library, 2021). It is possible to export games to multiple platforms with a purchased license and there are additional features such as drag and drop visual scripting and custom Game Maker Language (GML) for advanced programming (Eugene Public Library, 2021). The engine is proprietary for exporting (building/packaging) in different platforms and only provides GX.games exporting for the free version (GameMaker, 2021). GX.games is a platform for games to be directly played in Opera GX (Opera Team, 2021). The exporting of games to merely GX.games in the free version alone limits the overall freely available functionalities for this research.

Moving forward with the options, Godot is an engine that provides support for developing 2D/3D games with multiplayer features with the help of its High-Level Multiplayer API (Liniestsky et al., 2019). The Godot Engine documentation points out in the direction of some disadvantages with Godot networking API and its bullet objects system. For example, implementing bullets mechanics for a gun in any shooter game requires one of the methods having each bullet calculate its own path and handle its own collision for a lot of flexibility, this comes at the cost of performance (Liniestsky et al., 2019). There are some additional performance hits along with additional synchronisation of positions needed with the connected clients on the server (Liniestsky et al., 2019). Though, the documentation does provide a very detailed explanation and steps to work in the multiplayer API of Godot. This means, Godot is one of the engines in the list that can aid in the development of a generic multiplayer pipeline to support a novice student team. Unfortunately, Godot is a fresh new engine with low community support and deleterious for continuous development (Kritskiy et al., 2022). The nature of its limited number of published games, limited community, and assets in comparison to Unity and Unreal Engine allowed us to cross it off the list.

Considering modern game engines, Unity and Unreal Engine are the most advanced game engines and well supported, providing good capabilities for the purpose of both 2D/3D Game Development (Kritskiy et al., 2022). The most common and popular middleware solutions to tackle development of multiplayer games are compatible with Unity and Unreal Engine along with their own inbuilt integrated networking module (Eden, 2020). Both Unity and Unreal Engine have a large community with extensive documentation, tutorials, support, and free assets (Eugene Public Library, 2021), which is why these are the two most popular game engines in the market with wide utilisation in the games industry. This is an important factor in choosing game engines for novice teams. Both engines support multiplayer, although Unreal Engine is the only with integrated support. Unity's integrated multiplayer is still in-development although there are many 3rd-party frameworks such as photon which can aid in the development of multiplayer games in Unity (Buckley, 2021). Unity's own framework for multiplayer games is being deplored in the current version (Kritskiy et al., 2022).

Considering the factors of detailed documentation, technical support, large community, free to use, and support for multiplayer development, the options for developing an online multiplayer game narrowed down to the two most popular engines in the market, i.e., Unity and Unreal Engine. With Unity, some companies have recently started to develop and publish multiplayer games such as Fall guys, Genshin Impact (Murray, 2021) but they are not developed with features of the current multiplayer module in the engine itself as it is currently obsolete and still in beta for the upcoming update (Buckley, 2021). A game as basic as Pong can be developed with Unity's multiplayer but to develop online games with more features, developers still require extra support from other frameworks (Stagner, 2013). As mentioned previously, the middleware solutions for multiplayer games are compatible with both Unity and Unreal Engine such as third-party frameworks such as photon (Eden, 2020). Photon provides support for both Unity and Unreal Engine (Photon, 2021a). Although, the free solution provided by photon has some limits to it, such as, only 20 connected users at the same time, 16 peers per room or per lobby, 60 GB traffic per month (Photon, 2021b). The proprietary paid versions of this framework increase these limits according to the pricings. This means that to develop a multiplayer game in Unity, it is necessary to have solutions or framework like Photon which is also provides support in Unreal Engine but is limited to features according to the pricing, be it free or paid. Additionally, it would require extra knowledge and understanding of the SDK (Software Development Kit) to integrate with the game engine. Since, Unreal Engine has inbuilt and integrated multiplayer module to support developers to create multiplayer games. It provides a robust multiplayer framework (Arora, 2021). This means, to build an online game in Unreal, the developers are just required to have the knowledge of working in Unreal Engine along with basic knowledge of the client-server architecture widely used for multiplayer games. It is an ideal starting point for multiplayer games given its reputation with widely published and popular multiplayer games that are also operating successfully in the esports industry (Buckley, 2021).

Summarising the review above, many factors had to be considered when choosing the game engine for development. Which engine is the best, is rather very subjective and depends on the requirement of the project (Freiknecht, 2016). The requirement of this research project was to have a game engine with the most readily available networking support with wide documentation, tutorials, and community to create a multiplayer framework for the development of a multiplayer game by a novice team. The options in this review section narrowed down to Unity and Unreal Engine. Unreal proved to be more promising for this project due to the requirement of multiplayer support (Rychkova, 2020). More sources have chosen Unreal as the winner for multiplayer games (Burleson, 2021). This review helps in choosing the game engine and the game type/genre for this research project as discussed in the methodology (section 4.1.1).

3.1.2 - Challenges/Technical Assessment

The challenges or technical inspection involves identification of the common technical problems faced by teams while developing multiplayer games and their possible solutions. It is important to note that some common problems found in the lower engine level are already fixed in some of the middleware or frameworks, for example, the inbuilt networking module of Unreal Engine (Levchenko et. al, 2020). This section reviews the technical issues related to multiplayer game development. This aspect is crucial and directly links to the research question 1. Additionally, the section is framed to help in understanding the knowledge gaps faced by novice teams developing games, software or even applications as first-hand experience. This helps in gaining useful insight for investigation required to answer research question 1 as well.

3.1.2.1 - Common Technical Multiplayer Problems and Solutions

Every online game belongs to a specific genre and the development of each genre brings different types of challenges that developers face during development with it. Every research paper has reported different mechanical problems with the development of multiplayer games and each problem is case specific.

Madhav and Glazer (2015) in their book, *Multiplayer Game Programming: Architecting Networked Games* identify these common challenges which occur during development are –

1. Correctly formatting game data for efficient internet data relaying.
2. Synchronizing states in order to correctly share the same world across all players.

3. Mitigating issues with latency and jitter problems causing delays or loss of data or simply lag compensation (Lee and Chang, 2018).
4. Scaling the game code base without any loss in performance.
5. Cheat protection.

Wang et al. (2008) identified the common problems as –

1. Player's positions being incoherently represented.
2. Player object movement being jagged.
3. Player objects and environment missing collision detection.
4. Player objects wrongly missing or simply missing collision detection.

The possible solutions to each problems listed above are tackled case by case and researchers have reported the solutions or workarounds to these problems as following –

1. Correctly transmitting data for efficient data transmission involves serialisation of data for faster, effective, and lossless transmission that saves the state of the objects (Ignatchenko, 2016).
2. The major problem of a multiplayer game is to maintain the same instance of the game across all the clients (Spurling, 2005). To tackle this, methods such as *State Synchronization* and *Input Synchronization* are utilised, usually in a hybrid form like using input synchronization for aspects that are not governed by time and state synchronization for random features such as traffic position on the road (Spurling, 2005). One example can be, a player in an MMORPG doesn't need to know the location of the monsters 500 meters away which another player is fighting.
3. Network Latency or lag refers to the time taken for a data packet sent from its source to its destination. It is impossible to eliminate the lag completely, but a range of latency is defined as accepted values for different games which means that acceptable latency depends on the genre or type of the game (Spurling, 2005). A good solution to compensate lag would be the use of interpolation to prevent jumps between positions.
4. Usually, when the game projects increase in size with a greater number of scripts or classes, it is best to follow the best programming practices such as design patterns, object-oriented design, and SOLIDS to minimise loss in performance and avoid code breaks (Gamma et. al., 1994).
5. Mostly, cheat protection is ensured with the help of server authorising almost all of the gameplay. This ensures that a client never communicates with the

other clients, or it doesn't govern its own gameplay (Levchenko et. al, 2020). Some examples are Vanguard, VAC (Valve Anti Cheat), Global eye Anti-Cheat.

The player character or object problems discussed by Wang et al. (2008) have solutions which can be described as (Wang et al., 2008) -

1. Misrepresentation of players objects can mostly happen because of the collision either with the other players or the environment. The correct solution to deal with the collision has been described in point 3.
2. Player object movement is programmed in a smooth way through interpolation which means if a player object client is supposed to move, the movement occurs by the server with interpolation from the previous position in order to eliminate the jerky movement.
3. Handling collisions in any way is usually carried out by the server detecting the collisions and governing the position or the movement of the players based on the collision. Each player object client is then moved based on the force calculated from the collision.

There are some additional issues that have never been completely eliminated from multiplayer games such as peeker's advantage (advantage for players seeing their opponent first while peeking if their ping is higher than the opponent's) which is attributed to the nature of the internet (Donlon and Ziegler, 2020). Online games are affected by people with high latency client as well because their player data reaches the server slower than the one with low latency client. The high ping/latency players never get an overall gameplay advantage but, in some instances, they see low ping players earlier than the low ping players see them (Donlon and Ziegler, 2020). In a fully gameplay driven game like *EA Sports FIFA* or *NBA 2K*, the high ping players cause lag in the overall gameplay because the positions of all players on the level field have to be correctly replicated. Therefore, most of the eSports tournaments are held on LAN where each computer/console system has the same specifications and is on the same network as others, so that the pings are all the same and very low (Jansz and Marten, 2016). Furthermore, there are issues of Client FPS drops because of overload on the Server. Clients also get forcefully disconnected from a server session when too many reliable RPCs (Remote Procedural Calls) are called in the codebase (Unreal Engine, 2021a).

Research papers have also addressed the multiplayer model presented to this date which are – *Peer-to-peer* or *P2P* model and *Client-Server* model (Spurling, 2005). The most popular and most utilised model in today's world of multiplayer gaming is the *Client Server* model because it is easier to implement than the *P2P* model. The game publisher gets more administrative control meaning it can perform authentication, copy protection, billing, and accounting along with easy update or patch updates of the client copy (Spurling, 2005). Unreal Engine also uses the *Client-Server* model (Levchenko et. al, 2020). Similarly, PUN (Photon Unity Networking) also supports this model (Jetsonen, 2016).

In summary, development of different types of multiplayer games comes up with different types of technical problems and different solutions. Throughout the section, we have seen common technical problems and issues encountered with online games and their possible solutions.

3.1.2.2 – Knowledge Gaps and Issues Faced by Novice Teams and Solutions

It is important to note that we are investigating the knowledge gaps faced by novice teams when development multiplayer games which means most of the technical issues teams face and the knowledge they lack is specifically related to multiplayer/network programming. It is crucial to first understand the challenges novice programmers face in general when first programming in a proper project and their workarounds. That is because novice programmers may tend to make a relatively worse technical design choices or initially adopt worse programming practices (Agrahari and Chimalakonda, 2020). Furthermore, the common professional programming practices are also applied in game programming as well. Thus, this section primarily reviews the knowledge gaps and issues programmers face when encountering a new environment or problems and finding the solutions to those problems.

Some of the common issues encountered by novice programmers are –

a) Proposing an optimal and formal solution to a programming problem without bugs or issues (Sim and Lau, 2018).
b) Implementing a solution using a new tool, i.e., a new programming environment such as Unity, Visual Studio or Unreal Engine (Sim and Lau, 2018).
c) Bad choices or bad programming practices adopted by novice programmers (Agrahari and Chimalakonda, 2020).
d) Implementing solutions to problems efficiently, i.e., faster code (Agrahari and Chimalakonda, 2020).

The workaround or the optimal solutions for these problems mentioned above involve planning, designing the solution, implementing the efficient solution along with testing, documentation, and deployment (Sim and Lau, 2018). This is called an optimal software development cycle, that also includes one of the earliest methods called waterfall method which was the base model for other development like prototyping, agile development, and rapid software or application development (Sim and Lau, 2018). In this case, the similar methodology can be applied to game development or game programming in addition to implementation of new programming environments to simplify the problems and enhance students understanding on the code via visualisation (Sim and Lau, 2018). This type of visualisation environment is built in Unreal Engine called blueprints to aid developers implement mechanics and features faster (Boyd and Barbosa, 2017). A Siggraph paper has defined this visualisation language environment as KPL or Kids Programming Language (Schwartz et al., 2006). Unreal Engine does add an abstraction networking layer to the scripting languages for the developers in support of multiplayer Games (McEachen, 2004).

Novice game developers can very easily miss game-breaking issues such as clunky controls and puzzling game play mechanics (Darby, 2011). This is because the developers are already used to these issues and know how to mitigate them. These game breaking issues can be solved through testing and feedback from external testers (Darby, 2011).

Several sources and papers mention the most challenging part of game development, i.e., engineering (Blow, 2004). Writing code that should run quickly on target systems with clever optimisation is quite a challenging task and is mainly case dependent. Additionally, the technical challenges include getting the code to work to produce a desired functionality (Blow, 2004). Many novice programmers are able to learn the basics of programming, but they may struggle to apply these skills to solve non-routine problems and develop a level of fluency in programming. This can be a challenge for novice developers when working on complex projects, such as the development of online multiplayer games (Bachu and Bernard, 2012). This problem can be solved through collaboration which allows knowledge and skills sharing in a faster and efficient way.

The most common problem faced by novice programmers learning programming languages is the need to gain practical experience in order to become more proficient at programming. This often involves a significant amount of practice and repetition in order to develop and hone their skills. As a result, many students find it difficult to gain the level of proficiency required to effectively apply their programming skills to real-world projects. This also involves memorising the syntax in the muscle memory to be used constantly throughout the project (Khaleel et al., 2017).

Therefore, frameworks/APIs such as the Unreal Engine Blueprints have been implemented for novice programmers to tackle such problems. This further brings the idea of implementation of multiplayer framework to support novice developers.

Additional challenges faced by novice developers include the need to adapt to changes in the project plan and scope, as well as difficulties in maintaining a consistent work pace as the project grows in complexity. These challenges can be particularly difficult for novice developers, who may lack the experience and resources needed to effectively manage and overcome them (Westerdahl, 2019). In the area of communication, the challenges faced by novice developers may not be as clearly defined. However, some issues may arise when agile development methodologies are not fully implemented, which can lead to difficulties in effectively communicating with team members and managing the project as a whole (Al-azawi, Ayesh and Obaidy, 2014). The team lead plays an important role in overcoming the technical challenges while developing a multiplayer game or simply video games. This include prioritising the important mechanics, fixing game breaking bugs with effective communication.

Specific to multiplayer development, Graham Morgan’s (2009) journal, Simulation and Gaming, describes a review about the challenges of online game development. Some challenges attribute to the diverse skill set required to develop multiplayer games such as networking, databases, graphics, clustered computing, etc (Morgan, 2009). There is a need to balance the graphical and physical representation of the objects with optimisation to achieve maximum performance across all client machines. Furthermore, the developers are required to carry out the client-side prediction to reduce delay in the actions of the players such as movement, shooting, jumping, etc (Fiedler, 2018). The difficulty is in properly synchronising the state of the game between the client and server. This involves reconciling any discrepancies that may arise when the client and server disagree about the position and actions of player characters (Fiedler, 2018). The solution is that when the client and the server disagree, the client must accept the update for the state from the server. When this happens, the client discards the previous state stored and replays the state starting from the corrected state from the server to the predicted client state (Fiedler, 2018). The client essentially ‘rewinds and replays’ the last few frames of local player character movement, while keeping the rest of the game world fixed. Novice developers are more likely to run into these issues than experienced ones while developing multiplayer games.

Online multiplayer games have gained widespread popularity in recent years, and their development can be a complex and challenging task. Novice game developers, in particular, may face a variety of challenges when attempting to create an online multiplayer game. Gathering from the literature and text found above, the common challenges faced by novice developers can be described as -

1. A common challenge faced by novice developers is the need for specialized technical skills. Multiplayer game development requires a deep understanding of programming languages and frameworks, as well as networking and server architecture. Novice developers may struggle to acquire these skills or to apply them effectively in a real-world development context.
2. Another challenge faced by novice developers is the need for effective teamwork and communication. Multiplayer game development often involves a large and diverse team of developers, designers, and artists, who must work together to create a cohesive and enjoyable experience for players. Novice developers may struggle to effectively communicate their ideas or to coordinate their efforts with those of other team members.
3. A third challenge faced by novice developers is the need to balance the demands of the development process with the limitations of their resources and expertise. Multiplayer game development can be a time-consuming and resource-intensive task, and novice developers may struggle to allocate their resources effectively or to meet the demands of the project within the constraints of their available time and budget.

Overall, the challenges faced by novice developers while developing online multiplayer games can be significant, and they may require significant effort and resources to overcome. However, by addressing these challenges through effective learning and collaboration, novice developers can successfully create good quality online multiplayer games.

In summary, novice programmers and developers face several knowledge gaps which might lead to inelegant code base and inefficient game build. This is because a novice developer might try to implement features without extra emphasis on professional and elegant practices for the project. The general procedure for novice developers can sometimes be carrying out implementation via brute force method to complete the delegated tasks in order to progress towards finishing the project.

4. Methodology

The aim of this project was to develop an original generic multiplayer pipeline for a novice multi-disciplinary team so that the team could utilise it to develop a game artefact with multiplayer features. This was to retrieve crucial insights of the issues in the development of multiplayer games along with the mitigated knowledge gaps of the members of the team.

Following the previous review of the current state of the art and the literature on multiplayer game development, the following research questions were posed:

RQ1: What challenges, knowledge gaps, and risks do novice development teams face when working on concepts for competitive multiplayer games?

RQ2: What multiplayer development pipelines, workflows, and tools exist that can support novice teams, and how can a generic multiplayer pipeline be developed for use by teams working on a self-directed multiplayer game concept?

RQ3: How will a generic multiplayer pipeline impact on the design process and knowledge development of a novice team, what are its limitations, and how will the end product be received by players?

To address these research questions, a research methodology composed of three steps was developed (as shown in Figure 5):

(i) Development of generic multiplayer pipeline (reported in 4.1)

(ii) Game development case study (reported in 4.2)

(iii) Play testing of the original multiplayer game artefact (reported in 4.3)

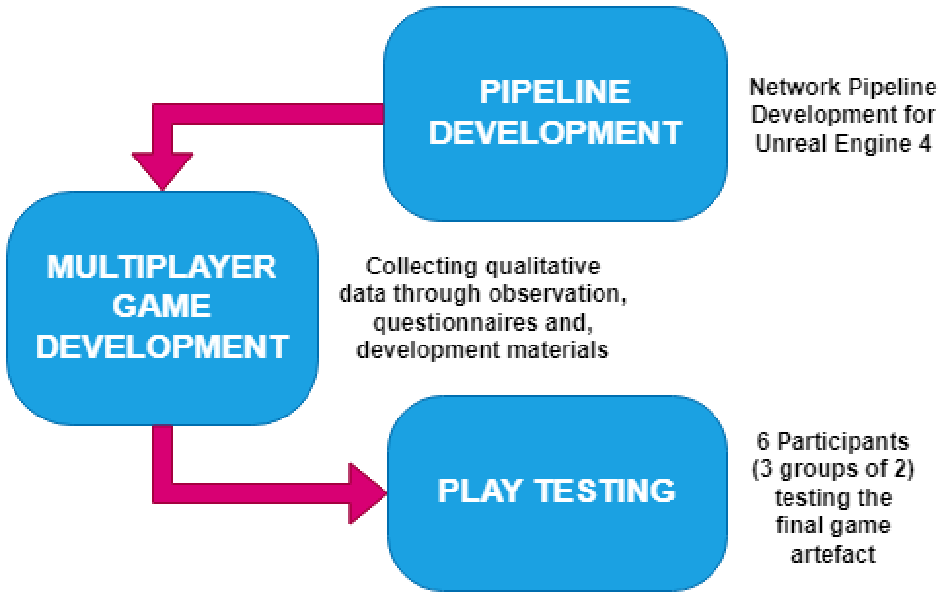

PIPELINE DEVELOPMENT
Network Pipeline Development for Unreal Engine 4
MULTIPLAYER GAME DEVELOPMENT
Collecting qualitative data through observation, questionnaires and, development materials
PLAY TESTING
6 Participants (3 groups of 2) testing the final game artefact

Figure 5. Methodology Flow Diagram

Figure 6 illustrates the link between the research questions and the procedure along with the literature that has been reviewed. Initially, a review of the technical details of the multiplayer game development was carried out with emphasis on the technical challenges of the development which provided useful insights for research question 1. Then, a middleware review was carried out to find out possible tools, pipelines in game engines available to develop a generic multiplayer framework for the team to use it to develop a multiplayer game which helped in answering research question 2. The response of the questionnaires from the team are each linked with the multiplayer game development which provided data to help in answering research question 1. Finally, the framework used by the team and the response received by the testers helped in answering research question 3.

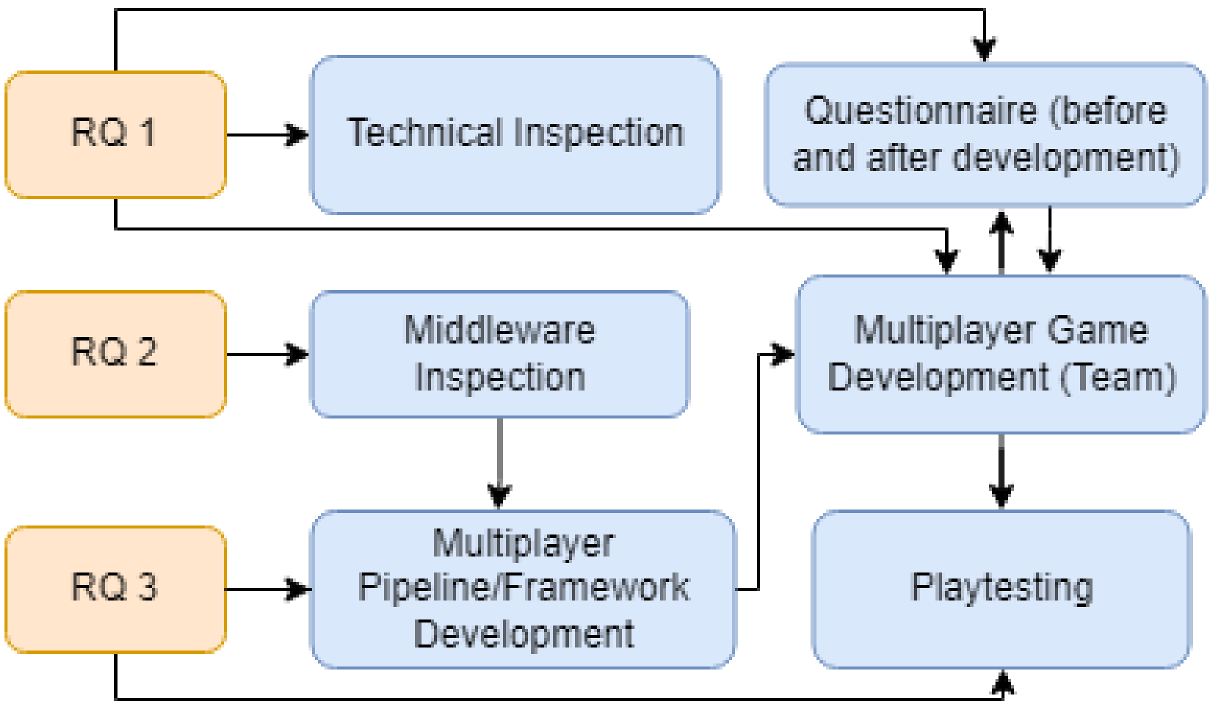

RQ 1
Technical Inspection
Questionnaire (before and after development)
RQ 2
Middleware Inspection
Multiplayer Game Development (Team)
RQ 3
Multiplayer Pipeline/Framework Development
Playtesting

Figure 6. Link between Research Questions and Procedure

To answer the research questions stated above, the plan to fulfil the objective has been described in the following sections.

4.1 - Deciding a Genre to Build the Game Upon

The first phase of the research involved the development of a generic multiplayer game development pipeline that could be used by a novice development team. This research was informed by the literature on existing tools, technologies, and developer knowledge as discussed in section 2.2, and sought to address the second aspect of the research question 2 identified above.

4.1.1 - Choosing a game engine

The extensive review earlier covered a number of game engines with their advantages and disadvantages along with the multiplayer support in the engines. Most of the sources suggested that Unreal Engine would be an ideal choice for multiplayer games with extensive integrated multiplayer support without any need to have external tool or plugin to use. Some extra sources such as Game Ace Creative Studio compared the two most popular game engines in the market, i.e., Unreal Engine and Unity and clearly concluded Unreal Engine as the better engine for multiplayer games owing to the fact that it offers robust networking tools that lets the developers achieve peak performance (Game Ace, 2021).

The combination of blueprints and C++ scripting provides the developers with multiple departments to collaborate easily. The only drawback of having C++ compiler is that the compilation time is slower. Although, blueprints compile much faster than C++ scripts in Unreal Engine (Forsythe, 2021) and developers can use it with the provided integrated multiplayer support to use it for implementing multiplayer features. It is more optimal to blend C++ and blueprints for the projects to achieve implemented mechanics. Therefore, it was concluded that Unreal Engine would be the starting point for this research to first develop a generic multiplayer framework for the team to use it for further development of a multiplayer game.

4.1.2 - Choosing a game type/genre

The first step was to identify a game type to serve as the focus of the multiplayer development pipeline. Considering the limitations of the assets available and the need to account for the skillset of a novice team, it was deemed appropriate to ensure that the game type was limited in complexity. Choosing a complex multiplayer type could result in extra responsibilities for the team, and thus have an impact on the quality of the results.

Therefore, a turn-based genre rather than a real-time multiplayer was selected so that this mitigated the risk of technical issues for the team. Turn-based multiplayer games are less challenging to develop than real-time multiplayer games (Apple Developer, 2016). With turn-based games, the team would not have to deal with any client-side prediction or lag issues that are typically found in real-time multiplayer games.

