University of Reading

Department of Computer Science

Creating a Top-Down Online Shooter

Author: Jaweed Inayathulla

Supervisor: Prof Shuang-Hua Yang

A report submitted in partial fulfilment of the requirements of
the University of Reading for the degree of
Bachelor of Science in *Computer Science*

September 14, 2024

Declaration

I, Jaweed Inayathulla, of the Department of Computer Science, University of Reading, confirm that this is my own work and figures, tables, equations, code snippets, artworks, and illustrations in this report are original and have not been taken from any other person's work, except where the works of others have been explicitly acknowledged, quoted, and referenced. I understand that if failing to do so will be considered a case of plagiarism. Plagiarism is a form of academic misconduct and will be penalised accordingly.

I give consent to a copy of my report being shared with future students as an exemplar.

I give consent for my work to be made available more widely to members of UoR and public with interest in teaching, learning and research.

Jaweed Inayathulla
September 14, 2024

Abstract

Tank Tactics is an engaging online multiplayer tank battle game developed using the Unity game engine and its multiplayer technologies, including the Netcode for GameObjects (NGO) framework and Unity Gaming Services (UGS) Relay & Lobby. The project aimed to provide an enjoyable and accessible multiplayer experience for up to 20 players simultaneously, addressing the common challenges of traditional online games, such as the need for manual port forwarding and IP address sharing.

Through an extensive literature review, the research explored the history of multiplayer gaming, tracing its evolution from early electronic games to the rise of modern online multiplayer experiences. It examined the challenges faced by developers in creating online multiplayer games, including connectivity issues, complex network configurations, scalability, and security concerns. The study investigated various network topologies, such as client-server and peer-to-peer models, leading to the adoption of a client-hosted listen server model for Tank Tactics, balancing the advantages of both architectures.

The methodology involved leveraging Unity's powerful game development tools, including its intuitive editor, component-based architecture, and scripting capabilities using C#. The NGO framework and UGS services were implemented to enable seamless multiplayer connectivity and self-hosting, eliminating the need for manual port forwarding or IP address sharing. The game's core gameplay mechanics, including intuitive tank controls, precise shooting, and a dynamic coin system, were meticulously designed to deliver an engaging and rewarding experience.

The results showcased the successful integration of seamless multiplayer connectivity, engaging gameplay mechanics, and advanced features such as a real-time leaderboard, mini-map, healing zones, and bounty coins. Player feedback and survey results validated the effectiveness of the development efforts, demonstrating the game's ability to address initial challenges and deliver an accessible and polished multiplayer experience.

The project's findings highlighted the significance of leveraging modern technologies and frameworks to overcome traditional multiplayer game development challenges. The modular and well-structured approach to development ensured maintainability and extensibility, facilitating future expansions and improvements. Potential areas for future work include cross-platform support, advanced matchmaking systems, social features, performance optimization, and monetization strategies.

Overall, Tank Tactics exemplified the power and versatility of Unity's multiplayer technologies in creating an accessible and engaging online