4.1.3 - Generic Multiplayer Pipeline Development

Development research was then carried out with a view to creating a multiplayer framework within Unreal Engine 4. The purpose of the generic multiplayer pipeline was to support a novice team in the creation of a turn-based multiplayer game where the team would retain creative freedom over design decisions.

This method included review of existing tools, pipelines, and frameworks that support a generic multiplayer architecture to develop any multiplayer game upon. Some of the popular tools and repositories found that satisfied this criteria were, Advanced Sessions Plugin utilised to develop Cardinal Menu System (Metahusk, 2016), Advanced Server Manager (Reverse Winter Studios, 2021), Puzzle Platform repository (Hunt, 2021).

All of the plugins or frameworks found above were useful, although all had identified strengths and weaknesses. Cardinal Menu System provided by Metahusk was a versatile framework with a lot of useful features such as server browser (with steam and LAN support), internet and LAN hosting menu, steam friends list with avatars, customisable graphics settings, player control mapping with controller support and game settings. The only issue with this framework is that its repository is not thoroughly linked on its official website but only on GitHub. On top of that, the project is outdated and not concurrent with latest Unreal Engine versions such as 4.26 and 4.27.

Advanced Server Manager was also a versatile plugin with extensive features such as creating and finding match, player queueing, server browser, region search, adjustable settings, access to all servers and easy customization but it was a proprietary plugin published by Reverse Winter Studios costing around £ 90.

Puzzle Platform repository was a free to use repository with basic minimal features such as menus, hosting and joining of sessions (LAN and steam), server browser, etc. On the other hand, it missed some important features such as the in-game menu to end and clean up server sessions (auto clean up), chat system, friend server sessions, etc.

On review of existing documentation and solutions for multiplayer pipelines (including several assets on the Unreal Engine marketplace and online repositories) it was assessed that some were blueprints scripting based while some were proprietary and not free to use. Some frameworks mentioned had some limitations such as the engine version support and no chat system. While the pipeline provided by Reverse Winter Studios is the best among the mentioned pipelines with updated support in the latest engine versions, the others lack some features or are outdated with the latest engine versions. Puzzle Platform repository could be the best free to use plugin for this research, but it needs have its feature voids filled in order to be an optimal and fully functional multiplayer framework. Furthermore, the free to use framework such as the Advanced Sessions Plugin is mainly for blueprints scripting and is not very customizable, this is not an optimal solution to develop a multiplayer game on. An elegantly developed multiplayer game in Unreal Engine should be well balanced between C++ and blueprints with C++ objects (variables, functions) exposed to blueprints so that the designers can make use of those functionalities. Additionally, it is not ideal to develop the entire project in blueprints while parallelly working with multiple departments since it is harder to maintain the version control with blueprints only.

The first phase of the research therefore involved the development of a new pipeline, incorporating existing resources and knowledge with bespoke steps and producing documentation to assist a development team. The outcomes of this process are reported in chapter 4.

Two extremely useful sources from Cedric Neukirchen (Neukirchen, 2021) and Unreal Engine documentation (Unreal Engine, 2021a) have provided an insight of a robust built multiplayer game architecture.

Regardless of the type of genre chosen to build the game upon, the multiplayer game framework in Unreal Engine was supposed to be independent of the genre. Additionally, the framework did not depend on whether the game is a real time multiplayer or turn-based multiplayer. The documented resource provided by Cedric Neukirchen (Co-Founder of Salty Panda Studios) well described the complete working architecture of client-server model in Unreal Engine with detailed steps.

The document further helped in creating lobby session functionality along with a basic placeholder user interface and basic features such as custom server creation, joining, etc. Additionally, a basic game architecture with high-level managerial classes such as *Game Mode Base*, *Game Instance*, etc., were written to synchronize the Game States following the Object-Oriented Design.

4.2 - Game Development case study

The primary research method to address the research questions of the project was a game development case study, where a team of novice developers were tasked with producing a multiplayer game supported by the multiplayer game framework.

This research was conducted in collaboration with undergraduate students at Abertay University of game development belonging to disciplines such as Programming, Art, Design, and Production, who were studying for their third-year module ‘Professional Project,’ which tasks students working with multidisciplinary teams to deliver game prototypes and associated materials according to set briefs.

A well-documented, extensible, maintainable, and readable framework was laid out as foundation with the help of ‘Unreal Engine 4 Network Compendium’ and several other sources such as blog and YouTube channel of Sneaky Kitty Game Dev.

But to develop a solid online multiplayer game artefact, a well composed student team with members of multiple disciplines was deployed.

The development of the online multiplayer game artefact was the crucial aspect for answering the research questions since it helped in addressing both of them (research questions 1 and 3) by finding potential problems that the team ran into while working on a multiplayer game with the generic multiplayer pipeline developed. Furthermore, the success/failure of the project eventually helped in answering the research question 3 addressing the utilising of a generic multiplayer pipeline to support and impact the design process and knowledge development of a novice team.

An important part of this research was taking input and collecting data from the team members and their workflow during the development of the game. This was carried out with the help of initial input from the members through documents such as the Game Design Document (GDD). Going on further, a scheduled plan to note down weekly meetings log, daily work updates log in the team's discord server was laid out. The team was prompted to adopt the agile development methodology or scrum which proved to be fruitful in the long run as it is widely practiced in the industry (Lemarchand, 2021). This also helped in collecting data regarding any challenges and problems the team ran into while developing the artefact. A production plan covering milestones such as the first playable demo, Alpha, Beta, and the Gold Master, was laid out to ensure the integral progress of the team.

Additionally, periodic testing of the builds after developing playable features was carried out to find any game breaking bugs such as level A or level B bugs. This process was carried out via collaboration between third year undergraduate DES310 students and first year students. The DES310 module is organised in such a way that the builds of the games developed by the DES310 students are tested by the students of first year GDP (Game Design Production) undergraduate program before every vital milestone. The students from the first year have a Quality Assurance and User Experience module, DES103. These particular testers weren't part of the research but part of the module DES103 collaborating the two batches of the undergraduate program. This also ensured periodic bug fixing and removal of unwanted issues from the project.

4.2.1 - Participants

The participants were a team of 9 students enrolled in the following programmes–

1. Game Design and Production
2. Computer Arts
3. Computer Games Technology
4. Computer Games Application Development

The participants were organized into a team by the module leaders and selected for the brief of development of turn-based multiplayer game. All participants were above 18 years of age and were provided with the Participant Information Sheet and Research Consent form.

4.2.2 - Role of the Researcher

The role of the researcher in this project was initially providing the team with generic multiplayer framework and interacting closely with the team to observe and note issues and challenges faced by the team. It is important to note that the researcher was not a core member of the team. The researcher was at a critical distance from the team so that he could credibly observe and ask questions about the development of the game. Additionally, the researcher adapted the framework to tackle specific issues related to the game while maintaining the network layer overall.

4.2.3 - Materials

The materials required and utilised for this research method were the software/game engine for the development of the game, the repository of the framework, documentation provided by Cedric Neukirchen, some additionally links referred for the development of the framework and the assets provided freely on Unreal Engine marketplace for free along with some free audio assets.

1. Game Engine/Software : Unreal Engine 4.26
2. GitHub Repository : https://github.com/animeshsharma1996/MbR-eSports-Multiplayer (Appendix A)
3. Multiplayer Framework : Standard Unreal Engine 4 framework using UE4 Network Compendium (by Cedric Neukirchen - https://cedric-neukirchen.net/Downloads/Compendium/UE4_Network_Compendium_by_Cedric_eXi_Neukirchen.pdf)
4. Additional Resources/Links :

    a. https://docs.unrealengine.com/4.27/en-US/ProgrammingAndScripting/Online/Steam/
    b. https://mediacbl.com/menu-widget-unreal-engine/
    c. https://www.youtube.com/c/SneakyKittyGameDev

5. Assets/Resources :
    a. Free Assets provided by Epic Games on UE marketplace
    b. Free Sound assets (https://mixkit.co/free-sound-effects ; https://www.zapsplat.com)

4.2.4 - Procedure and Data Collection

The procedure and data collection were carried out in three ways in the form of questionnaire, researcher observation, and development data.

1. **Questionnaire:** A questionnaire was given to the participants before and after the development of the game. Questionnaire was given before, to assess their skills, knowledge, and understanding of multiplayer game design and development, etc. Questionnaire was given after, to interrogate what they learned from working on a multiplayer project, extent to which framework helped for the development, strengths, weaknesses, improvements needed, their reflection etc. The questionnaire given to the team before and after the production of the game are mentioned in Appendix D.1 and Appendix D.2 respectively.
2. **Researcher Observation:** Observation notes were collected by participating in two meetings per week. Meeting 1 was a general project update meeting, meeting 2 was a targeted meeting focusing on the framework and its utilisation in the project. The data was collected in the form of anonymized notes.
3. **Development Data:** Materials were gathered from development logs, documentation, source code, build, other pitching/evaluation materials. The communication server of team's discord was utilised for coherent communication within the team. This helped in noting down both minor and major challenges faced by the team and individuals anonymously.

4.2.5 - Timeline

This phase of the research took place between January 31st and May 17th, 2022. Overall development took place over 14 weeks, with final work submitted in the fourteenth week. Teams scheduled two meetings per week that the researcher participated in; one weekly update with the module tutor which covered general project progress, and one targeted meeting with the researcher where the team focused on issues specifically related to development and implementation of multiplayer features utilising the framework.

4.2.6 - Analysis

The data collected before the development of the game, during the development and after the development of the game was analysed. The three types of data collection were combined and analysed to determine the strengths and weaknesses of the multiplayer framework along with the extent to which the team was benefited by the support and the knowledge gaps of the technicalities of the multiplayer game development. Additionally, the notes taken during development determined clear barriers encountered during development of the game, the risks taken and mitigated by the team.

4.3 - Testing and Evaluation

To assess the success of the multiplayer implementation and determine whether the developed game satisfied player expectations of online multiplayer game, the last research method was to conduct playtesting with participants. This method sought to address research question 3, while also offering an opportunity to explore questions more specifically targeted towards the player experience using the framework's features in the game, and the feasibility of novice teams working towards the development of competitive play experiences in this context.

4.3.1 - Participants

6 Participants for three test groups of 2 were recruited from the community discord servers to test out the game and report the player experience. The community discord servers had several candidates for testing the game because of their

experience in playing competitive games on a regular basis. The potential candidates were prompted to take part in research and 6 candidates agreed to participate in the research. These participant testers are separate from the first-year testers who tested the game before each vital milestone. These 6 testers are participants of this research with given consent form before agreeing to take part in the research. The testers belonged to diverse regions and played against each other in their respective groups. The data collected based on the questionnaire is mentioned in detail in Appendix D.3.

4.3.2 - Materials

The materials required and utilised for this research method were the turn-based multiplayer game artefact developed by Stormy Night Games and the testers' questionnaire specifically targeting towards the framework utilised for the development of the game.

1. Turn-based multiplayer game (Appendix E)
2. Multiplayer Game Testing Questionnaire (Appendix D.3)

4.3.3 - Procedure

The participant testers were be given an opportunity to test the fully functional multiplayer turn-based game developed by the student development team as outlined in 4.2. As explained earlier in 4.3.1, the 6 participants were recruited from the community discord server, and they agreed to take part in the research and test the game. They were provided with the copy of the game build and clear instructions on how to play the game. The testers were then asked to complete the questionnaire after they tested the game. The questions were related to the generic multiplayer framework utilised for developing the game which contained features such as menu interface for creating and finding custom server sessions, steam friend sessions, chat system, etc.

4.3.4 - Analysis

The data was analysed after the testers responded to the questionnaire given to them after testing the game. The data helped in determination of the player experience of the game developed based on multiplayer framework. The questions tracked back to the framework used as a foundation for the game. This in-turn reported the adequateness or inadequateness of the framework and helped in answering research question 3. Each question was related to the player experience when using each feature of the framework and how convenient or inconvenient it was to use the framework's features.

5. Results/Findings

5.1 – Multiplayer Framework Development

To carry out the development of a generic multiplayer framework for development of any genre of multiplayer game, the basic minimum features required for multiplayer games were implemented. This helped in answering research question 2.

The implementation carried out was to find out, how can a generic multiplayer framework be developed to support a student team with limited knowledge working towards the production of a multiplayer game. As already stated earlier, Unreal Engine was chosen as the game engine which provides support for steam subsystem. Additionally, Unreal Engine supports multiplayer development and gives the developer an opportunity to develop features involving customized server sessions (creation, update, joining, and destruction, etc.).

The development team was to be provided with the basic framework as a foundation to build the game upon. Based on the common features that are present in the multiplayer games in the market, a list was made to implement features in chronological order.

The list of common features implemented to aid the development of the game are as listed follows –

1. Menu Interface
    a. Main Menu
    b. Custom Host Menu
    c. Join Menu
    d. In-Game Menu
2. Creating and Finding Server Sessions
3. Steam Support
4. Creating and Finding Friend Sessions
5. Registering and Unregistering players
6. End and Destroy sessions
7. Chat System

To fully understand how the framework was developed, there was a need to understand the life cycle of a server session for a listen server as explained by Cedric Neukirchen (Neukirchen, 2021) and on the webpages, Epic wiki (Olsson, 2015) and Unreal Community Wiki (Gamibt, 2021). The life cycle of a server session is shown in the flow diagram (Figure 7). The server session is created by a client or a player which is also the host (as how it works in a listen server). Once the server is created, it needs to be initiated or started through *SessionInterface->StartSession(FName sessionName)* function call. After the start of the server session, the host player needs to be registered with the session so that the session shows a player in the lobby when the session data is retrieved for the purpose of the server browser information. After the server session is running, the other clients or players can find the server session on the server browser. If the session is found, the clients can join the session to get inside the lobby and further get registered with the server session.

The server settings can be modified with the help of *UpdateSession* function if the configuration of the server needs to be modified. The host player can also close the lobby by the ending the session (on the host and all the clients which is an RPC function) and further cleaning the server session by destroying the session.

It is crucial to understand that each of the functions mentioned in the light blue colour (Figure 7) are functions that have post function call delegates and are supposed to be linked with custom delegate functions inside the Game Instance class. Each time one of these functions are called, their respective post functions (for e.g., *OnStartSessionCompleteDelegate*) are called when the primary function is successfully run. These post function calls can be customised or overridden with custom delegates to implement new functionalities.

Once the server session life cycle was thoroughly understood, there was a need to link or bind the server session functionalities with a user interface or in specific terms, menus. An optimal and elegant technique to design and implement a UI manager or menu system has been explained on the webpage by Media CBL (Media CBL, 2021). Additionally, the youtuber Sneaky Kitty Game Dev (Sneaky Kitty Game Dev, 2020) has explained one of the optimal ways of linking or binding the session functionalities with the user interface through both C++ and Blueprints.

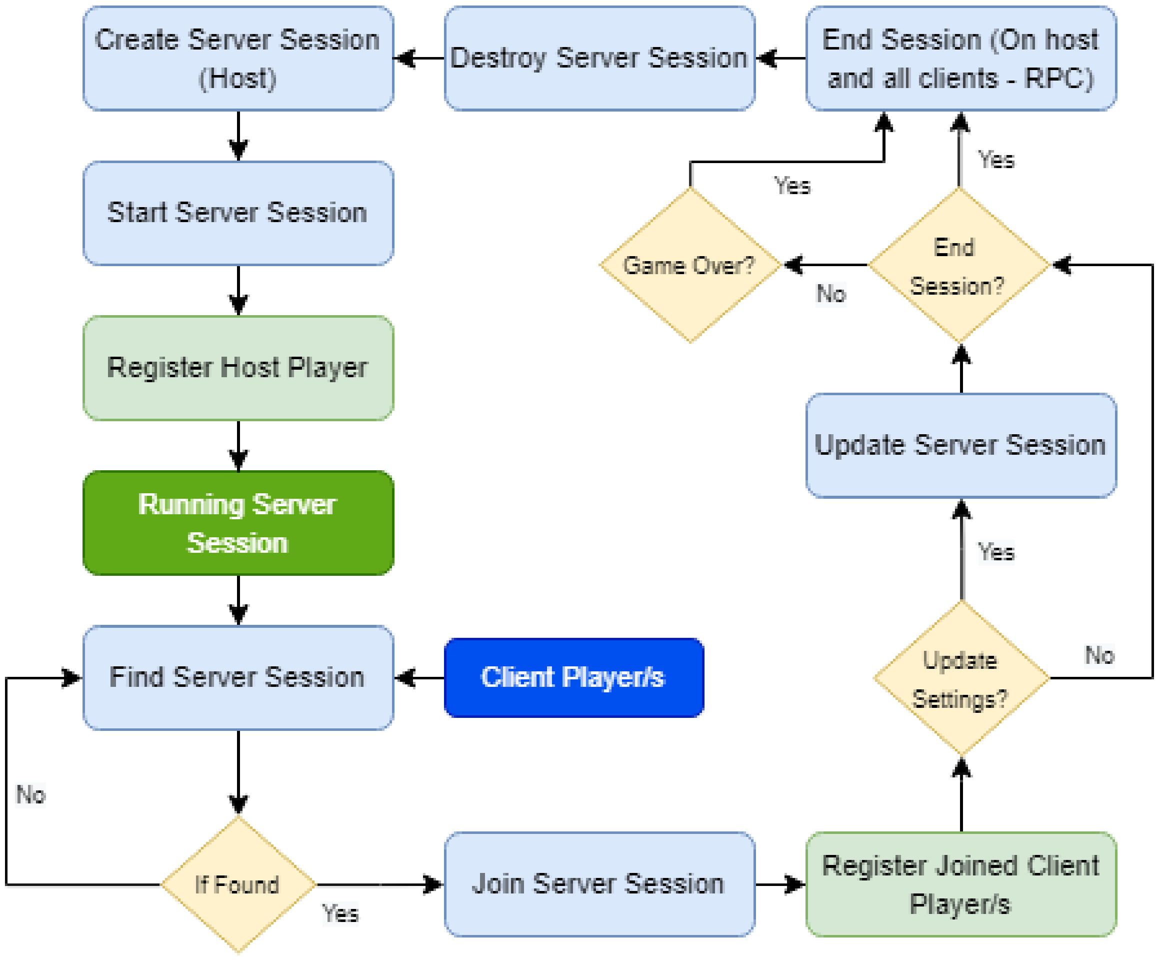


Figure 7. Server Session Life Cycle

The task at hand was to implement the whole menu system in C++ and link or bind it with the session functionalities. This was done by creating a Main Menu Widget class from the base C++ class. The Main Menu widget class contains the button variables, text block variables, input block variables composing the whole main menu screen. Each variable is bound to its design element in the child blueprint class. The main menu screen is shown in figure 8.

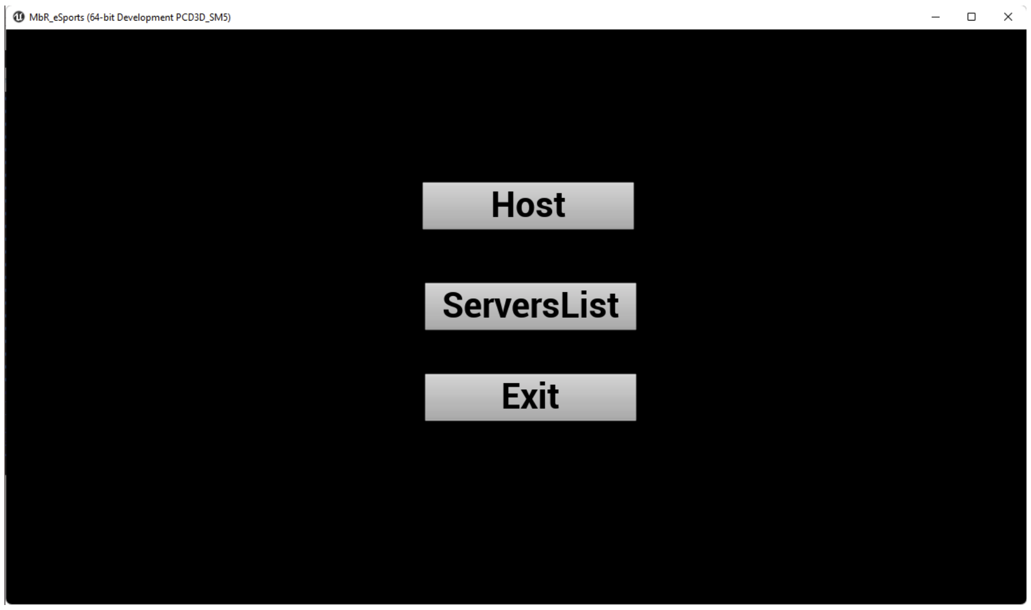


Figure 8. First iteration of the framework main menu

The first iteration of the main menu only had the programmer art (Figure 8). It was modified and reskinned with the help of some button assets, text fonts, and background to give it a professional visual (Figure 9). The main menu blueprint class's designer layout has several layers to allow transition between screens. The host screen (Figure 10) and server browser screen (Figure 11) are all linked together with the main menu through the buttons. It can be seen from figure 11 that the number of players shown in the hosted lobby by a host is zero. This is because the *RegisterPlayer* function wasn't initially implemented. This bug was then fixed by inserting the register player functionality for the host and also the clients that join the sessions in the correct sequence as shown in the life cycle of a server session in figure 7. The players were also supposed to be unregistered when they leave the lobby or the server session which was carried out with the help of *UnregisterPlayer* function. Additionally, the server information row as displayed in figure 11 is actually a User Widget created under the name, Server Slot Widget which was implemented to contain the information of created server to be displayed on the server browser. The whole menu system was governed and manager by the class UI Manager derived from the base HUD class to initialise the lower hierarchical classes in the beginning of the level initialisation.

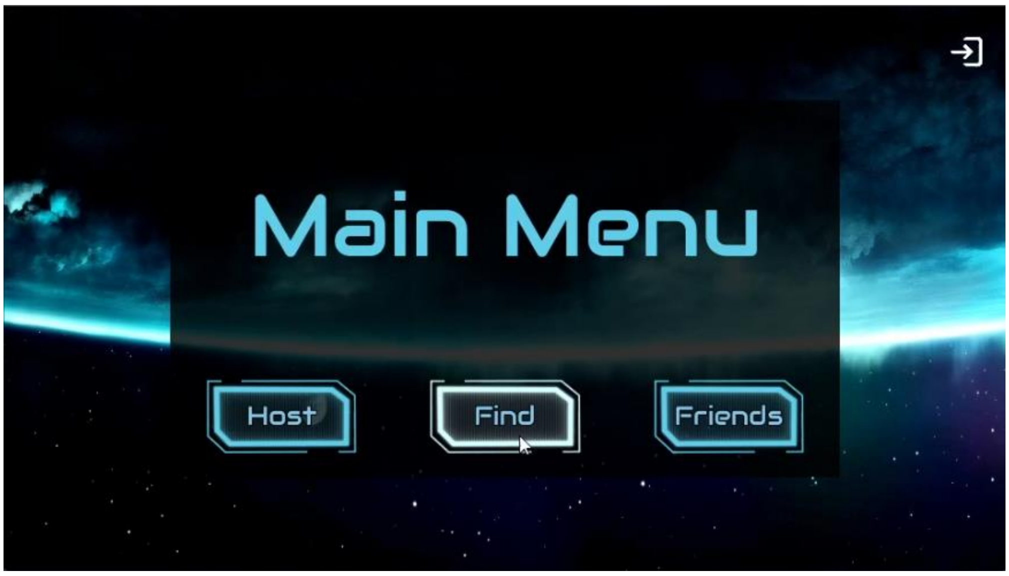


Figure 9. Modified version of the framework main menu

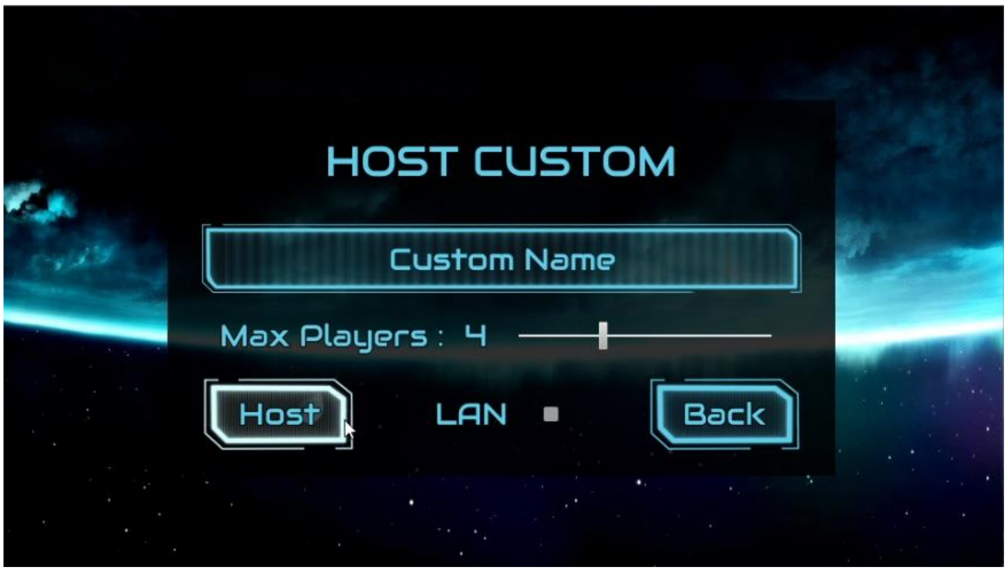
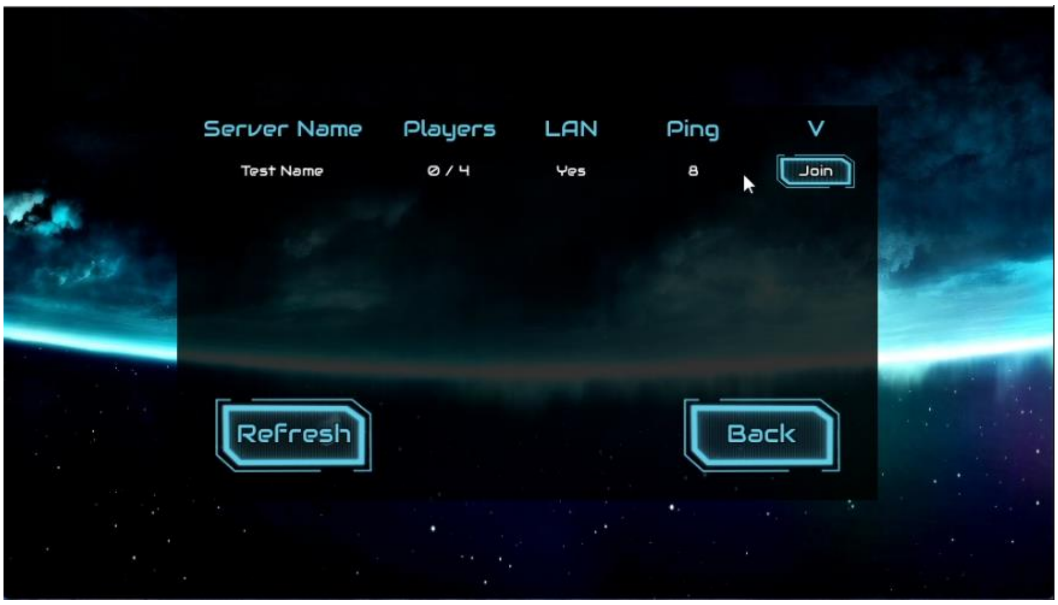


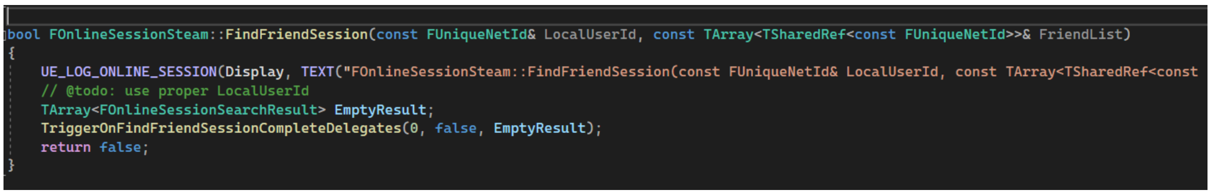
Figure 10. Custom host menu screen



Server Name	Players	LAN	Ping	V
Test Name	0 / 4	Yes	8	Join

Figure 11. Server browser menu/Join menu screen

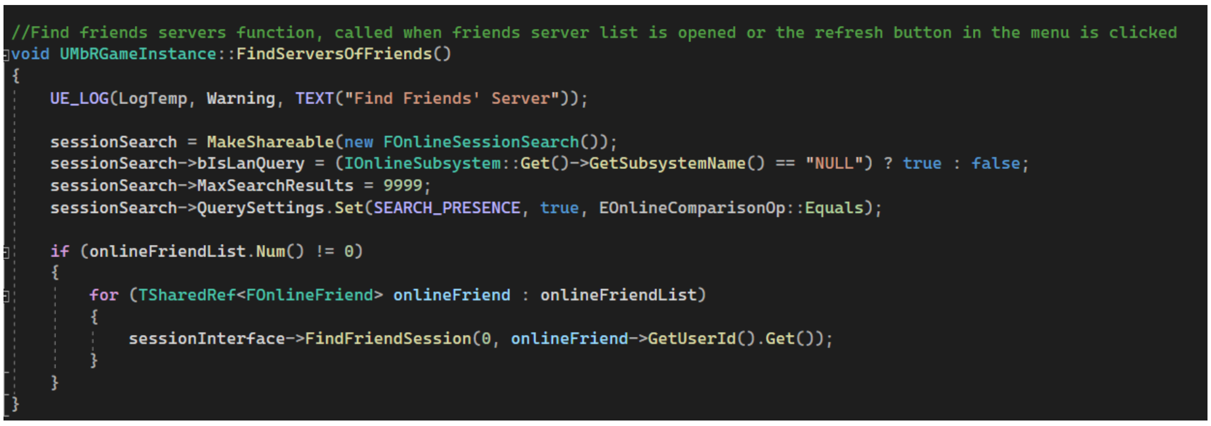
The friend button shown in figure 9 is used to retrieve and the servers hosted by friends on steam. Implementing the friend's session functionality was the most challenging part of the framework. This is because Unreal Engine's documentation (Unreal Engine, 2021b) mentions the friend session functionality, but it is not actually implemented in the lower engine level in the source code (Figure 12). The function always returns false and doesn't find a friend session.



```
bool FOnlineSessionSteam::FindFriendSession(const FUniqueNetId& LocalUserId, const TArray<TSharedRef<const FUniqueNetId>>& FriendList)
{
    UE_LOG_ONLINE_SESSION(Display, TEXT("FOnlineSessionSteam::FindFriendSession(const FUniqueNetId& LocalUserId, const TArray<TSharedRef<const
    // @todo: use proper LocalUserId
    TArray<FOnlineSessionSearchResult> EmptyResult;
    TriggerOnFindFriendSessionCompleteDelegates(0, false, EmptyResult);
    return false;
}
```

Figure 12. Find Friend Session function in Unreal Engine 4.26 source code

This problem was solved with the help of retrieving the friends list (from steam) and using it to iterate and find friend session of each friend, if the friend actually hosts the server (Figure 13).

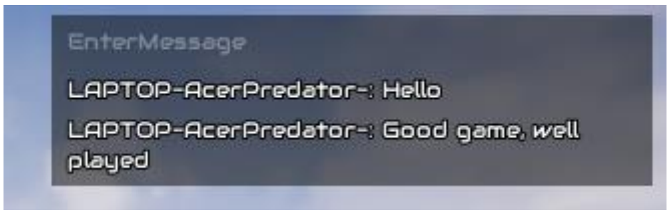


```
//Find friends servers function, called when friends server list is opened or the refresh button in the menu is clicked
void UMbRGameInstance::FindServersOfFriends()
{
    UE_LOG(LogTemp, Warning, TEXT("Find Friends' Server"));

    sessionSearch = MakeShareable(new FOnlineSessionSearch());
    sessionSearch->bIsLanQuery = (IOnlineSubsystem::Get()->GetSubsystemName() == "NULL") ? true : false;
    sessionSearch->MaxSearchResults = 9999;
    sessionSearch->QuerySettings.Set(SEARCH_PRESENCE, true, EOnlineComparisonOp::Equals);

    if (onlineFriendList.Num() != 0)
    {
        for (TSharedRef<FOnlineFriend> onlineFriend : onlineFriendList)
        {
            sessionInterface->FindFriendSession(0, onlineFriend->GetUserId().Get());
        }
    }
}
```

Figure 13. Implementation of Find Server Session of Friends (Steam)

Furthermore, the framework was required to have an in-game menu which would allow the host to end the session on the host server and all the clients and also allow the clients to leave the lobby mid-game. This was carried out with the help of a multicast RPC (Remote Procedural Call) inside the Player Controller class which basically ends the server session on all clients and the server (if called from the host which is the server in this case) or only on the respective client if called from the client.

The Player Controller class is an actor class which receives input from the player and is replicated. This type of class follows the definition of RPC functions and thus was also utilised in implementing the chat system. The chat system implementation was carried out with the help of a text box layout containing the rows for text messages sent by the player and an input text box to send these messages (Figure 14). In this way, the chat messages upon typing are sent to the server with the help of a server RPC and then sent to all the clients with the help of the player controller iterators which contained all the player controllers of each connected clients. The chat box gets expanded and reduced in size depending upon the number of active messages being sent. If the messages are increased in number, the chat box expands to a certain threshold and after certain time delay, it gets reduced in size.



EnterMessage
LAPTOP-AcerPredator-: Hello
LAPTOP-AcerPredator-: Good game, well played

Figure 14. Chat system visual representation

Finally, the chat system was integrated with a toxicity prevention system which was a relatively easy task. The toxicity prevention system basically regulates the messages sent by client and matches against a list of offensive words, if the messages have any of these words, it will send a warning to the client and won't send the message to all clients. The client will be kicked and disconnected after 3 warnings.

The framework follows the definition of generic architecture to serve as a base foundation for any multiplayer game, i.e., any genre of multiplayer game can be developed using the framework and the features implemented for the framework will always be required for any multiplayer game. Although, it is worth noting a crucial point here, that the framework creates sessions of a listen server which means the host is a client and also the server. Usually in multiplayer games published by authoritative publishers have dedicated servers where none of the clients is the host. That is how matchmaking works. But in this case, the game was developed by a student team and thus the framework did not require the need to have dedicated servers. The development of the framework required around 6 to 8 weeks of initial time with additional time involved in updating framework during development of the game. The deployment of the framework had some important key takeaways observed –

1. The framework was developed to fill in all the gaps of features left void by the free to use framework on the Puzzle Platform repository (Hunt, 2021) with features such as friends' session, in-game menu to end server session and an elegant chat system.
2. The framework isn't as advanced as the Cardinal Menu (Metahusk, 2016) in terms of customisation of settings (graphics, controller-support) but it has options for developers to customise and modify the framework with extended blueprint support derived from C++ classes along with the additional chat system, easy access, and compatibility with the current Engine's versions.
3. As mentioned before, Advanced Server Manager would be the best plugin to utilise for the multiplayer game development, but it is proprietary and costs around £ 90. The framework developed for this research is a good free to use replacement for Advanced Server Manager which also doesn't have the chat system.
4. The framework does not implement dedicated server which is a limitation for overall authoritative control for the developers, but this isn't really needed for novice developers or student teams.

In summary, the framework is on its own is customisable, expandable, and modifiable following Unreal Engine's object-oriented design with applied optimal programming principles and higher hierarchical classes governing the loop for the running of the framework. The overall adequateness of the framework was clearly determined through the usage by the team and playtesting by the testers as observed and reported in the sections below. The framework was modified during development (of the game) to implement the required and missing features to fulfil the criteria of a complete generic multiplayer framework, this has been reported in the production section below.

5.2 - Production

The findings of the game development case study have been presented in two ways, the observation notes about the production process in a timeline and second, through the questionnaire data provided by team before and after the development of the game. The production or the development of the turn-based multiplayer game provided data on the issues/challenges faced by the team while developing a fully functional multiplayer game. This directly helped in answering research question 1. The challenges or the barriers faced by the team have been described in chronological order in a timeline. Additionally, the timeline is mentioned in a table format in Appendix C.

The crucial point here is to note the initial game overview proposed by the team upon the commencement of the project and how the iterations have modified the structure of the game throughout the development cycle. The initial game mechanics proposed have been mentioned below -

- A turn-based, online multiplayer game of 4-8 players.
- The main objective was to get to the other side of the map and destroy the enemy base while trying to prevent your own base from being destroyed.
- Ways of obtaining points were by killing bosses which will spawn throughout the map, killing enemies, completing class objectives, and surviving the day/night cycle.
- Players can also win the game by having the most points within so many turns/a specific time.
- Players will be able to choose from 4 different characters with different stats.

As the team commenced the production of the multiplayer game, one of very first issues encountered was the syntax of the Unreal Engine C++ language. Unreal Engine C++ is quite a different language than the original C++ with the variables and classes starting with a prefix such as struct data structure is written as FStruct, array data structure is written as TArray<> and it is case specific with each class. The programmers initiated the project in Unreal Engine 4.27 in C++ where it was supposed to be initiated in Unreal Engine 4.26 as that was the version in the systems in the campus of the University. Switching back to 4.26 created delays in production with initial work to be redone.

During the first few weeks of production, the procedure followed by the programmers involved one programmer handling the networking of the movement of a character on a grid system, while the other two programmers handling minor tasks and waiting for the other programmer to finish the networked movement. Since following such a procedure can cause delays in production and they were already working in C++ which can be a tedious task with slower compile time, the programmers switched to Blueprints for faster development, compile time, and fewer engine crashes.

Although the trade-off here is that integrating progress can be complicated since version control does not support merging of blueprint files as they are binary files, and it is a risk of having worse performance of the game. But to speed up production with faster compilation time and faster debugging, the team resorted to this methodology.

The team also required additional features from the framework along with some mandatory features such as in-game menu, chat system and friend's server session for global servers in order for the multiplayer game to be generic and globally playable.

The initial prototype only saw mechanics implemented such as the assignment of teams to the players, turn-based networking system and the joining of players in the same server session/lobby. The prototype was missing features such as turn-based movement on a grid system.

In the first playable demo phase, the team implemented the missing features and full-game loop along with the A-star pathfinding for correct tile to tile movement. One crucial feature was the turn-based replicated movement and the team was faced with the issue of the movement being very incoherent and clunky. A workaround was utilising the inbuilt replicated movement (utilising *AddMovementInput*) and customising it to have the turn-based movement upon clicking of the mouse button on a desired tile.

This feature hindered progress in production and caused a delay of 1-2 weeks. Once the issue was fixed, the team was able to quickly implement the remaining features such as the combat system with attack, damage intake and death of the player character, RNG (Random Number Generator) dice sprite, HUDs (Heads Up Display), and menus.

Additionally, there were some issues encountered while updating the framework such as implementation of steam friend's sessions. Unreal Engine has documented the functionality, but it is not fully implemented in the source code (discussed in section 5.1). The other issue was the functionality of *End Session* since most of the code of networking sessions is supposed to be written in Game Instance class, but it is not a replicated actor and thus an RPC (Remote Procedural Call) cannot be called inside it. Thus, the Player Controller was needed to implement this functionality.

Tackling the issues towards alpha and implementing features saw most of the crucial features implemented in the game, the delays in production saw some features cut from the potential implementation such as the number of players, the game was made to be one vs one, the day-night cycle, bosses' system, and win condition by points were scrapped.

Post alpha was the integration of the framework update including the chat system and toxicity prevention system in chat. The chat system introduced a bug with Player Controller class and the end session was not thoroughly removing the player pawn from the scene. This was because the Player Controller class's *Begin Play* should be called from the blueprint rather than C++ if the blueprint is chosen as the main class in Game Mode class's settings.

Beta only had two weeks after alpha and to make the game content and feature complete, the framework update was integrated with additional integration of audio, art assets (UI, animations) and replaced 2D dice with 3D dice.

Finally, the last framework update was integrated with a fully functional in-game menu and registering and unregistering of players for correct update of the number of players in the lobby. The 3D dice had to be removed from the game due to the lack of time and the dice causing unintended break in the game.

In summary, the production process of the turn-based multiplayer game provided critical assessment of the utilisation of the framework and the framework itself. The key takeaways from this case study about the framework are outlined as follows –

1. The framework saved around 6-8 weeks of time (as mentioned in 5.1) of the team to setup basic multiplayer functionalities which was effectively handled by the framework. Additionally, the team didn't have to address and fix any issues related to the framework during development. In short, the team was free of any development work required for the framework.
2. The framework was fully independent of the type of the game and thus any modifications required in the framework didn't interfere with the mechanics and features of the game.

3. The framework had to be iterated on several times during development to include the missing features such as friends' servers, in-game menu for ending and destroying session, registering-unregistering players, and the chat system.
4. The implementation of the framework in pure C++ allowed the team to develop the game independently and implement mechanics in blueprints without needing to worry about any break or disruption in the framework or the game.
5. Any changes, modifications carried out in the implementation of the gameplay or encounter of any breaks in the code base of the main game was independent of the framework.
6. The framework didn't directly impact or fill the knowledge or skill gaps of the members of the team (as observed in 5.3) but only to a degree of understanding basic multiplayer architecture along with the common classes utilised for development in Unreal Engine.

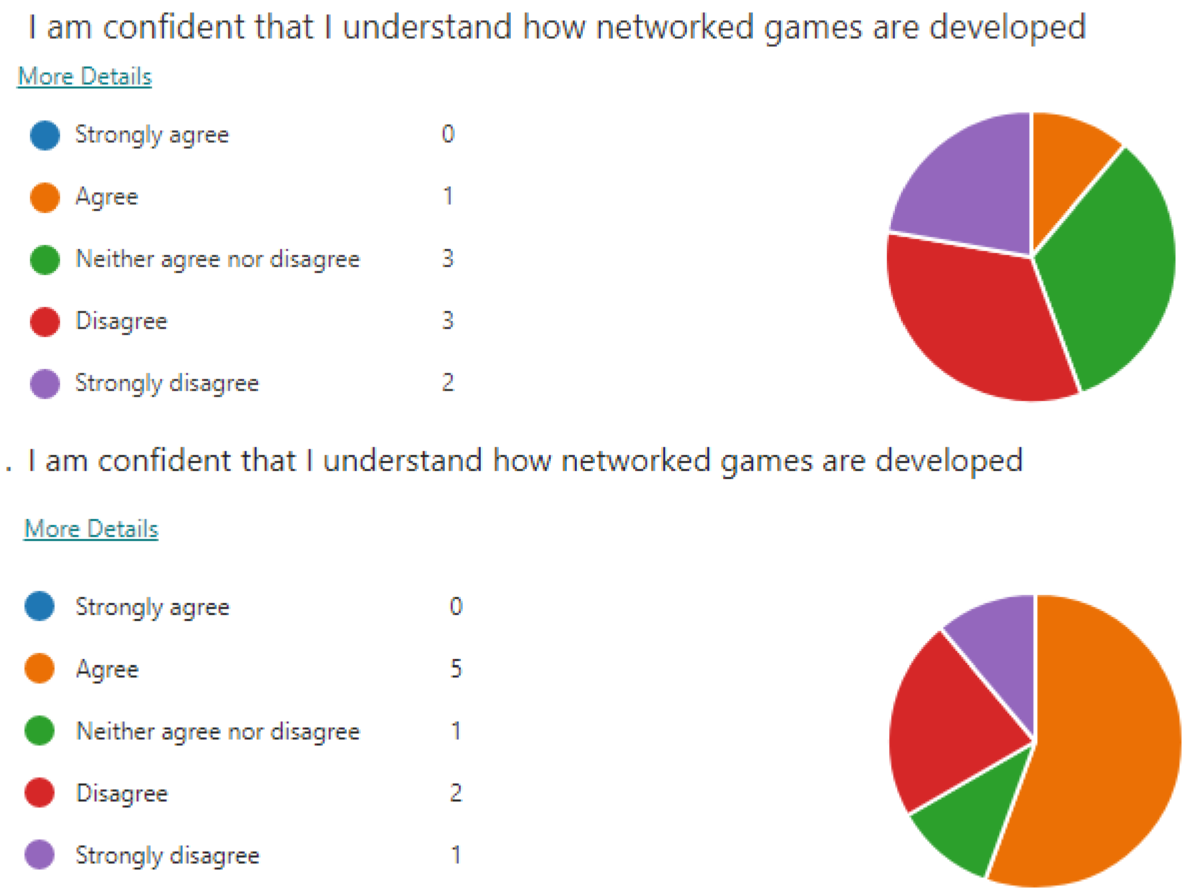
5.3 - Team Questionnaires Analysis

The questionnaire given to the team before the commencement of development of the game resulted in collection of initial insight of the knowledge of the team about development of multiplayer games and the questionnaire given to the development team after the completion of the project resulted in collection of upgraded knowledge and skills of the team about the development of multiplayer games and video games in general as well. The image before in the figures in the following sections is the image of response of the team members before the start of the project and the image after, is the image of response of the team members after the end of the project.

5.3.1 - Knowledge and Understanding of Multiplayer Games Design and Development

The first responses provided by the team members indicated that, about one-third of the team belonged to the programming discipline. Furthermore, only one member agreed with the understanding about how networked games are developed, while two strongly disagreed and three responded neutrally. This means that most of the team members were not fully confident with the development of multiplayer games.

Post completion, five members agreed on having an understanding of development of multiplayer games, only one member strongly disagreed, two disagreed and one member remained neutral while no member strongly agreed even after development (Figure 15).



I am confident that I understand how networked games are developed

More Details

● Strongly agree	0
● Agree	1
● Neither agree nor disagree	3
● Disagree	3
● Strongly disagree	2

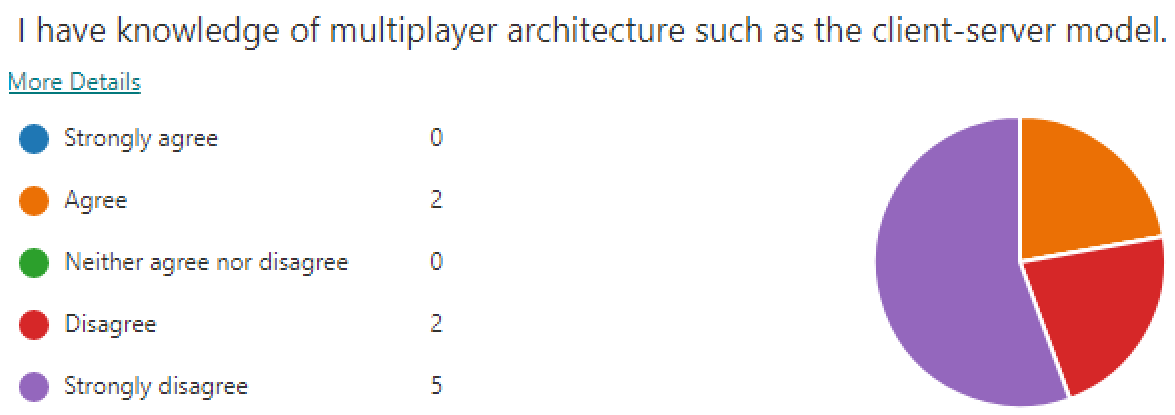
. I am confident that I understand how networked games are developed

More Details

● Strongly agree	0
● Agree	5
● Neither agree nor disagree	1
● Disagree	2
● Strongly disagree	1

Figure 15. Comparison of responses about how networked games are developed (before and after development)

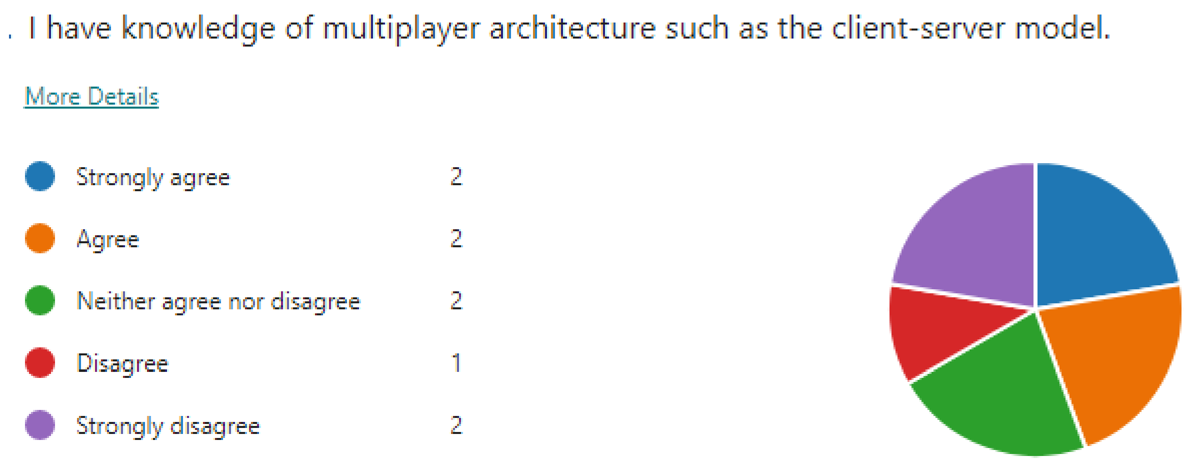
The majority of the team reported to have no knowledge of the client-server model utilised in the development of multiplayer games. Only two programmers had knowledge about this architecture. Post completion, two programmers strongly agreed with having the knowledge of the multiplayer architecture while the two members including a programmer and a designer agreed with having the knowledge of the architecture, two members stayed neutral, one disagreed and two strongly disagreed (Figure 16).



I have knowledge of multiplayer architecture such as the client-server model.

More Details

🟦 Strongly agree	0
🟧 Agree	2
🟩 Neither agree nor disagree	0
🟥 Disagree	2
🟪 Strongly disagree	5



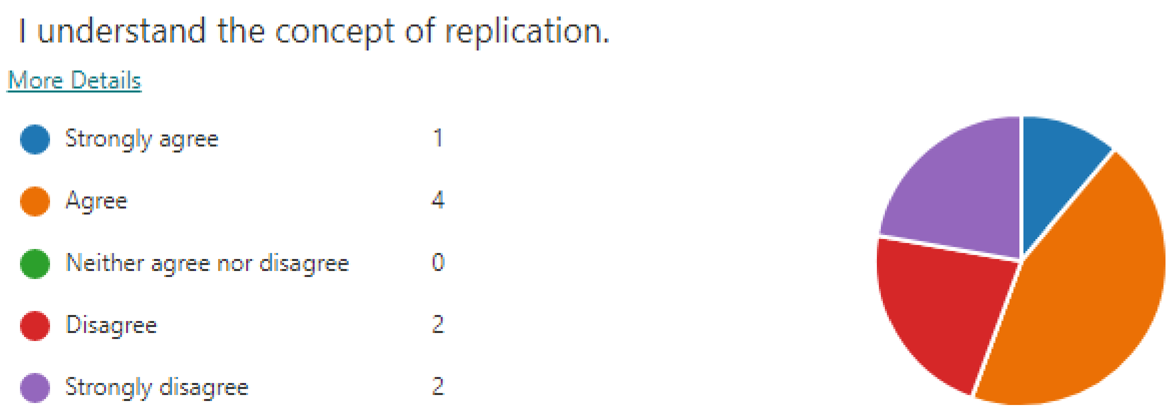
. I have knowledge of multiplayer architecture such as the client-server model.

More Details

🟦 Strongly agree	2
🟧 Agree	2
🟩 Neither agree nor disagree	2
🟥 Disagree	1
🟪 Strongly disagree	2

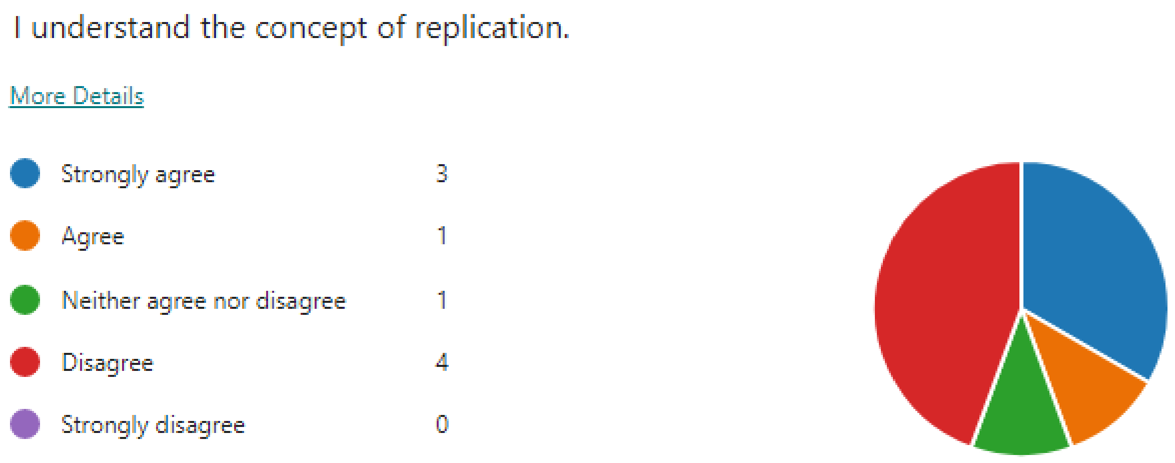
Figure 16. Comparison of knowledge of multiplayer architecture (before and after development)

The programmers, the audio designer and the animators agreed with having an understanding of the concept of replication, while the artists, the designer and the producer lacked the knowledge of replication (Figure 17). On the other hand, the majority of the team members weren't familiar with RPCs (Remote Procedural Calls). This means that initially, the team wasn't fully familiar with the implementation of multiplayer features for the project. Post completion, the three programmers fully understood the applied concept of replication while the designer understood moderately. The artists and the animator along with the audio designer only disagreed with understanding the concept of replication. The programmers and the designers agreed with understanding the concept of RPCs with two strong agreement while the rest of the team disagreed with 4 strong disagreement (Figure 18).


I understand the concept of replication.

More Details

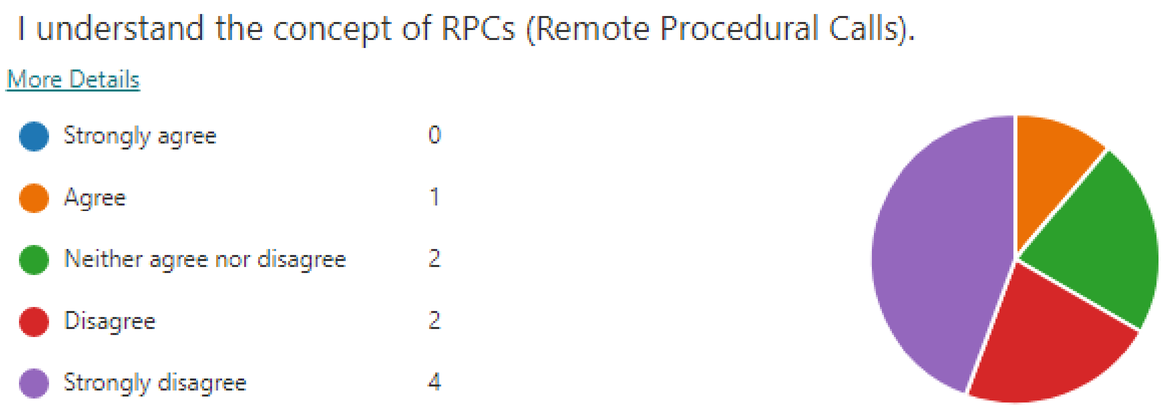
🔵	Strongly agree	1
🟠	Agree	4
🟢	Neither agree nor disagree	0
🔴	Disagree	2
🟣	Strongly disagree	2


I understand the concept of replication.

More Details

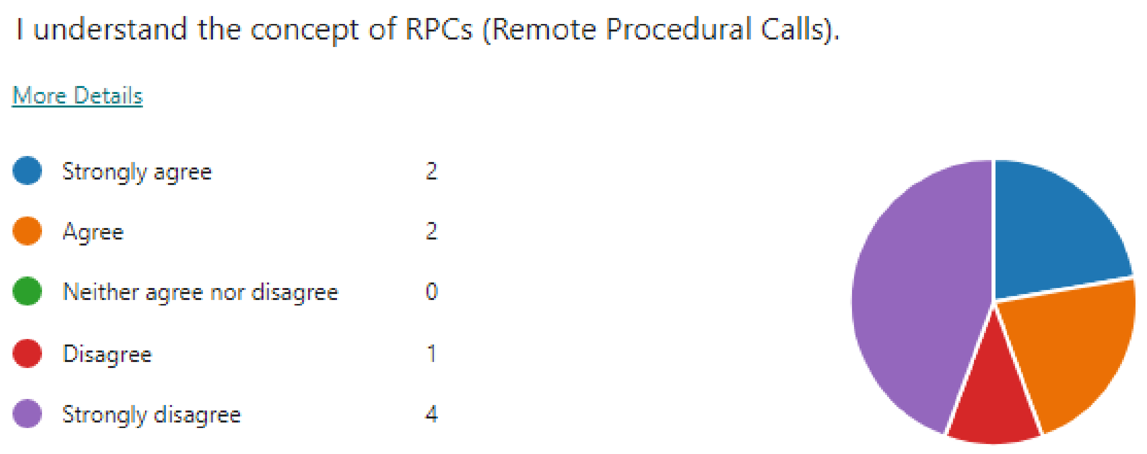
🔵	Strongly agree	3
🟠	Agree	1
🟢	Neither agree nor disagree	1
🔴	Disagree	4
🟣	Strongly disagree	0

Figure 17. Comparison of understanding the concept of replication (before and after development)


I understand the concept of RPCs (Remote Procedural Calls).

More Details

● Strongly agree	0
● Agree	1
● Neither agree nor disagree	2
● Disagree	2
● Strongly disagree	4

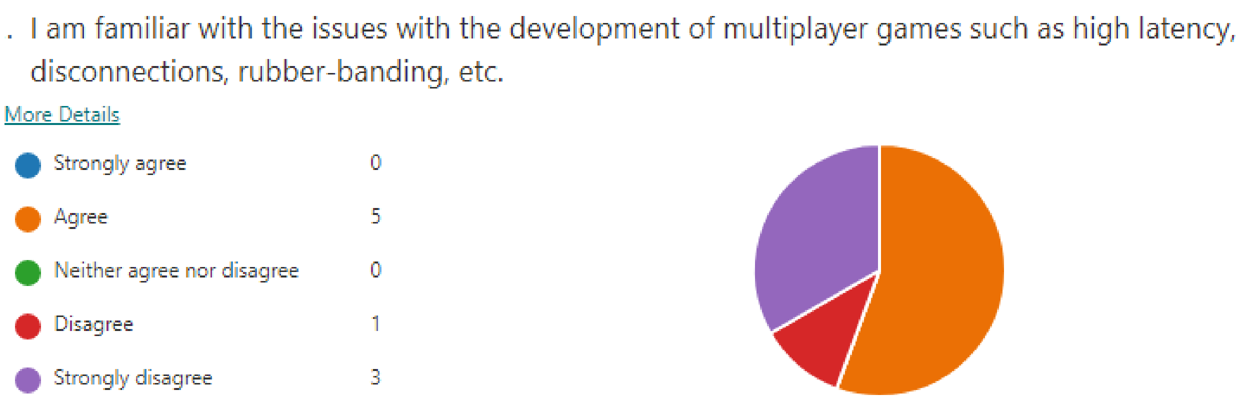

I understand the concept of RPCs (Remote Procedural Calls).

More Details

● Strongly agree	2
● Agree	2
● Neither agree nor disagree	0
● Disagree	1
● Strongly disagree	4

Figure 18. Comparison of understanding the concept of RPCs (before and after development)

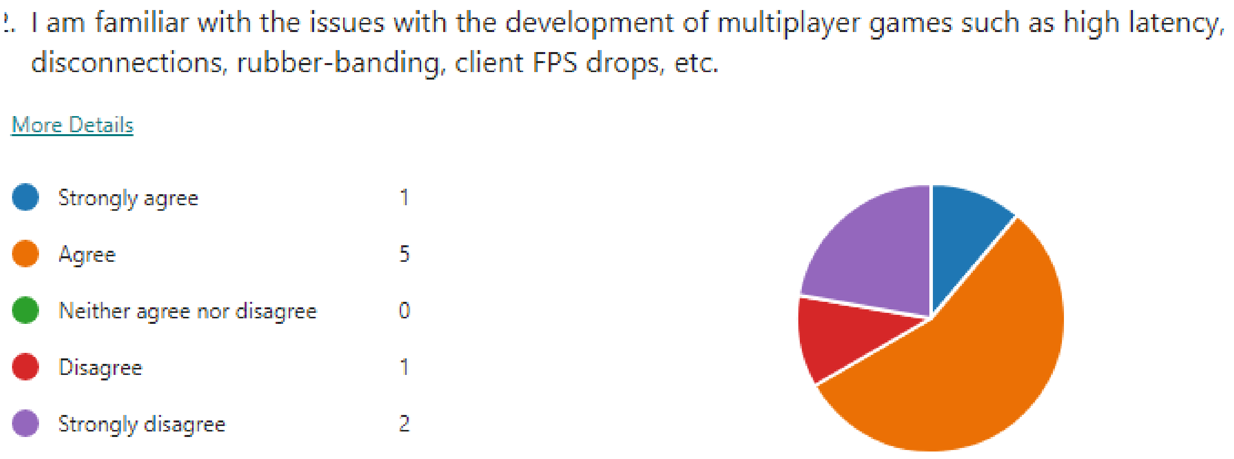
Tackling issues and bugs in multiplayer game development can be a tedious and unpredictable task. A start is usually understanding the potential technical issues that could occur during development. The programmers and the designers reported to have an understanding of the issues with the development of multiplayer games such as high latency, FPS drops, disconnections, etc. Post completion, majority of the team reported to have a good understanding of the issues in the development of multiplayer games. Only the two artists and the animator disagreed with having an understanding of the issues (Figure 19).



. I am familiar with the issues with the development of multiplayer games such as high latency, disconnections, rubber-banding, etc.

More Details

●	Strongly agree	0
●	Agree	5
●	Neither agree nor disagree	0
●	Disagree	1
●	Strongly disagree	3



!. I am familiar with the issues with the development of multiplayer games such as high latency, disconnections, rubber-banding, client FPS drops, etc.

More Details

●	Strongly agree	1
●	Agree	5
●	Neither agree nor disagree	0
●	Disagree	1
●	Strongly disagree	2

Figure 19. Comparison of familiarity with technical issues during development of multiplayer games (before and after development)

Overall, the responses in all of the categories of networking knowledge and skills have changes, especially around the programming side of the team. Meanwhile, some responses also changed for the other departments of the team as well. This means that the team was able to overcome most of the knowledge gaps regarding the networking technicalities through the development of this turn-based multiplayer game. No pie chart in particular recorded no change at all, which is quite expected because team members, especially the programmers are expected to implement and learn the specified aspects of the multiplayer programming. Additionally, it links back to research question 1 in finding about the knowledge gaps teams face in general while developing a multiplayer game.

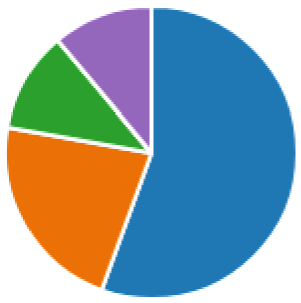
5.3.2 - Experience of Game Development

The responses provided by the team members in this section in possessing the general experience, knowledge and skills in game development before and after the development of the turn-based multiplayer game, Land of Morphie.

. I have experience working with Unreal Engine 4

More Details

●	Strongly agree	5
●	Agree	2
●	Neither agree nor disagree	1
●	Disagree	0
●	Strongly disagree	1



. I have experience working with Unreal Engine 4

More Details

●	Strongly agree	6
●	Agree	1
●	Neither agree nor disagree	2
●	Disagree	0
●	Strongly disagree	0

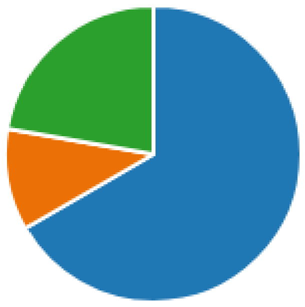
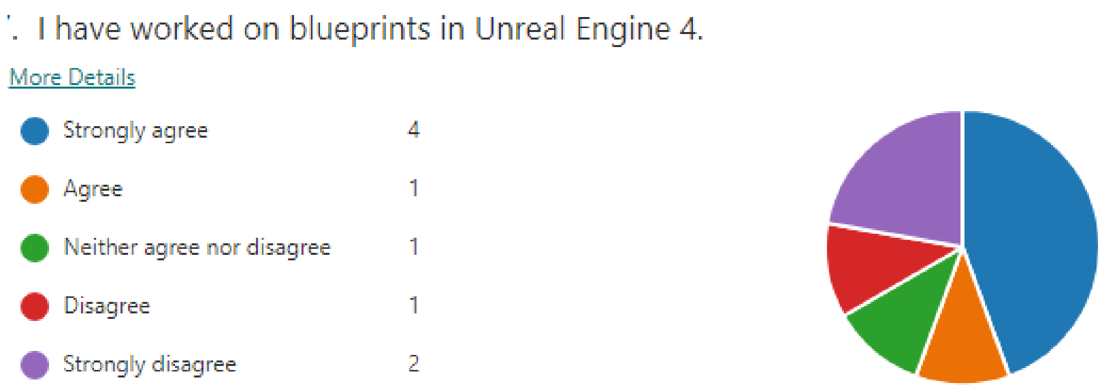


Figure 20. Comparison of the experience in working in Unreal Engine 4 (before and after development)

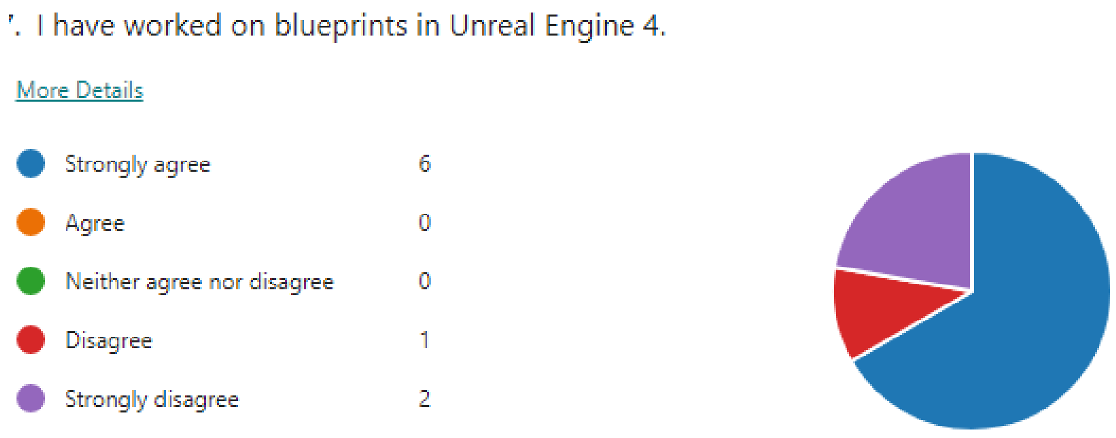
Based on the responses about experience of game development, majority of the team strongly agreed with having experience working in Unreal Engine 4. Only one member (belonging to the art department) did not have experience working in Unreal Engine 4 and only one member (also belonging to the art department) had some experience working in Unreal Engine 4. Regardless of that, the programming and design side had valuable experience in Unreal Engine 4. Post completion, majority of the team strongly agreed with having the experience of working in Unreal Engine 4, only the artists stayed neutral (Figure 20).



'. I have worked on blueprints in Unreal Engine 4.

More Details

● Strongly agree	4
● Agree	1
● Neither agree nor disagree	1
● Disagree	1
● Strongly disagree	2



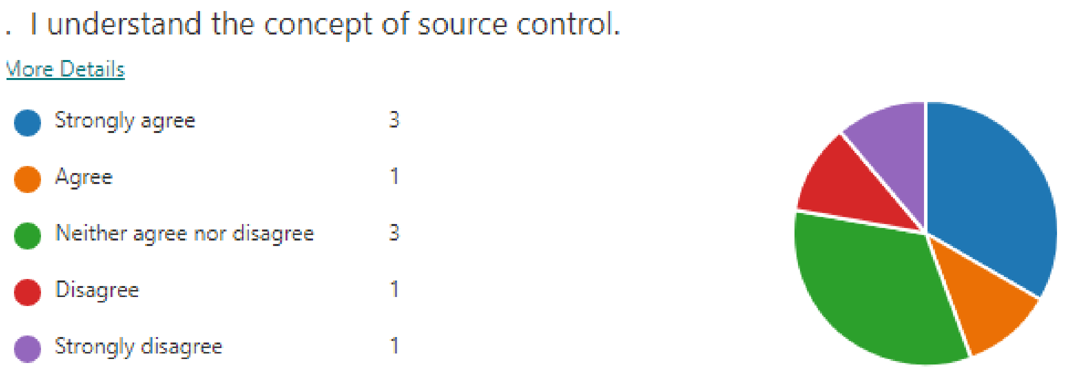
'. I have worked on blueprints in Unreal Engine 4.

More Details

● Strongly agree	6
● Agree	0
● Neither agree nor disagree	0
● Disagree	1
● Strongly disagree	2

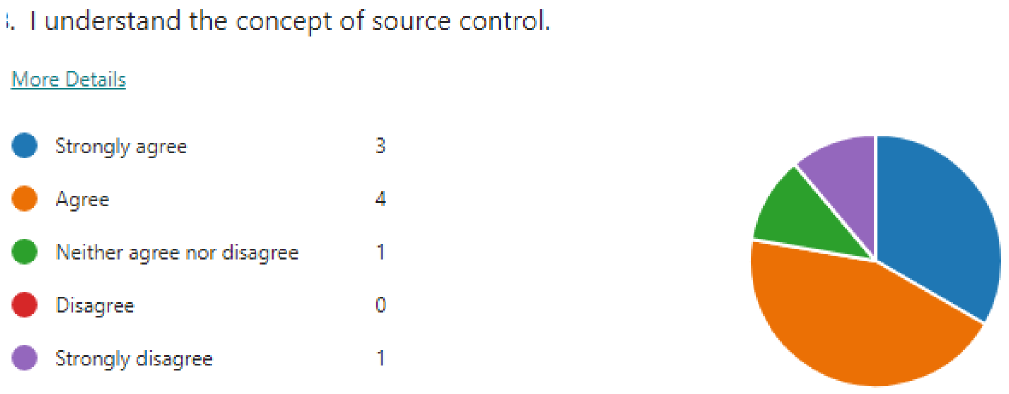
Figure 21. Comparison of the experience in working on blueprints in UE4 (before and after development)

Secondly, all the programmers and designers had valuable experience with blueprints in Unreal Engine. Meanwhile, only the artists and the animator had no experience of blueprints. After the completion of the project, most of the team strongly agreed with working in blueprints in UE4, only the animator and the artists disagreed with the experience in blueprints (Figure 21).


. I understand the concept of source control.

More Details

🔵 Strongly agree	3
🟠 Agree	1
🟢 Neither agree nor disagree	3
🔴 Disagree	1
🟣 Strongly disagree	1


ı. I understand the concept of source control.

More Details

🔵 Strongly agree	3
🟠 Agree	4
🟢 Neither agree nor disagree	1
🔴 Disagree	0
🟣 Strongly disagree	1

Figure 22. Comparison of the understanding of source control (before and after development)

Furthermore, the producer and the programmers excluding only one programmer understood the concept of source control. While three members including the 3D artist, designer and one programmer responded neutrally on the concept of source control. Two members including the 2D concept artist and animator did not understand the concept of source control. After the completion, most of the members understood the concept of source control with only one neutral response and one artist didn't understand the concept at all (Figure 22).

Lastly, no member had ever published a single player or multiplayer game on any of the games platforms such as itch.io, steam or epic games store. After the project, they were able to publish the game on games platforms such as itch.io.

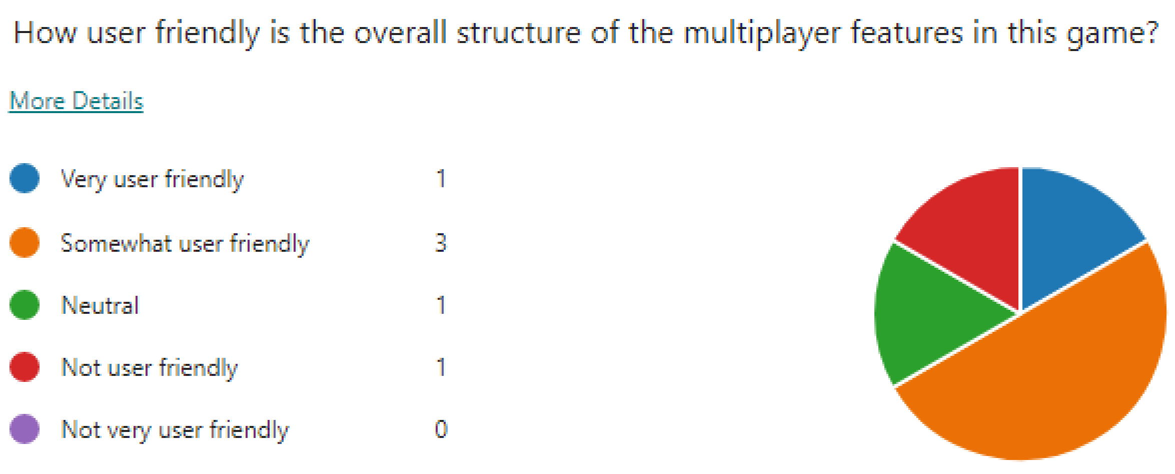
In summary, this provides useful information about the team's general knowledge of game development mostly relevant to Unreal Engine and source control. Similar to networking knowledge gap section above, all of the pie charts have recorded movement in dials meaning that the team was able to overcome the knowledge gaps in regard to general game development while using Unreal Engine. The members of the team overcame the knowledge and skill gaps through the implementation of the multiplayer gameplay mechanics with the help of tutorials, documentation by Unreal Engine and documentation of the framework. As above, it links back to research question 1 in finding about the knowledge gaps the teams face in general while developing a multiplayer game.

5.4 - Testers Questionnaire Analysis

Post-production of the turn-based multiplayer game, Land of Morphie (Appendix E) and receiving final responses from the members of the team, it was deployed and shared with the testers who played it thoroughly against each other to further provide responses of the questionnaire designed to further understand the adequateness of the multiplayer framework.

The testers questionnaire helped in better understanding the player experience in terms of the basic multiplayer features of the framework utilised for the development of the game. The framework had visible and usable features in the game itself and this helped in answering the research question 3. It is important to note that all the questions in the questionnaire linked back to the framework, which was developed as a base foundation for the game. The testers were not recruited from a specific region but from regions of Europe, the US and Asia. Since this is a turn-based game and the framework is meant to run global servers as well, the testers were anticipated not to deal with any issues of lag or regional servers only restrictions.

Most of the testers responded with a positive response for the overall structure of the multiplayer features in the game. The overall structure refers to the overall multiplayer framework around the game. Only one tester had some issues with some features of the framework, while one tester found the overall framework very user friendly (Figure 23).



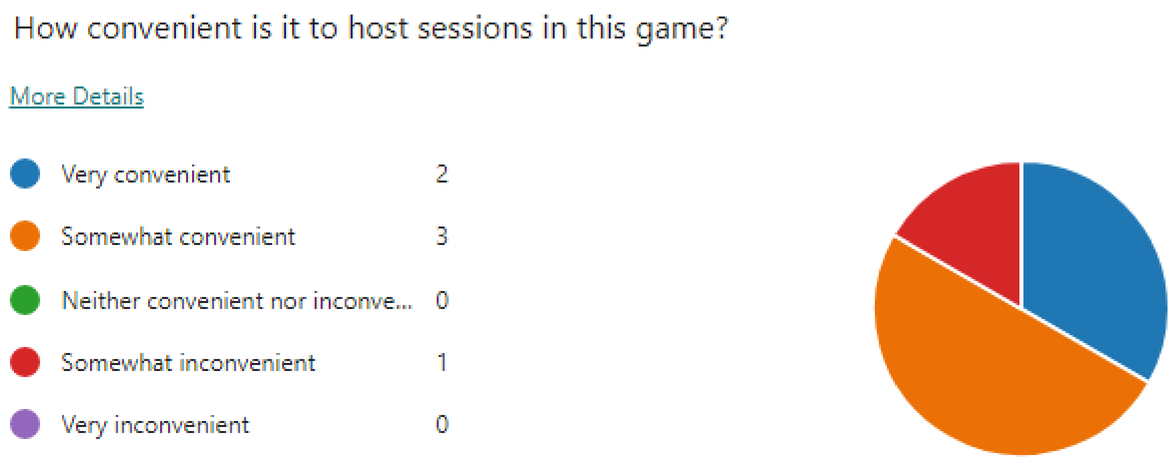
How user friendly is the overall structure of the multiplayer features in this game?

More Details

●	Very user friendly	1
●	Somewhat user friendly	3
●	Neutral	1
●	Not user friendly	1
●	Not very user friendly	0

Figure 23. Response of overall structure of the multiplayer features

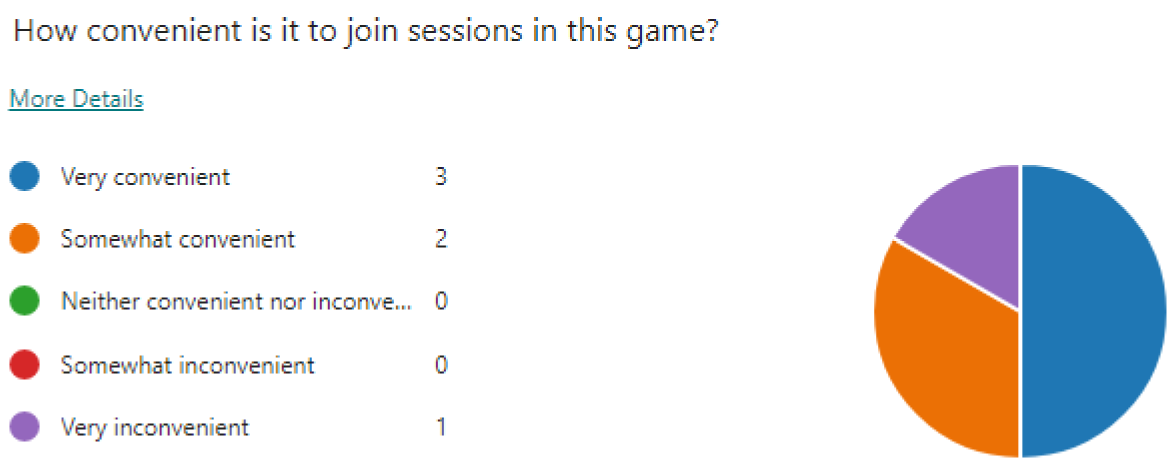
Most of the testers responded with a positive response for hosting and joining sessions in the game. Only one tester responded negatively owing to having vague idea about Steam sub-system being utilised for hosting and joining long distance servers. Two testers found it extremely easy hosting sessions in the game and three testers found it extremely easy joining session in the game (Figure 24).



How convenient is it to host sessions in this game?

More Details

●	Very convenient	2
●	Somewhat convenient	3
●	Neither convenient nor inconve...	0
●	Somewhat inconvenient	1
●	Very inconvenient	0



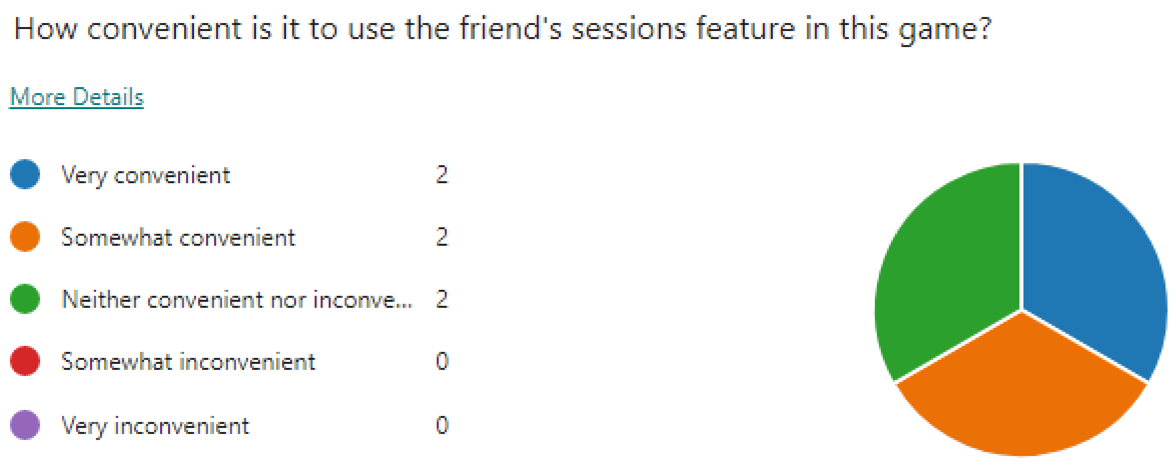
How convenient is it to join sessions in this game?

More Details

●	Very convenient	3
●	Somewhat convenient	2
●	Neither convenient nor inconve...	0
●	Somewhat inconvenient	0
●	Very inconvenient	1

Figure 24. Response of hosting and joining sessions in the game

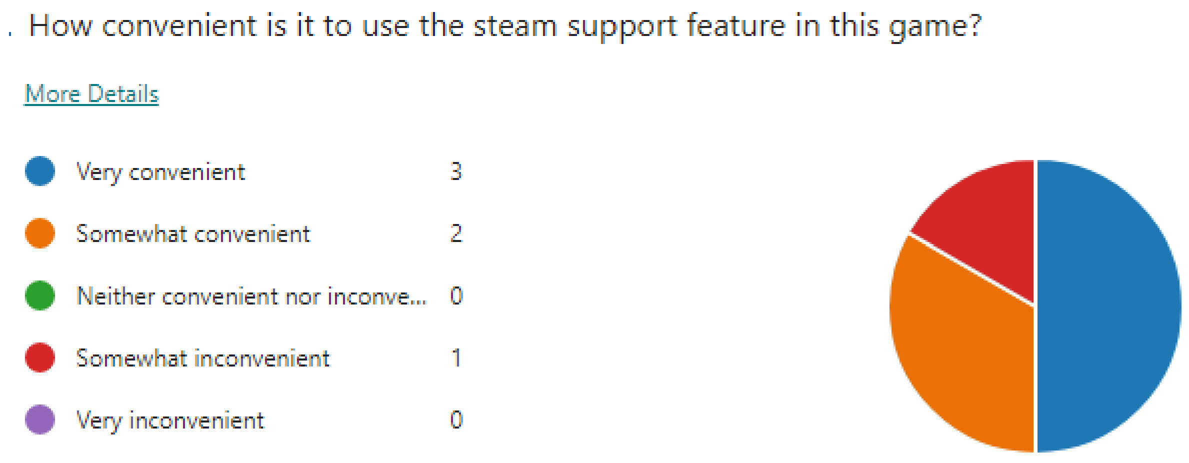
The friend’s session feature was specifically developed for the players to deal with issues of finding servers from distant regions. Since one tester particularly reported issues with hosting and finding session with normal search feature. The testers used the friend’s session with the steam support feature to easily find sessions over long distance, which they were having trouble finding using the normal search feature. Only one tester found it inconvenient using the steam support feature owing to the fact that the game had to be restarted to open steam first and then open the game to properly initialise steam for the game. Meanwhile, no testers had any issue using the friend’s sessions feature in the game (Figure 25).



How convenient is it to use the friend's sessions feature in this game?

More Details

●	Very convenient	2
●	Somewhat convenient	2
●	Neither convenient nor inconve...	2
●	Somewhat inconvenient	0
●	Very inconvenient	0



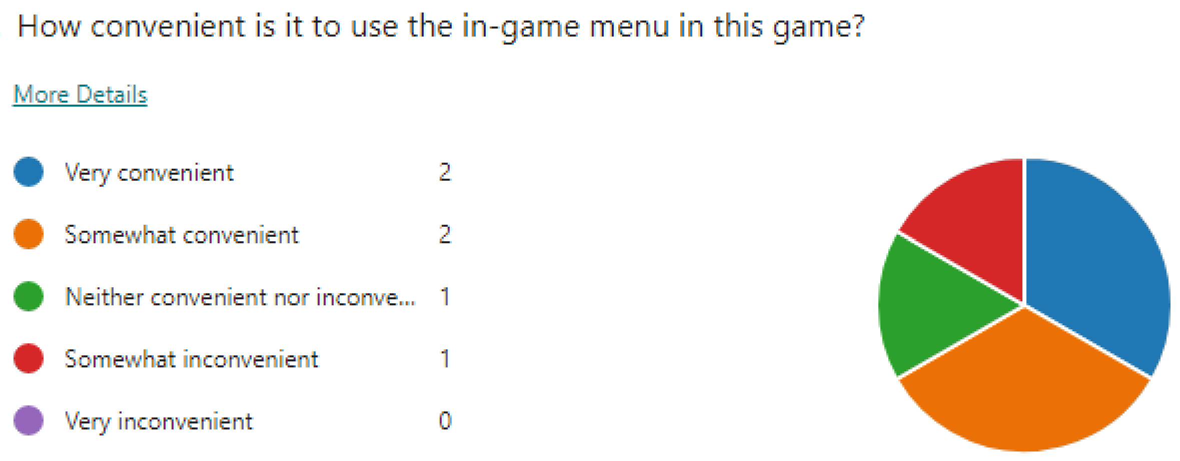
. How convenient is it to use the steam support feature in this game?

More Details

●	Very convenient	3
●	Somewhat convenient	2
●	Neither convenient nor inconve...	0
●	Somewhat inconvenient	1
●	Very inconvenient	0

Figure 25. Response of using friend’s sessions feature and steam subsystem support feature

The in-game menu was developed merely for the purpose of leaving the game to go back to the main menu or exiting the game. Most of the testers did not have any issue using the in-game menu, while one tester in particular found it uneasy to use the in-game menu and one tester was neutral in using the in-game menu (Figure 26).



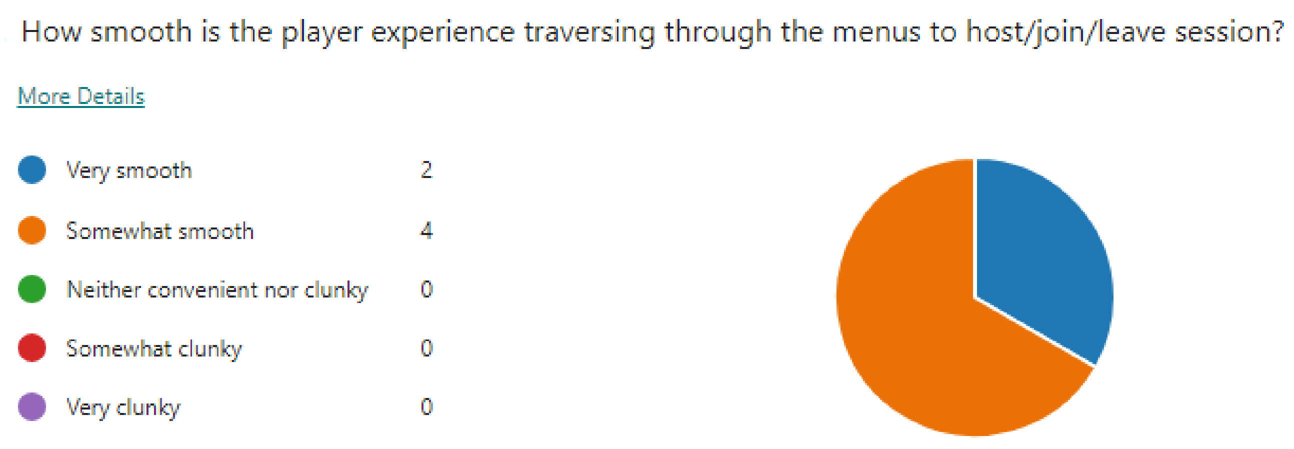
How convenient is it to use the in-game menu in this game?

More Details

●	Very convenient	2
●	Somewhat convenient	2
●	Neither convenient nor inconve...	1
●	Somewhat inconvenient	1
●	Very inconvenient	0

Figure 26. Response of using in-game menu in the game

All the testers found the player experience traversing through the menus, smooth while two testers found the experience very smooth (Figure 27).



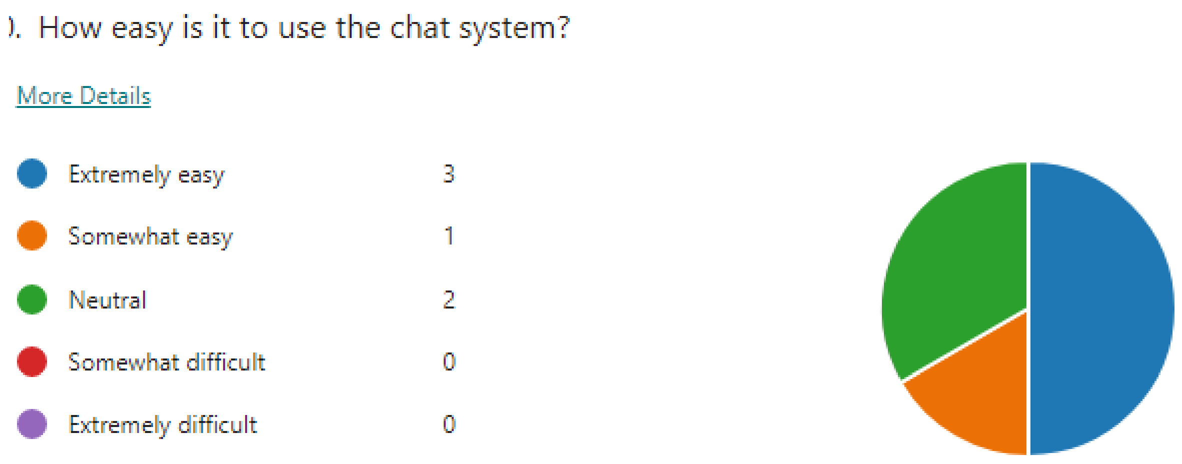
How smooth is the player experience traversing through the menus to host/join/leave session?

More Details

●	Very smooth	2
●	Somewhat smooth	4
●	Neither convenient nor clunky	0
●	Somewhat clunky	0
●	Very clunky	0

Figure 27. Response of player experience traversing through the menus

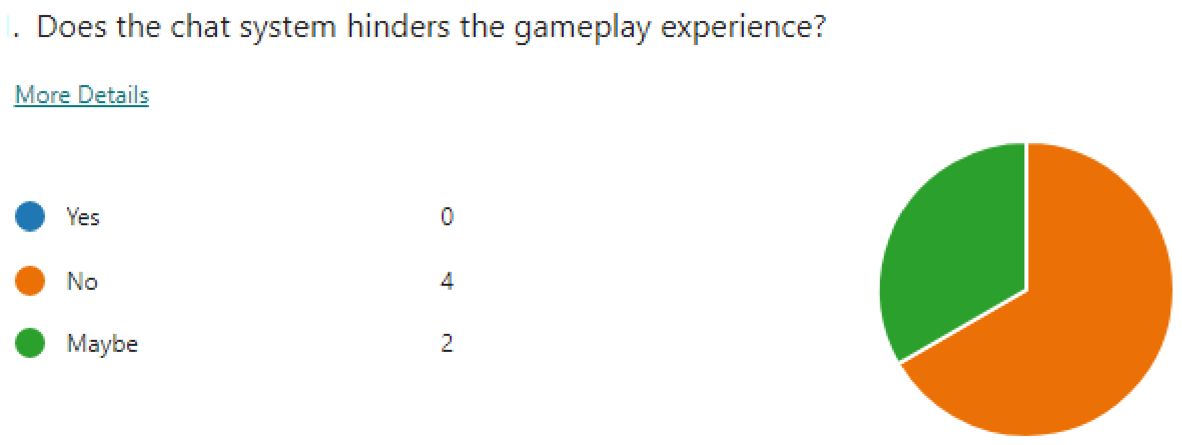
Finally, the testers responded for the last feature of the framework, i.e., the chat system. Most of the testers found it easy to use the chat while two testers were neutral about the use of the chat system. Additionally, two testers were unsure about the chat system hindering the gameplay experience due to using the feature in the game and playing it the very first time (Figure 28).



). How easy is it to use the chat system?

More Details

●	Extremely easy	3
●	Somewhat easy	1
●	Neutral	2
●	Somewhat difficult	0
●	Extremely difficult	0



. Does the chat system hinders the gameplay experience?

More Details

●	Yes	0
●	No	4
●	Maybe	2

Figure 28. Response of player experience traversing through the menus

Summarising the findings above, the testers mostly reported positive experience while using the features of the framework during the testing of the game. This provides useful information linking back to the research question 3. Additionally, this adds extra value to the research in terms of the utilisation of the framework for the development and deployment of the multiplayer game.

6. Discussion

The results or findings gathered from the observations above seek to brief the objectives of this technical research –

I. Understand the development pipelines, tools available for novice teams and how a multiplayer pipeline can be developed to support them.

II. Explore and understand the issues or challenges and knowledge gaps, the novice teams face when working on a multiplayer game and the impact of a generic multiplayer framework on the team's knowledge and development process.

III. Investigate the feedback received from testers playing the developed multiplayer game artefact, to find out the adequateness and limitations of the framework.

Each of the objectives provided area of focus to understand and investigate the development of complex multiplayer games and the technical issues faced during development. Additionally, the development of framework, it's utilisation by the team and receival by the testers, highlighted its strengths and weaknesses. This chapter will also discuss about how the framework and the overall process of deployment of the framework can be improved to develop any multiplayer game by a novice team.

Section 3.1.1 reviewed the literature about game engines, tools, and technologies with proper comparison between the best and the popular game engines in the market. Not every source or paper highlighted the multiplayer functionality of the game engines in detail. Although some papers and sources confirmed the versatile usage of Unreal Engine for multiplayer games and declared it as the winner for the development of the multiplayer games. After the full development of the multiplayer framework and the turn-based multiplayer game, one question arises, was Unreal Engine the right choice? and the answer would be, yes, but to an extent. This is because Unreal Engine has its pros and cons, but it provides a smooth development support with a contingency measure of having blueprint support for easy and fast method of implementation of features which proved to be valuable for this research project as reported in section 5.2. Development in Unreal Engine is not about implementing classes from scratch to create a fully functional game loop but customising and populating the inbuilt classes which are linked by default following the object-oriented design to implement features and create a fully functional game loop.

6.1 - Multiplayer Framework

The development of the multiplayer framework helps us understanding the multiplayer architecture of the Unreal engine and how it can be utilised to develop a generic pipeline to aid the foundation of any multiplayer game. This also highlights the strengths and weaknesses of the framework and its comparison with other frameworks or middleware out there. This section will revisit the previous sections related to framework and discuss the overall outcomes of the methods.

Section 4.1.3 reviewed the most commonly used and popular plugins or frameworks available for Unreal Engine in the market, which were, Advanced Sessions Plugin utilised to develop Cardinal Menu (Metahusk, 2016), Advanced Server Manager (Reverse Winter Studios, 2021) and Puzzle Platforms repository (Hunt, 2021). It also reviewed these frameworks in detail with their pros and cons. Section 5.1 reported the full and thorough development process of the framework in detail along with the comparisons with the framework mentioned above.

Given the time and the scale of this research, there is a limitation to which a framework can be developed. If the limitation can be exceeded, there are a lot of elements that can be improved in the framework to make it a completely generic framework, which means that the missing features from the framework can be implemented. This includes all the features that are also usually present in most of the games in general, such as the settings tabs or menu to drive the general settings of the game along with the audio system to drive the sound effects of button clicks, music volume, etc. This is similar to the features in the Cardinal Menu (Metahusk, 2016) as discussed in section 5.1. In this manner, the framework would be able to overcome the limitations that makes it worse than the Cardinal Menu (Metahusk, 2016). At the same time, the question is, is it really crucial? and the answer would be, no. Since these features would only help in easy the extra effort needed by the developers using the framework to implement them, they are not a necessity because they have nothing to do with networking and are mainly local. On the other hand, the features needed to overcome the limitations of Puzzle Platform repository (Hunt, 2021) such as the chat system and the in-game menu with *End Session* functionality are indeed necessities because they are the basics of any multiplayer game, and the *End Session* functionality completes the life cycle of a server session as described in section 5.1 (Figure 7).

The other features missing in this research framework are, a proper matchmaking system with dedicated servers. These features would make a multiplayer framework a professional framework with less or even no limitations. But again, are they really needed or crucial? the answer would be, no. This is because these features would not at all help the novice developers in any extra way owing to the fact that custom listen server hosting can fulfil the purpose of creating a server session lobby and allowing other clients to join. A proper matchmaking system is mainly needed for the games with large scale and potential in the market which are effectively the multiplayer games with eSports potential backed up by large publishers. The Advanced Server Manager (Reverse Winter Studios, 2021) has all these features present. The limitation of this framework as discussed before is that it’s costly and not a viable option for novice or student developers. These developers mostly look for free to use solutions for development, which is there was a need to have an optimal free to use solution for the team.

6.2 - Multiplayer Game Development

The game development case study along with the response to the questionnaires help us understanding the issues encountered by the team during development and their solutions. Additionally, we are able to investigate the utilisation of framework for the game and the upgradation of skills and knowledge of the novice developers to fill the gaps when they first started out developing the multiplayer game.

In hindsight, it might be easier to summarise and outline the technical issues encountered by the team developing a game. But the code preparation and proper planning or technical design is a must for any type of game or multiplayer game development. This also includes better deployment of framework for the team to use without any difficulties in laying out the basic architecture for the game. On top of that, there are several factors with novice development team on how they will carry out the development process such as usage of blueprints throughout the development. Couch Learn Tutorials (Matt, 2019) explain the creation of multiplayer sessions in Unreal Engine with Blueprints. But that wouldn’t be the most optimal method to implement and deploy the framework to help the team, which is why, usage of C++ with an option to extend to blueprints is the one of best ways to implement any framework in order to help the developers. This is similar to writing code just above the lower engine level to improve workflow for Engine users. On the other hand, Unreal Engine provides the game framework overview for both blueprints and C++ to understand the basic classes system inside the engine for multiplayer development (Unreal Engine, 2021c) as also discussed in the section above. The rules of classes for C++ and Blueprints are the same. Because of this, the team was able to utilise the documentation of blueprints and the C++ source code of the framework to find a workaround for faster development of the multiplayer game. Coming back to the point of elegant technical design for planning and easy maintenance, modification, expansion and readability of the code base, the principles remain the same for both C++ and blueprints. This would typically mean setting up of higher-level classes to run the game loop and manage the lower-level classes. All game engines commonly have the concept of *Begin Play* and *Tick* (Unreal Engine) or *Start* and *Update* (Unity) to drive the game loop. It is a common and elegant practice to have one highest level class (typically a Game Manager) to run these custom functions and drive the game loop. A typical game loop architecture with high level classes composing the code architecture has been demonstrated below (figure 29). The Game Manager initialises and ticks or updates the other managerial classes managing the responsibility assigned to them. For example, UI Manager will manage all the User Interface and the smaller classes related to it, Audio Manager will manage the Audio in the game, Turn Manager will manage the Turns and so on. But again, this obviously depends on the type of the game. Turn Manager wouldn't be a class in the code base for real-time games.

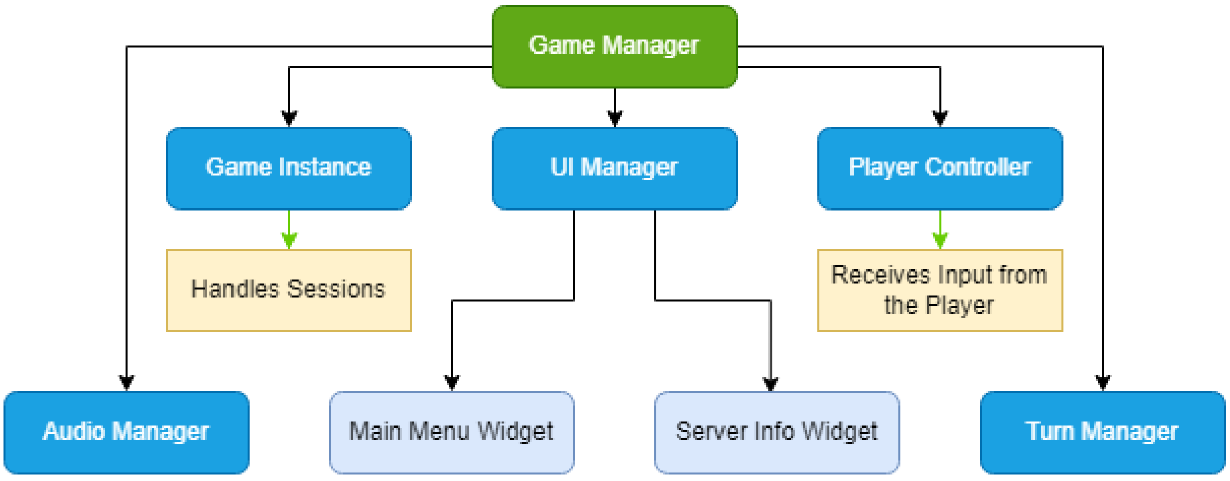


Figure 29. Typical class hierarchy to drive the game loop

In this case, Unreal Engine has already done the work of creating these classes as default, the programmers are merely supposed to create custom classes deriving from these base classes, for example, as reported in section 5.1, UI Manager was created deriving from inbuilt HUD class or *AHUD* governing the HUD of the game along with the User Interface. A Game Manager in Unreal Engine would be the same as the Game Mode or Game Mode Base. But an important lesson learnt here during the development of the framework was that this class only runs on the server and never on any client. This means for multiplayer games; the Game Mode class cannot be a Game Manager and is mainly responsible for writing rules of the games. Therefore, it was necessary to move the program to initialise the basic structure of the framework to UI Manager as it was previously implemented in the Game Mode.

On the other hand, usage for blueprints for implementing features isn't as risky as having no architecture for the game loop, because blueprints only cost performance and no support from source control for parallel programming. The switching of method of implementation from blueprints to C++ simply outweighs the risk of performance (mainly FPS, smoothness, etc.) as this is a game developed by a student team. The team had to worry about getting the basic mechanics implemented before even thinking about the performance. Performance is no doubt important but if a fully functional game isn't there, the performance itself is redundant. Additionally, an Unreal Engine developer can never avoid Blueprints, this is because all C++ classes are supposed to be extended to blueprints in order to be customised in the editor.

Overall, there are a lot of elements to be considered in terms of risk, challenges, and knowledge gaps when it comes to development of multiplayer games or even normal games within the context of student teams or novice teams. In this case, a fully functional multiplayer framework does help in providing support to a novice team by forming a foundation at a lower engine level and saving time required to implement the basic and common features. Although, it's not enough in providing support throughout the development process because it's generic and meant to be a foundational support for all multiplayer games.

6.3 - Multiplayer Game Testing

The response to the testers' questionnaire established that most of testers found it convenient and easy to use the features originating from the framework. Although one tester in particular had issues with the features of the framework due the fact that connecting to server session over longer distances must need the steam subsystem initialised and friends' servers search feature. This is a limitation to the framework, and it tells us, that the game artefact built on the framework, when opened should automatically open steam and initialise it internally, so that the player doesn't get an issue of not finding and connecting to the server sessions.

In specific terms, the testers found it easy to use most of the features of the multiplayer framework such as the hosting and joining of sessions including traversing through the menus. This means that the deployment of the game utilising the framework was relatively easier for the testers. The framework was fairly elegant to aid the testers feel comfortable with the structure of the basic multiplayer features of the game. However, the framework can be thoroughly improved in terms of smoothness when initiating lobbies in the game and have the missing feature of auto initialising of steam as mentioned above. The clients are able to join the lobbies but with a delay. This means that the join button should be disabled once the players press the join button, this will avoid any breaks or crashes in the server sessions. The framework also doesn’t familiarise the players with the rules of the game, and this isn’t possible since it’s a generic framework and is supposed to support any genre of multiplayer game. As mentioned in section 6.1, the framework should demonstrate the feature of customisation of settings such as the volume of the music or sound effects. This would ease the load on the developers using the framework and aid the testers playing the game in providing better player experience. In the end, the framework cannot directly impact the main gameplay experience of the players because the core gameplay doesn’t depend on the framework.

Overall, the framework does come up with some limitations failing to impact the game in certain areas even when it comes to players testing the games. However, the framework is fairly capable of providing elegant player experience to the players in areas such as basic multiplayer features of creating, joining, and ending the lobbies along with an in-game chat system.

7. Conclusion and Future Work

The research aimed to address the research questions (section 1.2) and investigate the development of a generic multiplayer framework and how it can support a novice student development team working towards the production of a multiplayer game along with the challenges, knowledge gaps, risks faced by the team. It also reviewed the literature on multiplayer development pipelines, workflows, and tools that exist to support a novice team. Finally, the research also investigated the limitations of the generic multiplayer framework and the feedback, the game received while it was played by the testers which in turn links back to the framework. With help of this research conducted, the challenges, knowledge gaps and risks faced by novice development teams can be described as follows –

Challenges : The challenges faced by the novice development team involve the implementation of features involving complex networking problems with game engines and their unique language syntaxes. The team face issues of implementing features correctly in one go. Moreover, in order to achieve tasks of deploying features in the game, the team usually resort to easier methods. The team may overlook the best practices followed for the development of the games.

Knowledge Gaps : The knowledge gaps are usually crucial for the programmers since developing multiplayer games require additional networking programmatic knowledge. Thus, the programmers are required to have knowledge of the following -

I. Client Server Model
II. Deep understanding of server sessions
III. Applied concept of replication
IV. Applied concept of RPCs
V. Understanding of solutions to issues such as Client FPS drops, disconnections, etc.

Risks : The risks faced by the teams are usually the success of the implementation of the projected features in the game which are cut out due to the lack of time. In order to deploy features, developers resort to methods which may not be optimal. Additionally, implementing features this way can cause level A bugs, which means breaks in the game.

Some development pipelines available for free in the market for novice teams are plugins such as the Advanced Sessions Plugin, Puzzle platform repository, etc. A generic multiplayer pipeline can be developed with Unreal Engine with inbuilt features and classes to support networking heavily emphasized by Unreal Engine.

Additionally, the conclusion derived, and the entire research process revolved around the impact of the framework on the development process. The research has concluded that the development and the foundation of a generic multiplayer framework indeed positively impacts the multiplayer development process.

The pipeline can be developed with help of some detailed documentation and guides provided by useful sources such as Unreal Engine documentation, compendium provided by Cedric Neukirchen, etc. The generic multiplayer pipeline assists the development team to tackle basic features which are compulsory in every multiplayer game and thus saves time. Although, the pipeline assists the team but doesn't directly impact the knowledge of the novice team. The team is upskilled as the project goes through the development cycle. The limitations of the generic multiplayer pipeline can be –

I. Lack of presence of dedicated servers
II. Lack of customisation in menus for a general game options menu

The end product is received positively by the players using the features of the multiplayer framework to play the game.

As explained in chapter 6, the generic multiplayer framework developed comes up with some limitations discussed in detail. On top of that, the framework does need to be elegantly deployed to aid a novice development team in fully understanding the importance of a meticulous game loop architecture which is crucial for optimal development of any kind of a game. This leads to more work in the future with research questions in mind such as –

1. How can a generic multiplayer framework be improved to fully satisfy a professional pipeline and aid any developer working towards the development of a multiplayer game.
2. How can dedicated servers be integrated within a basic multiplayer framework with proper matchmaking system, ranking system, leader boards, etc.

Finally, the framework developed for this research project can be directly downloaded from the repository (Appendix A) and used for real-world multiplayer game development inside Unreal Engine 4. It is a free to use framework and can be utilised, especially by students for any multiplayer game.

8. Glossary

Actor Class – Actor class or *AActor* in Unreal Engine is the base class for an object that can be placed or spawned in a level.

Add Movement Input – *Add Movement Input* adds movement input along with the given world direction vector (usually normalized) scaled by a scale value. This is a replicated function inbuilt in Unreal Engine.

A-star pathfinding – A-star or A* path finding algorithm is a graph traversal and path search algorithm to find the shortest path between two points.

Begin Play – An event in Unreal Engine called on an actor when play begins for the actor.

Blueprint – The blueprint visual scripting system in Unreal Engine is a complete gameplay scripting system based on the concept of using a node-based interface to create gameplay elements from within the Unreal Editor. As with many common scripting languages, it is used to define object-oriented classes or objects in the engine.

Client-Server Model – In a client-server model, one computer in the network acts as a server and hosts a session of a multiplayer game, while all of the other players' computers connect to the server as clients. The server then shares game state information with each connected client and provides a means for them to communicate with each other. In case of dedicated servers, the main server (without being a client) will fulfil the purpose of being a host.

FStruct – *FStruct* is the Unreal Engine 4 syntax for the struct data structure.

Game Instance Class – Game Instance class or *UGameInstance* is a high-level manager object for an instance of the running game in Unreal Engine.

Game Mode Class – Game Mode class or *AGameMode* is a subclass of AGameModeBase in Unreal Engine, that has extra functionality to support multiplayer matches and legacy behaviour. It contains a state machine that tracks the state of the match or the general gameplay flow.

Game Mode Base Class - Game Mode Base class or *AGameModeBase* is the class that defines the game being played in Unreal Engine. It governs the game rules, scoring, what actors are allowed to exist in this type of game, and who may enter the game. It is only instanced on the server and will never exist on the client.

HUD (Heads Up Display) – A heads-up display (HUD) is the display area where the players can see useful information related to their game such as health, abilities, armour level, ammunition, etc.

Latency/Ping – Latency is a time delay between the cause and the effect of some physical change in the system being observed. This is typically a term in multiplayer games where high ping or high latency means more delay and less ping or low latency means less delay.

MOBA – MOBA or Multiplayer online battle arena is a subgenre of strategy video games in which two teams of players compete against each other on a predefined battlefield.

Player Controller Class – The Player Controller class or *APlayerController* in Unreal Engine implements functionality for taking the input data from the player and translating that into actions, such as movement, using items, firing weapons, etc.

Player Pawn – The player pawn or the pawn is the physical representation of a player within the game world.

Replication – Replication is a term used to describe mirroring properties, actors, functions between clients and the server.

RNG (Random Number Generator) – RNG or Random number generator generates a sequence of numbers or symbols in a random fashion.

RPC (Remote Procedural Call) – RPCs are functions that are called locally but executed remotely on another machine (separate from the calling machine).

Rubber banding – A phenomenon that is caused because of an undesirable effect of the latency in which a moving actor or object appears to jerk or leap from one place to another without interpolating or traversing through the space in between the places. This is similar to unwanted teleporting.

Server Session – A server session is basically an instance of the game running on the server with a given set of properties so that it can be found and joined by players wanting to play the game.

Source Control – Source control or version control is the practice of tracking, managing and merging changes to code.

TArray – *TArray* is the Unreal Engine syntax of the array data structure.

Turn-Based – Turn-based is a type of genre of game where players take turns when playing.

9. Appendices

Appendix A – Multiplayer Framework

Link - https://github.com/animeshsharma1996/MbR-eSports-Multiplayer

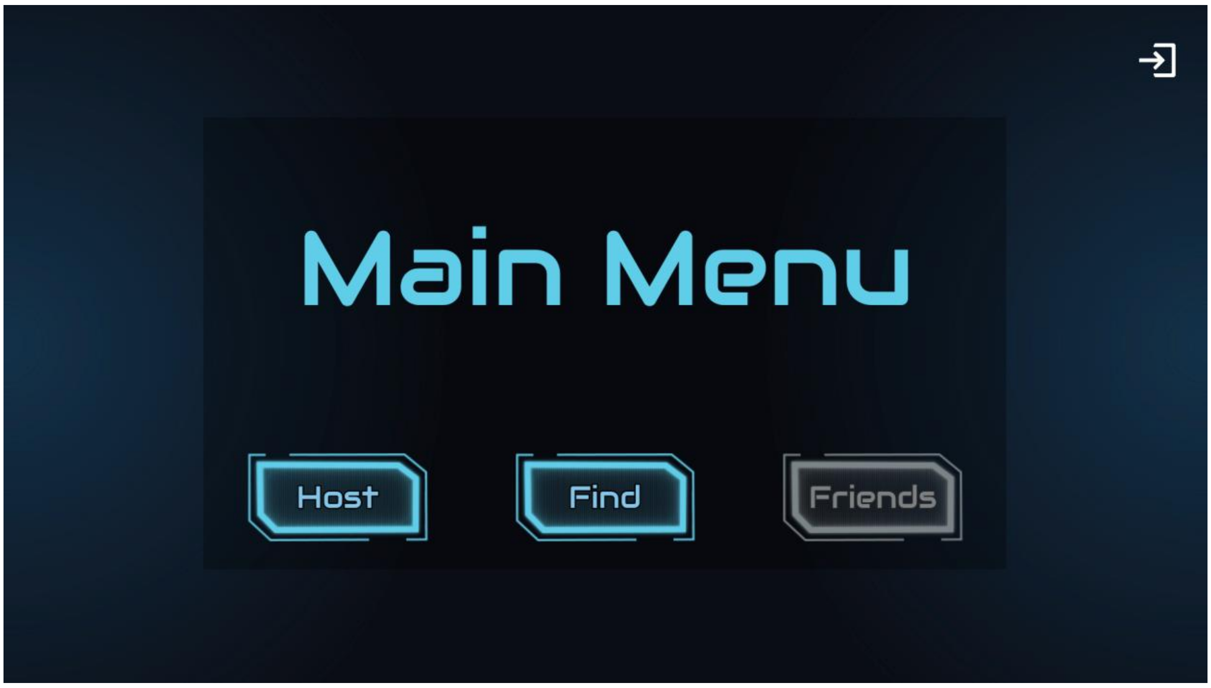
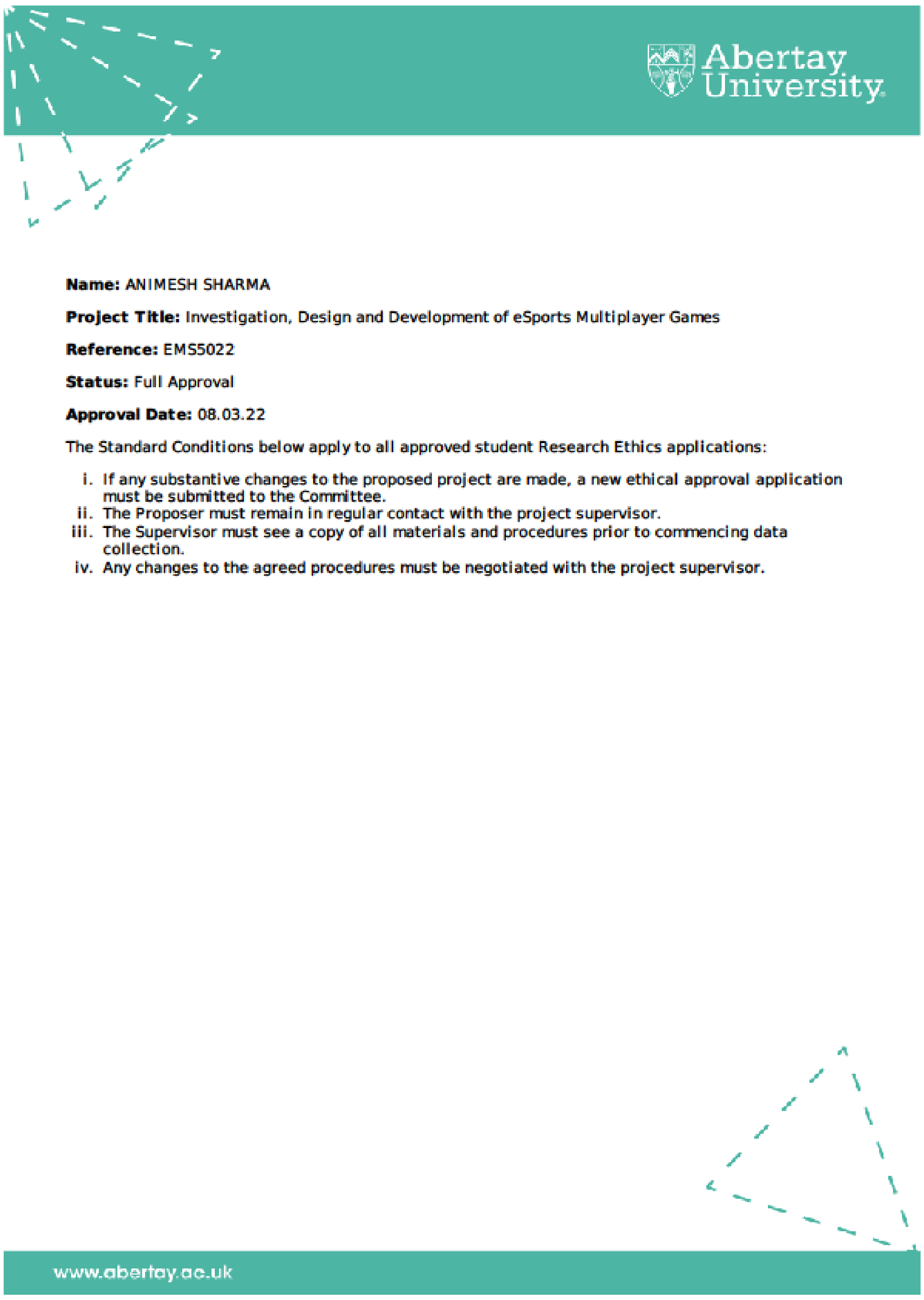


Figure 30. Framework Outline Image

Appendix B – Ethics Approval


Abertay University

Name: ANIMESH SHARMA

Project Title: Investigation, Design and Development of eSports Multiplayer Games

Reference: EMS5022

Status: Full Approval

Approval Date: 08.03.22

The Standard Conditions below apply to all approved student Research Ethics applications:

i. If any substantive changes to the proposed project are made, a new ethical approval application must be submitted to the Committee.
ii. The Proposer must remain in regular contact with the project supervisor.
iii. The Supervisor must see a copy of all materials and procedures prior to commencing data collection.
iv. Any changes to the agreed procedures must be negotiated with the project supervisor.

www.abertay.ac.uk

Appendix C – Production Timeline

The production timeline below highlights the keynotes recorded from meetings each week. The focus has been on the technical aspect of the development of the game.

Week	Notes	Data Collection
Week 1	Informed consent of all participants confirmed	Introduction to the Team – • Multidimensional team with programmers, animator, artists, designers, and producer • 3 Programmers with good programming background and networking experience • 1 animator • 1 producer and designer • 1 sound producer and designer • 1 designer and level designer • 1 concept artist • 1 3D artist
Week 2	Pitch and Project Commencement	Game Style and Mechanics Proposal – • Turn-based MOBA style game • Characters/classes with different stats • Tile based character movement
Week 3	Team introduced with framework and documentation	Roles assigned for programming team (Main focus) – • Networking • Turn based mechanics Game start and to do – • Designed hex map basic version • Mechanic of player taking damage/doing damage • Dice roll RNG • Destroy Base and game end
Week 4	Updated Project	Mechanics Implementation – • Synchronized movement on multiple clients • Grid creation for players to click on tiles to move • The horizontal movement is a priority over the vertical movement.

		To do for the programmers – • Template for a player class with HP, attack, and movement. • Mechanic of base destruction leading to game over. • Setting up teams and turn-based system
Week 5	Updated Project and More requirements for the framework	Mechanics – • The programming team switched from C++ to Blueprints for faster production, compile time and less engine crashes • Movement code shifted to Blueprints Framework requirements – • In-game menu • Steam Friend's session • Chat System
Week 6	First Playable Demo	Mechanics and Delivery – • Turn-based networking system • Team assignment to player and only one moves at a time • Instructions for playing the game
Week 7	Core Mechanics (Replicated Turn-based Movement)	Mechanics Implementation, Issues/Bugs and their fixing – • Clunky and incoherent turn-based replicated movement • Full game loop • A* pathfinding for correct tile movement • Combat System – Attack, damage intake and death • Project branches merging
Week 8	Framework Update 1	Mechanics Implementation – • UI/UX assets • Integration of playing cards UI • Timer text Framework Features Implementation – • Steam friend sessions • In-game menu interface • End server session
Week 9	Features pre-alpha	Mechanics Implementation – • 4 Classes of characters

		• Two players controlling four characters • RNG Dice Sprite • HUD • Environment map and 3D assets
Week 10	Alpha	Mechanics Implementation – • Fully functional A* path finding (with no issues) • Interactable features • Menus • Audio ○ SFX integration ○ Music integration ○ Directional sound integration
Week 11	Framework Update 2	Mechanics Implementation and bug fixing– • Timer fix • Menu assets updated • Dice roll feature merged • Spawns feature updated • Tile by tile movement on client side • Tile color highlight mechanic Framework Features Implementation – • Chat system • Toxicity prevention system in chat • Fixing of end session bug due to player controller class
Week 12	Beta	Mechanics Implementation – • Integration of the updated framework (with chat system hidden) • Integration of full audio system • Integration of all art assets (UI assets, animations) • Implementation of 3D Dice to govern the turns
Week 13	Final Framework Integration and Bug fixes	Integration – • Integration of chat system with the main project • Integrating of fully functional in-game menu with correct end server session feature • Register/Unregister players in server session Bug Fixing -

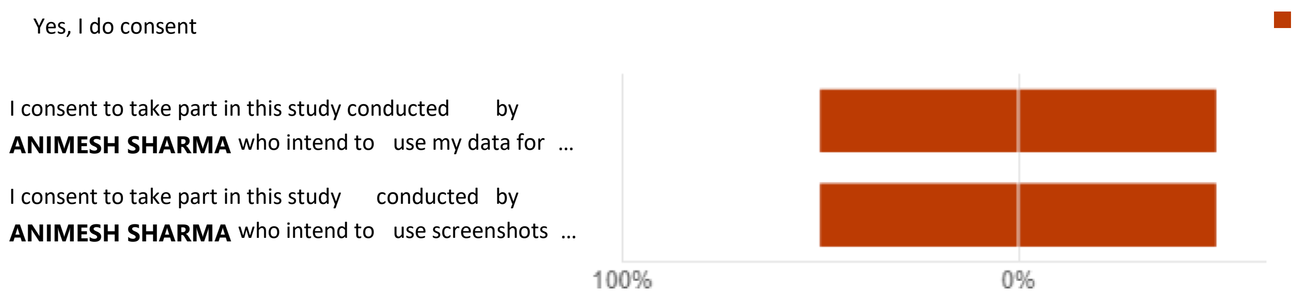
		• Fixing of bug involving client not being able to move or end turn for the game to progress further • Removal of 3D Dice from the game causing break in the game
Week 14	Gold Master	• Fully functional multiplayer game with networking features • Turn-based Game

Table 2. Timeline Table

Appendix D – Questionnaires

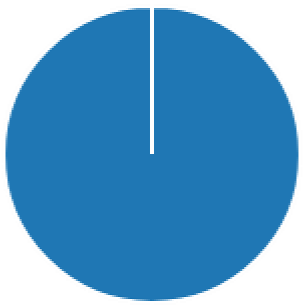
Appendix D.1 Questionnaire before development

1. PLEASE TICK BOX :



2. I have read the above statements.

●	Yes	9



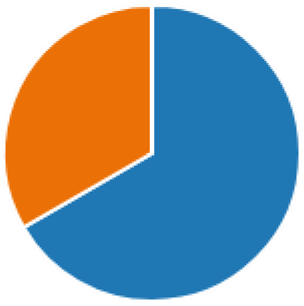
3. Please enter your participant ID number. This will have been provided to you by the researcher. Please contact the researcher if you are unsure and before continuing.

9
Responses

Latest Responses
"2"
"2003468"
"1"

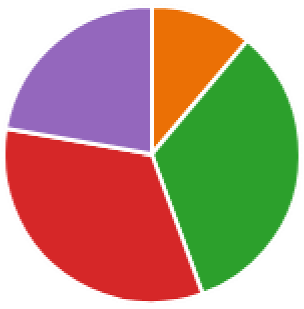
4. What type of degree programme are you studying for?

●	BA / Arts programme (e.g. GDP, ...	6
●	Bsc / Science programme (e.g. C ...	3
●	Other	0



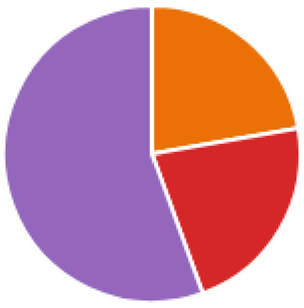
5. I am confident that I understand how networked games are developed

●	Strongly agree	0
●	Agree	1
●	Neither agree nor disagree	3
●	Disagree	3
●	Strongly disagree	2



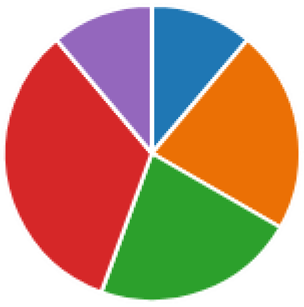
6. I have knowledge of multiplayer architecture such as the client-server model.

●	Strongly agree	0
●	Agree	2
●	Neither agree nor disagree	0
●	Disagree	2
●	Strongly disagree	5



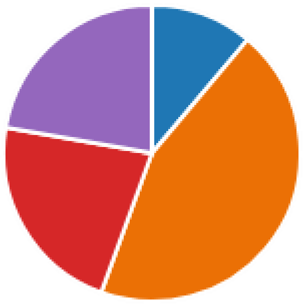
7. I understand the concept of game sessions and lobbies.

●	Strongly agree	1
●	Agree	2
●	Neither agree nor disagree	2
●	Disagree	3
●	Strongly disagree	1



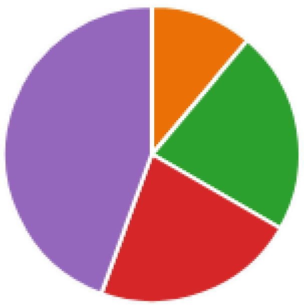
8. I understand the concept of replication.

●	Strongly agree	1
●	Agree	4
●	Neither agree nor disagree	0
●	Disagree	2
●	Strongly disagree	2



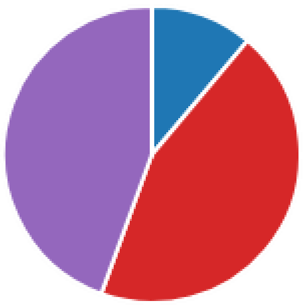
9. I understand the concept of RPCs (Remote Procedural Calls).

●	Strongly agree	0
●	Agree	1
●	Neither agree nor disagree	2
●	Disagree	2
●	Strongly disagree	4



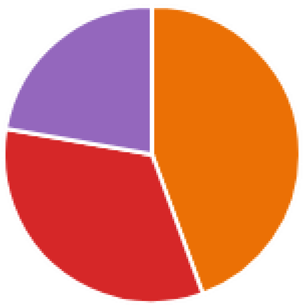
10. I am familiar with how to develop games and content for online services such as Steam OSS, Epic Games OSS, etc.

🔵	Strongly agree	1
🟠	Agree	0
🟢	Neither agree nor disagree	0
🔴	Disagree	4
🟣	Strongly disagree	4



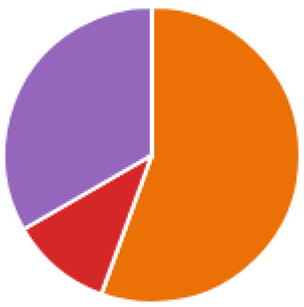
11. I understand the concept of synchronous states such as Game States, Player States, etc.

🔵	Strongly agree	0
🟠	Agree	4
🟢	Neither agree nor disagree	0
🔴	Disagree	3
🟣	Strongly disagree	2



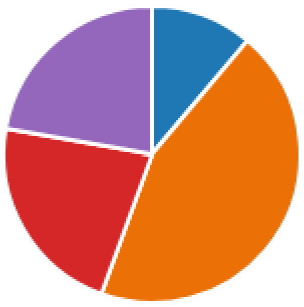
12. I am familiar with the issues with the development of multiplayer games such as high latency, disconnections, rubber-banding, etc.

🔵	Strongly agree	0
🟠	Agree	5
🟢	Neither agree nor disagree	0
🔴	Disagree	1
🟣	Strongly disagree	3



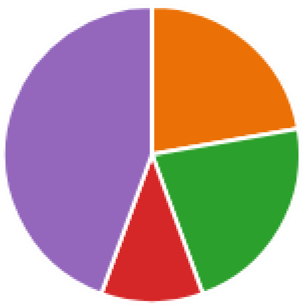
13. I am familiar with the types of servers such as dedicated servers, listen servers, etc.

🔵	Strongly agree	1
🟠	Agree	4
🟢	Neither agree nor disagree	0
🔴	Disagree	2
🟣	Strongly disagree	2



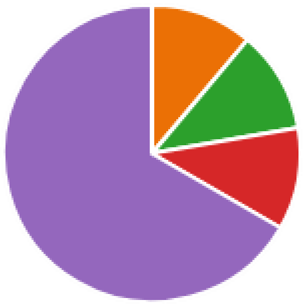
14. I am familiar with object relevancy and priority (for example, players apart in large maps are irrelevant at long distances)

🔵	Strongly agree	0
🟠	Agree	2
🟢	Neither agree nor disagree	2
🔴	Disagree	1
🟣	Strongly disagree	4



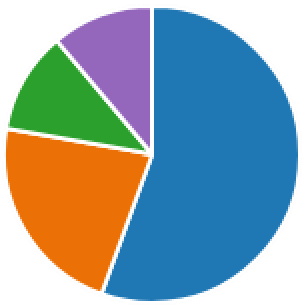
15. I am familiar with the working and implementation of multicast RPCs.

🔵	Strongly agree	0
🟠	Agree	1
🟢	Neither agree nor disagree	1
🔴	Disagree	1
🟣	Strongly disagree	6



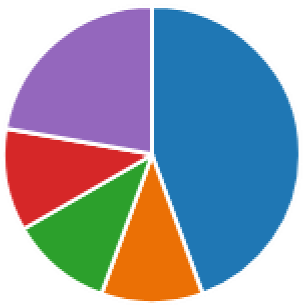
16. I have experience working with Unreal Engine 4

●	Strongly agree	5
●	Agree	2
●	Neither agree nor disagree	1
●	Disagree	0
●	Strongly disagree	1



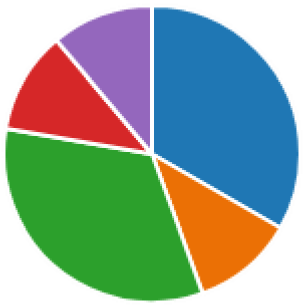
17. I have worked on blueprints in Unreal Engine 4.

●	Strongly agree	4
●	Agree	1
●	Neither agree nor disagree	1
●	Disagree	1
●	Strongly disagree	2



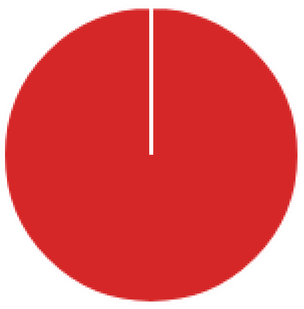
18. I understand the concept of source control.

●	Strongly agree	3
●	Agree	1
●	Neither agree nor disagree	3
●	Disagree	1
●	Strongly disagree	1



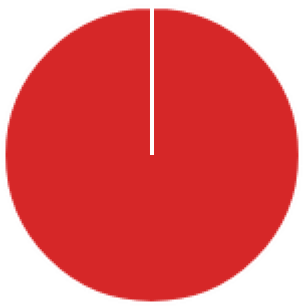
19. Have you published a single player game on a games platform? List all that apply

●	Online store such as Itch.io	0
●	Steam	0
●	Epic Games Store	0
●	Other	9



20. Have you published a multiplayer game on a games platform? List all that apply

●	Online store such as Itch.io	0
●	Steam	0
●	Epic Games Store	0
●	Other	9



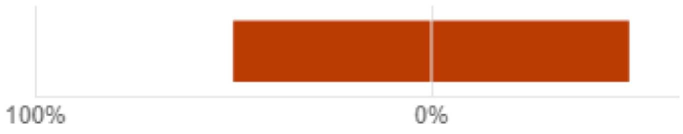
Appendix D.2 Questionnaire after development

9	04:29	Active
Responses	Average time to complete	Status

1. PLEASE TICK BOX :

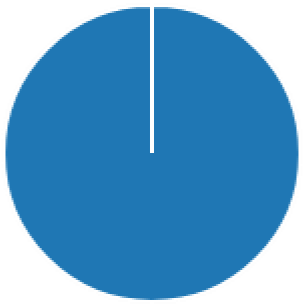
🟥 Yes, I do consent

I consent to take part in this study conducted by **ANIMESH SHARMA** who intend to use my data for ...



2. I have read the above statement.

●	Yes	9



3. Please enter your participant ID number. This will have been provided to you by the researcher. Please contact the researcher if you are unsure and before continuing.

9

Responses

Latest Responses

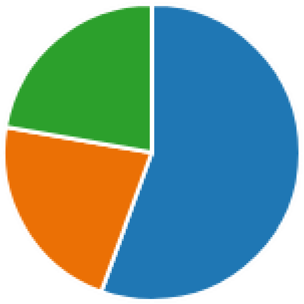
"*6*"

"*1900655*"

"*7*"

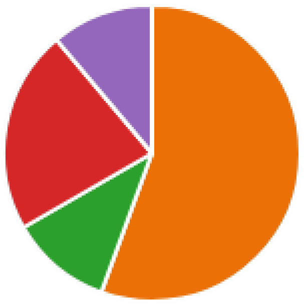
4. What type of degree programme are you studying for?

●	BA / Arts programme (e.g. GDP, ...	5
●	Bsc / Science programme (e.g. C ...	2
●	Other	2



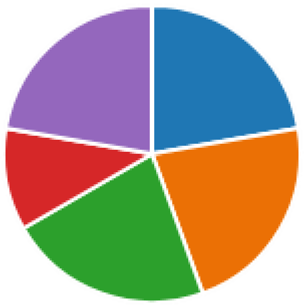
5. I am confident that I understand how networked games are developed

●	Strongly agree	0
●	Agree	5
●	Neither agree nor disagree	1
●	Disagree	2
●	Strongly disagree	1



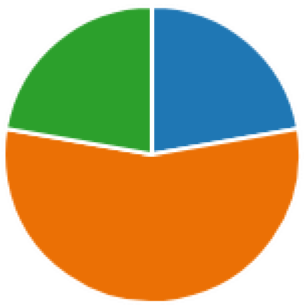
6. I have knowledge of multiplayer architecture such as the client-server model.

●	Strongly agree	2
●	Agree	2
●	Neither agree nor disagree	2
●	Disagree	1
●	Strongly disagree	2



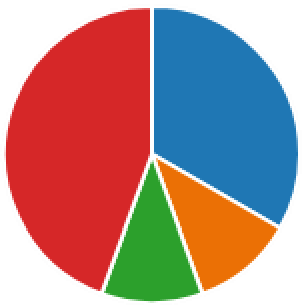
7. I understand the concept of game sessions and lobbies.

●	Strongly agree	2
●	Agree	5
●	Neither agree nor disagree	2
●	Disagree	0
●	Strongly disagree	0



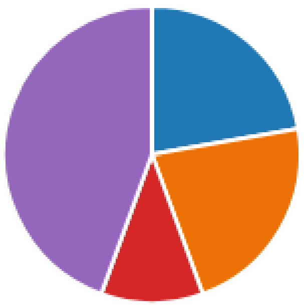
8. I understand the concept of replication.

●	Strongly agree	3
●	Agree	1
●	Neither agree nor disagree	1
●	Disagree	4
●	Strongly disagree	0



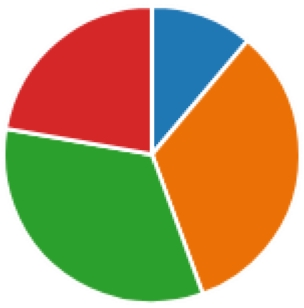
9. I understand the concept of RPCs (Remote Procedural Calls).

●	Strongly agree	2
●	Agree	2
●	Neither agree nor disagree	0
●	Disagree	1
●	Strongly disagree	4



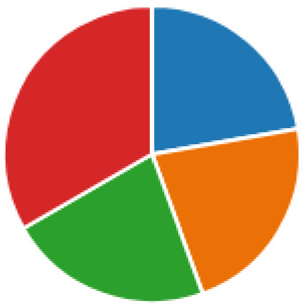
10. I am familiar with how to develop games and content for online services such as Steam OSS, Epic Games OSS, etc.

🔵	Strongly agree	1
🟠	Agree	3
🟢	Neither agree nor disagree	3
🔴	Disagree	2
🟣	Strongly disagree	0



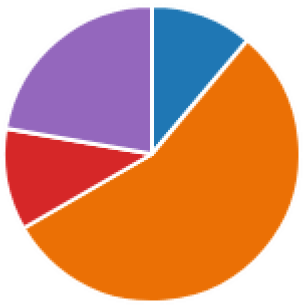
11. I understand the concept of synchronous states such as Game States, Player States, etc.

🔵	Strongly agree	2
🟠	Agree	2
🟢	Neither agree nor disagree	2
🔴	Disagree	3
🟣	Strongly disagree	0



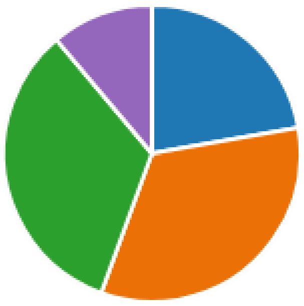
12. I am familiar with the issues with the development of multiplayer games such as high latency, disconnections, rubber-banding, client FPS drops, etc.

🔵	Strongly agree	1
🟠	Agree	5
🟢	Neither agree nor disagree	0
🔴	Disagree	1
🟣	Strongly disagree	2



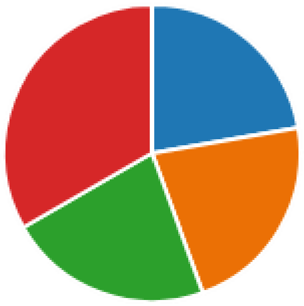
13. I am familiar with the types of servers such as dedicated servers, listen servers, etc.

●	Strongly agree	2
●	Agree	3
●	Neither agree nor disagree	3
●	Disagree	0
●	Strongly disagree	1



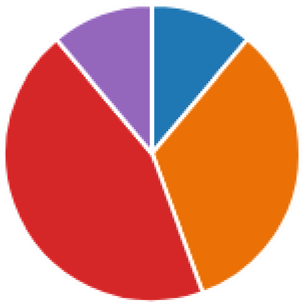
14. I am familiar with object relevancy and priority (for example, players apart in large maps are irrelevant at long distances)

●	Strongly agree	2
●	Agree	2
●	Neither agree nor disagree	2
●	Disagree	3
●	Strongly disagree	0



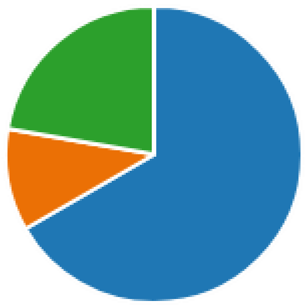
15. I am familiar with the working and implementation of multicast RPCs.

●	Strongly agree	1
●	Agree	3
●	Neither agree nor disagree	0
●	Disagree	4
●	Strongly disagree	1



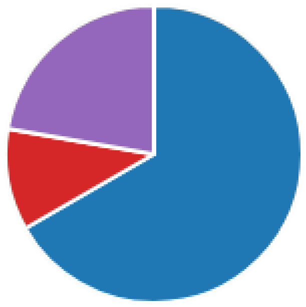
16. I have experience working with Unreal Engine 4

🔵	Strongly agree	6
🟠	Agree	1
🟢	Neither agree nor disagree	2
🔴	Disagree	0
🟣	Strongly disagree	0



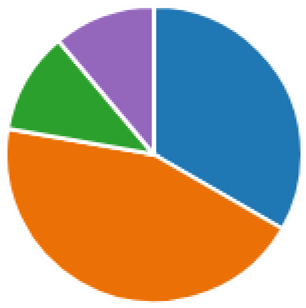
17. I have worked on blueprints in Unreal Engine 4.

🔵	Strongly agree	6
🟠	Agree	0
🟢	Neither agree nor disagree	0
🔴	Disagree	1
🟣	Strongly disagree	2



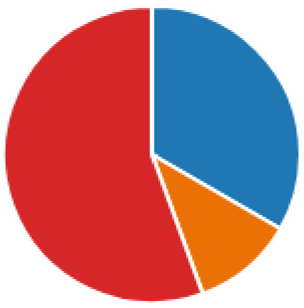
18. I understand the concept of source control.

🔵	Strongly agree	3
🟠	Agree	4
🟢	Neither agree nor disagree	1
🔴	Disagree	0
🟣	Strongly disagree	1



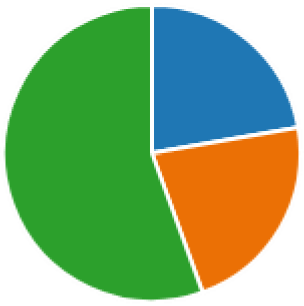
19. Have you published a multiplayer game on a games platform? List all that apply

●	Online store such as Itch.io	3
●	Steam	1
●	Epic Games Store	0
●	Other	5



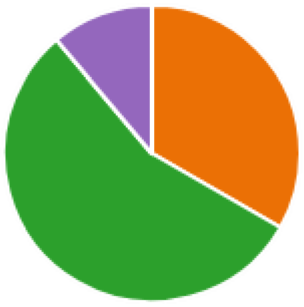
20. The framework has greatly assisted in the development of the game.

●	Strongly agree	2
●	Agree	2
●	Neither agree nor disagree	5
●	Disagree	0
●	Strongly disagree	0



21. How adequate has the framework been for the development of the game

●	Greatly adequate	0
●	Moderately adequate	4
●	Neither adequate nor inadequate	5
●	Moderately inadequate	0
●	Greatly inadequate	0



22. Is there anything missing from the framework?
Would you have any extra features included in the framework?

Latest Responses (9 Responses)

"I am a 3D artist and I have no knowledge of how the framework works."

"No opinions "

"I'm not sure."

"As 2D artist I know nothing about this topic".

"Unsure"

"Perhaps a lobby screen allowing players to ready up before starting the game".

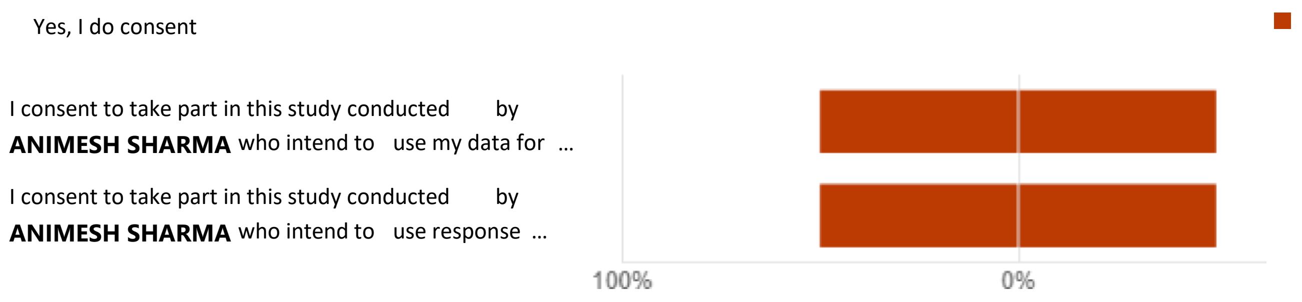
"No"

"Maybe a little extra documentation but with Animesh's hands on approach with the team we didn't need to look back on the documentation".

"No"

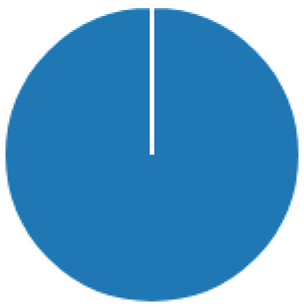
Appendix D.3 Testers Questionnaire

1. PLEASE TICK BOX :



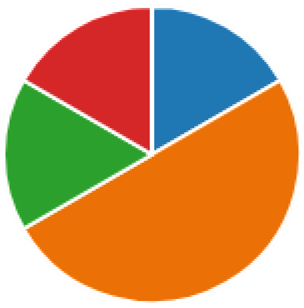
2. I have read the above statements.

●	Yes	6



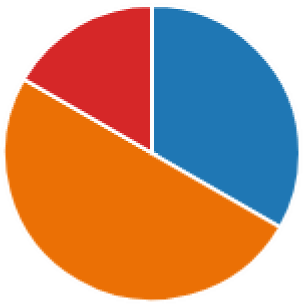
3. How user friendly is the overall structure of the multiplayer features in this game?

●	Very user friendly	1
●	Somewhat user friendly	3
●	Neutral	1
●	Not user friendly	1
●	Not very user friendly	0



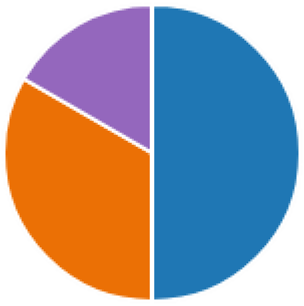
4. How convenient is it to host sessions in this game?

●	Very convenient	2
●	Somewhat convenient	3
●	Neither convenient nor inconve ...	0
●	Somewhat inconvenient	1
●	Very inconvenient	0



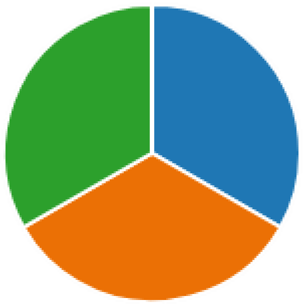
5. How convenient is it to join sessions in this game?

●	Very convenient	3
●	Somewhat convenient	2
●	Neither convenient nor inconve ...	0
●	Somewhat inconvenient	0
●	Very inconvenient	1



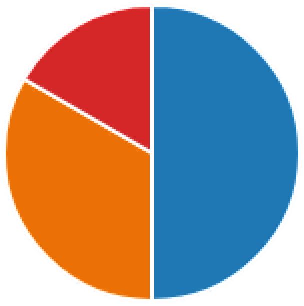
6. How convenient is it to use the friend's sessions feature in this game?

●	Very convenient	2
●	Somewhat convenient	2
●	Neither convenient nor inconve ...	2
●	Somewhat inconvenient	0
●	Very inconvenient	0



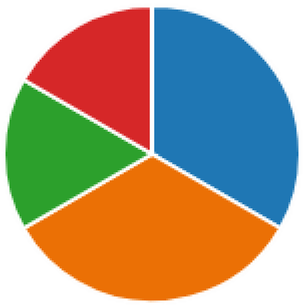
7. How convenient is it to use the steam support feature in this game?

●	Very convenient	3
●	Somewhat convenient	2
●	Neither convenient nor inconve …	0
●	Somewhat inconvenient	1
●	Very inconvenient	0



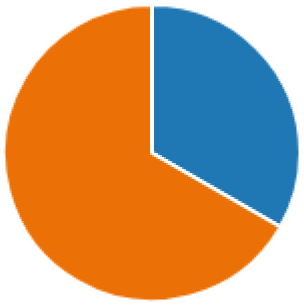
8. How convenient is it to use the in-game menu in this game?

●	Very convenient	2
●	Somewhat convenient	2
●	Neither convenient nor inconve …	1
●	Somewhat inconvenient	1
●	Very inconvenient	0



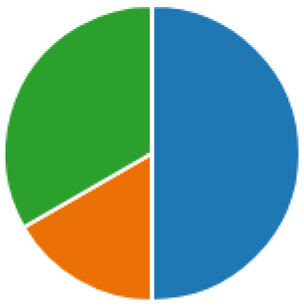
9. How smooth is the player experience traversing through the menus to host/join/leave session?

●	Very smooth	2
●	Somewhat smooth	4
●	Neither convenient nor clunky	0
●	Somewhat clunky	0
●	Very clunky	0



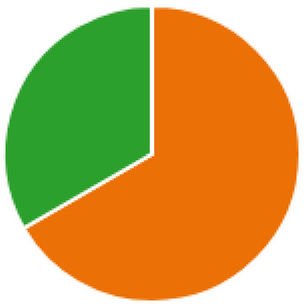
10. How easy is it to use the chat system?

●	Extremely easy	3
●	Somewhat easy	1
●	Neutral	2
●	Somewhat difficult	0
●	Extremely difficult	0



11. Does the chat system hinder the gameplay experience?

●	Yes	0
●	No	4
●	Maybe	2



12. Do you feel if there is a need to improve the multiplayer framework in this game?

Latest Responses (6 responses)

"Check below"
"Ability to generate random names and bot players."
"No "
"No, it's enough".
"Yes, the game needs a bit of a debug just to get things running smooth."
"I am not a programmer and therefore I have no knowledge of the technical aspects."

13. Do you have any feedback for the game?

Latest Responses (6 responses)

"The art style of the game could definitely be improved."
"Yes I do."
"Play dragon-age inquisition and it will help you see all the things this game lacks".

"Health bar graphics look odd to me; I would make them somewhat prettier than a red bar. Game should tell when it's your turn and the opponents turn because it gets confusing at times whose turn it is. Game joining and hosting system really buggy atm. after attacking I kept forwarding and I kept missing my last character's turn."

"There are a few bugs within the game itself: -an extra ralpie character model would sometimes appear in the water area, 2 appeared in a game I played. -attacking with a character and pressing the green skip button while the character was in an attack animation would skip the next characters turn. -clicking on tiles through characters would not work. some feedback on overall UI and mechanics: -clicking characters cards to select them would be nice. -character movement and attack range could need explaining better e.g., is the attack range the tile your character is on +1 +2?"
"The game needs a bit of a de bug because sometimes it skips your turn. Having a more clear character selection would be an advantage, as the green arrows can make you skip your turn easily if pressed more than once. Having an attack button on the screen would also be an advantage because sometimes when trying to attack you will move instead. The game is overall quite good, and I enjoyed the concept of it, also the sound effects are good/funny and add to the experience. I believe this would be a really cool game when some of the bugs are sorted out. I also like the fact that if you are too toxic in the chat, it will kick you out of the game to help people have a more positive experience. Thanks for the chance to see it! Houdini"

Appendix E – Multiplayer Game Artefact

Link - https://stormynightgames.itch.io/land-of-morphie

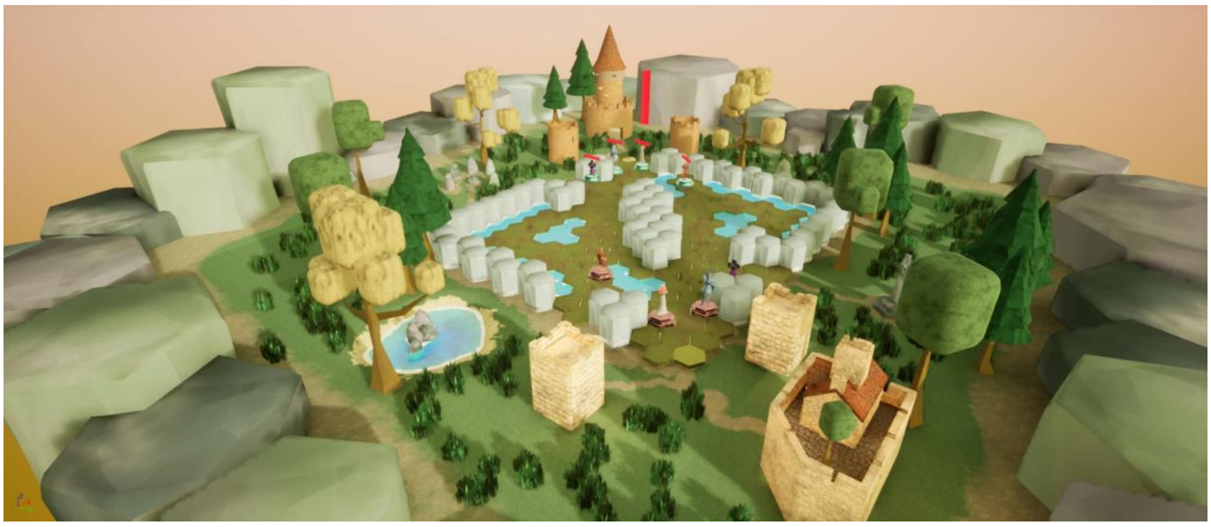


Figure 31. Online Game Artefact – Land of Morphie

10. References

Agrahari, V. and Chimalakonda, S. (2020) ‘Refactor4Green: A Game for Novice Programmers to Learn Code Smells’, in *Proceedings of the ACM/IEEE 42nd International Conference on Software Engineering: Companion Proceedings*. New York, NY, USA: Association for Computing Machinery (ICSE ’20), pp. 324–325. Available at: https://doi.org/10.1145/3377812.3390792.

Al-azawi, R., Ayesh, A. and Obaidy, M.Al. (2014) ‘Towards Agent-based Agile approach for Game Development Methodology’, in *2014 World Congress on Computer Applications and Information Systems (WCCAIS)*. *2014 World Congress on Computer Applications and Information Systems (WCCAIS)*, pp. 1–6. Available at: https://doi.org/10.1109/WCCAIS.2014.6916626.

American Psychological Association (2010) ‘Violent video games may increase aggression in some but not others, says new research’, *American Psychological Association* [Preprint]. Available at: https://www.apa.org/news/press/releases/2010/06/violent-video-games.

Apple Developer (2016) *Turn-Based Matches*, *Game Center Programming Guide*. Available at: https://developer.apple.com/library/archive/documentation/NetworkingInternet/Conceptual/GameKit_Guide/ImplementingaTurn-BasedMatch/ImplementingaTurn-BasedMatch.html#//apple_ref/doc/uid/TP40008304-CH15-SW30 (Accessed: 25 October 2021).

Arora, S. (2021) ‘Unity vs Unreal Engine: Which Game Engine Should You Choose?, Hackr.io’. Available at: https://hackr.io/blog/unity-vs-unreal-engine (Accessed: 15 December 2021).

Azmi, M. (2016) ‘Developing Game Based on Historical Event with RPG Maker MV’, in *International Conference on Learning Innovation and Quality Education*.

Bachu, E. and Bernard, M. (2012) ‘An Online Multiplayer Game for Collaborative Problem Solving’, *Semantic Scholar* [Preprint]. Available at: https://www.semanticscholar.org/paper/An-Online-Multiplayer-Game-for-Collaborative-Bachu/55685f1d463df3f58196b54e569ecddf431ef6b2 (Accessed: 7 January 2023).

Basuroy, T. (2022) *India - value of the gaming industry 2007-2022, Statista*. Available at: https://www.statista.com/statistics/235850/value-of-the-gaming-industry-in-india/ (Accessed: 15 June 2022).

Blow, J. (2004) ‘Game Development: Harder Than You Think: Ten or twenty years ago it was all fun and games. Now it’s blood, sweat, and code.’, *Queue*, 1(10), pp. 28–37. Available at: https://doi.org/10.1145/971564.971590.

Boyd, R.A. and Barbosa, S.E. (2017) ‘Reinforcement Learning for All: An Implementation Using Unreal Engine Blueprint’, in, pp. 787–792. Available at: https://doi.org/10.1109/CSCI.2017.136.

Buckley, D. (2021) ‘Unity vs. Unreal – Choosing a Game Engine’. Available at: https://gamedevacademy.org/unity-vs-unreal/ (Accessed: 28 June 2021).

Burleson, H. (2021) ‘Unity vs. Unreal Engine 2022—Which is Better?, Rigor Mortis Tortoise’. Available at: https://www.rigormortistortoise.com/unity-vs-unreal-which-is-better-2022/ (Accessed: 15 December 2021).

Bycer, J. (2019) ‘Asymmetrical Game Design, SUPERJUMP’, 25 February. Available at: https://medium.com/super-jump/asymmetrical-game-design-2d3ccbc2b4ab (Accessed: 8 September 2021).

C.D.W. (2021) *History of Esports*. Available at: https://www.cdw.com/content/cdw/en/articles/hardware/history-of-esports.html (Accessed: 7 December 2021).

Chahat (2020) ‘The Evolution of Multiplayer Mobile Games - Game App Studio’, 6 July. Available at: https://gameappstudio.com/the-evolution-of-multiplayer-mobile-games/ (Accessed: 19 July 2021).

Chen, T.T. (2015) ‘Online Games: Research Perspective and Framework’, *Computers in Entertainment* [Preprint]. Available at: https://doi.org/10.1145/2582193.2633445.

Chikhani, R. (2015) ‘The History Of Gaming: An Evolving Community, TechCrunch’. Available at: https://social.techcrunch.com/2015/10/31/the-history-of-gaming-an-evolving-community/ (Accessed: 7 July 2021).

Clement, J. (2021) *COVID-19 impact on multiplayer video games 2020, Statista*. Available at: https://www.statista.com/statistics/1188549/covid-gaming-multiplayer/ (Accessed: 15 June 2022).

Clement, J. (2022) *Online gaming - statistics & facts, Statista*. Available at: https://www.statista.com/topics/1551/online-gaming/ (Accessed: 1 June 2022).

Darby, J. (2011) *Wizards and Warriors: Massively Multiplayer Online Game Creation*. Boston, UNITED STATES: Course Technology. Available at: http://ebookcentral.proquest.com/lib/abertay/detail.action?docID=3136474.

Donlon, A. and Ziegler, J. (2020) '04: ON PEEKER'S ADVANTAGE & RANKED', *04: ON PEEKER'S ADVANTAGE & RANKED*, 13 May. Available at: https://playvalorant.com/en-us/news/game-updates/04-on-peeker-s-advantage-ranked/ (Accessed: 21 December 2021).

Eden, M. (2020) 'Top 5 Back-End Solutions For Multiplayer Games, Melior Games', *Melior Games*. Available at: https://meliorgames.com/game-development/back-end-solutions-multiplayer-games/ (Accessed: 15 September 2021).

e-Sports Earnings (2022) *Leading eSports games worldwide in 2021, by cumulative tournament prize pool (in million U.S. dollars)*. Statista, *Statista*. Statista. Available at: https://www.statista.com/statistics/501853/leading-esports-games-worldwide-total-prize-pool/ (Accessed: 22 July 2022).

Esports Insider (2022) 'Esports Around The World: India', *ESI Esports Insider*, 29 March. Available at: https://esportsinsider.com/2022/03/esports-around-the-world-india/ (Accessed: 15 June 2022).

Eugene Public Library (2021) *All Guides: Game Design and Development: Game Engines*, *Eugene Public Library*. Available at: https://eugene.libguides.com/gamedesign/engines (Accessed: 28 October 2021).

Fiedler, G. (2018) *What Every Programmer Needs To Know About Game Networking | Gaffer On Games*. Available at: https://web.archive.org/web/20180823014024/https://gafferongames.com/post/what_every_programmer_needs_to_know_about_game_networking/ (Accessed: 7 January 2023).

Forehand, R. *et al.* (2014) 'Evolution of Multiplayer'. Available at: https://web.cse.ohio-state.edu/~crawfis.3/cse5912/ReferenceMaterial/TechTeams/2014/EvolutioMultiplayer.pdf (Accessed: 30 August 2021).

Forsythe, A. (2021) *Blueprints vs. C++*, *Alex Forsythe*. Available at: http://awforsythe.com/unreal/blueprints_vs_cpp/#perf_comparison (Accessed: 7 August 2021).

Freiknecht, J. *et al.* (2016) *Game Engines*. Springer, Cham. Available at: https://link-springer-com.libproxy.abertay.ac.uk/chapter/10.1007/978-3-319-40612-1_6 (Accessed: 28 June 2022).

Game Ace (2021) *How to Make a Multiplayer Game in Unreal Engine, Game-Ace*, *Game-Ace*. Available at: https://game-ace.com/blog/multiplayer-game-in-unreal/ (Accessed: 10 September 2021).

GameMaker (2021) *GameMaker Subscriptions Products*, *GameMaker*. Available at: https://gamemaker.io/en/get (Accessed: 15 August 2021).

Gamibt (2021) *OnlineSubsystemNull | Unreal Engine Community Wiki*, *Unreal Engine Community Wiki*. Available at: https://unrealcommunity.wiki/online-multiplayer-vkje2zyn (Accessed: 29 November 2021).

Gamma, E. *et al.* (1994) *Design Patterns: Elements of Reusable Object-Oriented Software*. Addison-Wesley Professional.

Geyser, W. (2022) *The Incredible Growth of eSports [+ eSports Statistics]*, *Influencer Marketing Hub*. Available at: https://influencermarketinghub.com/esports-stats/ (Accessed: 15 June 2022).

Gough, C. (2022) *Worldwide eSports viewer numbers 2020-2025, by type*, *Statista*. Available at: https://www.statista.com/statistics/490480/global-esports-audience-size-viewer-type/ (Accessed: 31 July 2022).

Gupta, A. (2022) ‘The Rise of the Indian Gaming Market - Deep Dive’, *Product Growth*, 4 April. Available at: https://www.aakashg.com/2022/04/04/rise-of-the-indian-gaming-market/ (Accessed: 7 July 2022).

Hawkins, B. (2020) ‘Watch FIFA 20 ePL best bits as Liverpool ace loses final to Wolves star’, *talkSPORT*, 27 April. Available at: https://talksport.com/football/699487/epremier-league-invitational-liverpool-esports/ (Accessed: 8 July 2022).

Hunt, L. (2021) ‘C++ Modular Menu System’. Available at: https://github.com/lhurt51/UE4-CppMenuSystem (Accessed: 16 November 2021).

Ignatchenko, S. (2016) *Unity 5 vs UE4 vs Photon vs DIY for MMO, IT Hare on Soft.ware*, *IT Hare on Soft.ware*. Available at: http://ithare.com/unity-5-vs-ue4-vs-photon-vs-diy-for-mmo/ (Accessed: 30 August 2021).

Jansz, J. and Martens, L. (2016) ‘Gaming at a LAN event: the social context of playing video games’, *New Media & Society* [Preprint]. Available at: https://doi.org/10.1177/1461444805052280.

Jesper, J. (2007) ‘Without a Goal: on open and expressive games’, in *Videogame, Player, Text*. Manchester University Press, pp. 191–203. Available at: https://www.jesperjuul.net/text/withoutagoal/ (Accessed: 26 August 2021).

Jetsonen, T. (2016) *Development of online game prototype with Unity engine*. JAMK University of Applied Sciences. Available at: https://www.theseus.fi/bitstream/handle/10024/108970/Jetsonen_Timo.pdf;jsessionid=D18B711D79BDDC0274181B7A8EABF826?sequence=1 (Accessed: 9 September 2021).

Jones, P.K. (2022) 'Diogo Jota laughs as Man United are knocked out of the FA Cup whilst he plays FIFA on Twitch', *The Empire of The Kop*, 5 February. Available at: https://www.empireofthekop.com/2022/02/05/video-diogo-jota-laughs-as-man-united-are-knocked-out-of-the-fa-cup-whilst-he-plays-fifa-on-twitch/ (Accessed: 4 June 2022).

Kagedesu, P. (2018) 'Alpha NET [MV] – KageDesu Workshop', 6 March. Available at: https://kdworkshop.net/plugins/alpha-net/ (Accessed: 27 June 2022).

Karhulahti, V.-M. and Grabarczyk, P. (2021) 'Split-Screen: Videogame History Through Local Multiplayer Design', *Design Issues*, pp. 32–44. Available at: https://doi.org/10.1162/desi_a_00634.

Kelly, T. (2011) *Opinion: Synchronous or Asynchronous Gameplay*, *Game Developer*. Available at: https://www.gamedeveloper.com/design/opinion-synchronous-or-asynchronous-gameplay (Accessed: 8 August 2021).

Khaleel, F.L. *et al.* (2017) 'Gamification-based learning framework for a programming course', in *2017 6th International Conference on Electrical Engineering and Informatics (ICEEI)*. *2017 6th International Conference on Electrical Engineering and Informatics (ICEEI)*, pp. 1–6. Available at: https://doi.org/10.1109/ICEEI.2017.8312377.

Khromov, N. *et al.* (2018) 'Esports Athletes and Players: A Comparative Study', *IEEE Pervasive Computing*, 18(3), pp. 31–39. Available at: https://doi.org/10.1109/MPRV.2019.2926247.

Kritskiy, D. *et al.* (2022) 'Development of a Collaborative Platform for Education in Virtual Reality', in *Integrated Computer Technologies in Mechanical Engineering - 2021*. Springer, Cham. Available at: https://doi.org/10.1007/978-3-030-94259-5_25.

Lawless, J. (2020) *Sergio Aguero's Reaction To Missing A Chance With Himself On FIFA Is Priceless*. Available at: https://www.sportbible.com/football/news-sergio-agueros-reaction-to-missing-a-chance-with-himself-on-fifa-20200426 (Accessed: 14 August 2021).

Lee, S.W.K. and Chang, R.K.C. (2018) 'Enhancing the experience of multiplayer shooter games via advanced lag compensation', in *Proceedings of the 9th ACM*

Multimedia Systems Conference. New York, NY, USA: Association for Computing Machinery (MMSys ’18). Available at: https://doi.org/10.1145/3204949.3204971.

Lemarchand, R. (2021) *A Playful Production Process: For Game Designers (and Everyone)*. Cambridge, MA, USA: MIT Press. Available at: https://mitpress.mit.edu/books/playful-production-process.

Levchenko, B., Chukhray, A. and Chumachenko, D. (2020) ‘Development of Game Modules with Support for Synchronous Multiplayer Based on Unreal Engine 4 Using Artificial Intelligence Approach’, in *Integrated Computer Technologies in Mechanical Engineering*. Springer, Cham. Available at: https://doi.org/10.1007/978-3-030-37618-5_43.

Liniestsky, J., Manzur, A. and Godot community (2019) ‘Godot Engine Documentation Release 2.1’. Available at: https://www.academia.edu/42256850/Godot_Engine_Documentation_Release_latest (Accessed: 12 July 2022).

Madhav, S. and Glazer, J. (2015) *Multiplayer Game Programming: Architecting Networked Games*. 1st edn. Addison-Wesley Professional. Available at: https://www.informit.com/store/multiplayer-game-programming-architecting-networked-9780134034300?w_ptgrevartcl=Overview+of+Networked+Games_2461064 (Accessed: 29 June 2022).

Marín-Vega, H. *et al.* (2016) ‘Analyzing Proprietary Games Engines for Developing Educational and Serious Games’, *Research in Computing Science* [Preprint]. Available at: https://doi.org/10.13053/rcs-129-1-3.

Markov, R. (2022) *Most watched esports disciplines in 2021*, *Esports Charts*. Available at: https://escharts.com/news/most-watched-esports-disciplines-2021 (Accessed: 28 May 2022).

Matt (2019) ‘Multiplayer Sessions in Unreal Engine 4’, *Couch Learn*, 31 July. Available at: https://couchlearn.com/multiplayer-sessions-in-your-unreal-engine-4-game/ (Accessed: 18 September 2021).

McEachen, J.C. (2004) ‘A Self-Similarity Traffic Analysis of an Internet-Based Multiplayer Online Game’, in *Proceedings of 3rd ACM SIGCOMM Workshop on Network and System Support for Games*. New York, NY, USA: Association for Computing Machinery (NetGames ’04), p. 170. Available at: https://doi.org/10.1145/1016540.1016565.

Media CBL (2021) 'Menu Widgets & C++', *CBL Media*. Available at: https://mediacbl.com/menu-widget-unreal-engine/ (Accessed: 7 December 2021).

Metahusk (2016) *Community Project | Cardinal Menu System Instructions, Help, and Discussion | Metahusk's Community Discussion*, *Metahusk*. Available at: https://community.metahusk.com/topic/26/community-project-cardinal-menu-system-instructions-help-and-discussion (Accessed: 16 September 2021).

Mirowski, A. and Harper, B.P. (2019) 'Elements of Infrastructure Demand in Multiplayer Video Games', *Media and Communication*, 7(4), pp. 237–246. Available at: https://doi.org/10.17645/mac.v7i4.2337.

Morgan, G. (2009) 'Challenges of Online Game Development: A Review', *Simulation & Gaming*, 40(5). Available at: https://doi.org/10.1177/1046878109340295.

Murray, S. (2021) *Unity Will Support Nvidia DLSS Natively By The End Of 2021*, *TheGamer*. Available at: https://www.thegamer.com/unity-dlss-support-nvidia-2021/ (Accessed: 28 August 2021).

Neukirchen, C. (2021) 'UE4 Multiplayer Sessions in C++', *An Unreal Engine Blog*, 27 June. Available at: https://cedric-neukirchen.net/2021/06/27/ue4-multiplayer-sessions-in-c/ (Accessed: 18 November 2021).

Newzoo (2022) *eSports market revenue worldwide from 2019 to 2025 (in million U.S. dollars)*, *Statista*. Available at: https://www.statista.com/statistics/490522/global-esports-market-revenue/ (Accessed: 29 July 2022).

Olsson, K. (2015) 'Unreal engine and Online Multiplayer with the OnlineSubSystem', *Madebykrol*, 21 May. Available at: https://madebykrol.wordpress.com/2015/05/21/unreal-engine-and-online-multiplayer/ (Accessed: 11 November 2021).

Opera Team (2021) *Meet GXC, the new gaming platform that lets you play and compete for weekly prize challenges directly in Opera GX*, *Opera Desktop*. Available at: https://blogs.opera.com/desktop/2021/11/gxc-open-beta/ (Accessed: 19 December 2021).

Pearce, C. and Artemesia (2009) *Communities of play: emergent cultures in multiplayer games and virtual worlds*. Cambridge, Mass: MIT Press.

Photon (2021a) *Multiplayer Game Development Made Easy | Photon Engine*. Available at: https://www.photonengine.com/en-US/Photon (Accessed: 28 August 2021).

Photon (2021b) *Photon Realtime Pricing Plans | Photon Engine*. Available at: https://www.photonengine.com/en-US/Realtime/Pricing (Accessed: 28 August 2021).

PocketGamerbiz (2021) *eSports market revenue worldwide in 2021, by region (in million U.S. dollars)*, *Statista*. Available at: https://www.statista.com/statistics/443147/estimate-of-global-market-revenue-of-esports-by-region/ (Accessed: 25 July 2021).

PwC (2021) *Global eSports market revenue 2025*, *Statista*. Available at: https://www.statista.com/statistics/1129596/esports-revenue/ (Accessed: 28 May 2022).

Rama, A.M., Fernandez, V.R. and Camacho, D. (2020) 'Finding Behavioural Patterns Among League of Legends Players Through Hidden Markov Models', in. Springer. Available at: https://doi.org/10.1007/978-3-030-43722-0_27.

Research and Markets (2021) 'Global Esports Market Report 2021 Featuring Activision Blizzard, Modern Times Group, Tencent, Valve, Electronic Arts', *Cision PR Newswire*, 12 October. Available at: https://www.prnewswire.com/news-releases/global-esports-market-report-2021-featuring-activision-blizzard-modern-times-group-tencent-valve-electronic-arts-301398111.html (Accessed: 26 June 2022).

Reverse Winter Studios (2021) *Advanced Server Manager in Code Plugins - UE Marketplace*, *Unreal Engine*. Available at: https://www.unrealengine.com/marketplace/en-US/product/server-manager/reviews (Accessed: 25 November 2021).

Riot Games (2020) *Valorant* [Video Game]. Riot Games

Rychkova, A. *et al.* (2020) 'Orbital Battleship: A Multiplayer Guessing Game in Immersive Virtual Reality', *Journal of Chemical Education*, 97(11), pp. 4184–4188. Available at: https://doi.org/10.1021/acs.jchemed.0c00866.

Sabadello, M. (2006) *History of electronic games*. (Doctoral dissertation, Master's Thesis). Institute of Computer Graphics and Algorithms, Vienna University of Techn. Available at: https://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.478.4522&rep=rep1&type=pdf.

Schwartz, J., Stagner, J. and Morrison, W. (2006) 'Kid's Programming Language (KPL)', in *ACM SIGGRAPH 2006 Educators Program*. New York, NY, USA: Association for Computing Machinery (SIGGRAPH '06), pp. 52-es. Available at: https://doi.org/10.1145/1179295.1179348.

Semanová, M. (2020) *Emerging Technologies and Video Game Industry*. Bachelor's Thesis. JAMK University of Applied Sciences. Available at: https://core.ac.uk/download/pdf/326045969.pdf.

Sim, T.Y. and Lau, S.L. (2018) 'Online Tools to Support Novice Programming: A Systematic Review', in *2018 IEEE Conference on e-Learning, e-Management and e-Services (IC3e)*, pp. 91–96. Available at: https://doi.org/10.1109/IC3e.2018.8632649.

Simon-Kucher and Partners (2020) *Impact of COVID-19 on the frequency of playing multiplayer video games worldwide as of*, *Statista Inc*. Available at: https://www.statista.com/statistics/1188549/covid-gaming-multiplayer/ (Accessed: 22 July 2021).

Sneaky Kitty Game Dev (2020) *Unreal Engine Online Subsystem (steam)*. Available at: https://www.youtube.com/playlist?list=PLnHeglBaPYu9Iyr5jwiCEdKopTq0zvuK7 (Accessed: 15 November 2021).

Spurling, A. (2005) 'QoS Issues for Multiplayer Gaming', *Semantic Scholar*, p. 7. Available at: https://www.semanticscholar.org/paper/QoS-Issues-for-Multiplayer-Gaming-Spurling/71e8f8aca6d6a1ebe33a8636175851fae4e1aeb9#paper-header.

Stagner, A.R. (2013) *Unity Multiplayer Games*. Packt Publishing. Available at: https://www.packtpub.com/product/unity-multiplayer-games/9781849692328

Statista (2022a) *Video Games - Worldwide* | *Statista Market Forecast*, *Statista*. Available at: https://www.statista.com/outlook/dmo/digital-media/video-games/worldwide (Accessed: 29 July 2022).

Statista (2022b) *Online Games - Worldwide* | *Statista Market Forecast*, *Statista*. Available at: https://www.statista.com/outlook/dmo/digital-media/video-games/online-games/worldwide (Accessed: 29 July 2022).

Stephenson, N. (2008) 'A Sense of Wonder: Speculative.', in *Virtual Worlds, Real Libraries: Librarians and Educators in Second Life and Other Multi-user Virtual Environment*. Information Today, Inc., p. 115. Available at: https://books.google.co.uk/books?hl=en&lr=&id=jlOPKr9W61QC&oi=fnd&pg=PA113&dq=Stephenson,+N+2008.+A+Sense+of+Wonder:+Speculative.+Virtual+Worlds,&ots=AGtTW8q_n_&sig=V6qkVRaJWnnCqe-LNOKqJsNfwmc.

Sneaky Kitty Game Dev (2020) *Unreal Engine Online Subsystem (steam)*. Available at: https://www.youtube.com/playlist?list=PLnHeglBaPYu9Iyr5jwiCEdKopTq0zvuK7 (Accessed: 15 November 2021).

Swain, L. (2013) 'MMO: Massive micro-transactions online', *Computers for Everyone*, p. 156. Available at: https://doi.org/10.1.1.867.2936.

Syakah (2021) *Top 10 Most Played eSports Games (2021)*, *Counter Arts*. Available at: https://medium.com/counterarts/top-10-most-played-esports-games-2021-c5a8cfc56532 (Accessed: 3 March 2022).

Thomala, L.L. (2022) *Mobile game sales revenue share in China's gaming market 2012-2021*, *Statista*. Available at: https://www.statista.com/statistics/1058001/china-mobile-game-share-in-gaming-industry/ (Accessed: 11 June 2022).

Tyler, D. (2017) 'How to Choose the Best Video Game Engine', 11 March. Available at: https://www.gamedesigning.org/career/video-game-engines/ (Accessed: 15 September 2021).

Unreal Engine (2021a) *Networking Overview*, *Unreal Engine*. Available at: https://docs.unrealengine.com/4.26/en-US/InteractiveExperiences/Networking/Overview/ (Accessed: 28 November 2021).

Unreal Engine (2021b) *FindFriendSession, Unreal Engine API Reference*, *Unreal Engine API Reference*. Available at: https://docs.unrealengine.com/4.26/en-US/API/Plugins/OnlineSubsystem/Interfaces/IOnlineSession/FindFriendSession/ (Accessed: 5 January 2022).

Unreal Engine (2021c) *Multiplayer in Blueprints*, *Unreal Engine API Reference*. Available at: https://docs.unrealengine.com/4.26/en-US/InteractiveExperiences/Networking/Blueprints/ (Accessed: 28 November 2021).

Vohera, C. *et al.* (2021) 'Game Engine Architecture and Comparative Study of Different Game Engines', in *2021 12th International Conference on Computing Communication and Networking Technologies (ICCCNT)*. Available at: https://doi.org/10.1109/ICCCNT51525.2021.9579618.

von der Heiden, J.M. *et al.* (2019) 'The Association Between Video Gaming and Psychological Functioning', *Frontiers in Psychology*, 10. Available at: https://doi.org/10.3389/fpsyg.2019.01731.

Wang, A.I. *et al.* (2008) 'Issues related to mobile multiplayer real-time games over wireless networks', in *2008 International Symposium on Collaborative Technologies and Systems*. Irvine, CA, USA: IEEE, pp. 237–246. Available at: https://doi.org/10.1109/CTS.2008.4543937.

Weiss, B.A. (2014) *The Lost Vikings*, *All Game*. Available at: https://web.archive.org/web/20141114232128/http://www.allgame.com/game.php?id=11754 (Accessed: 8 August 2021).

Westerdahl, M. (2019) *Challenges in video game development - What does Agile management have to do with it?* Malmo University. Available at: https://doi.org/10.1177/1046878109340295.

Witkowski, W. (2020) *Videogames are a bigger industry than movies and North American sports combined, thanks to the pandemic*, *MarketWatch*. Available at: https://www.marketwatch.com/story/videogames-are-a-bigger-industry-than-sports-and-movies-combined-thanks-to-the-pandemic-11608654990 (Accessed: 28 May 2021).

Wodarczyk, S. and Mammen, S. (2020) 'Emergent Multiplayer Games', in *2020 IEEE Conference on Games (CoG)*. Osaka, Japan, pp. 33–40. Available at: https://doi.org/10.1109/CoG47356.2020.9231834.

Yuzyk, M. and Seidner, P. (2022) 'E-Sports Development', in N. Kryvinska and M. Greguš (eds) *Developments in Information & Knowledge Management for Business Applications: Volume 5*. Cham: Springer International Publishing, pp. 615–648. Available at: https://doi.org/10.1007/978-3-030-97008-6_28.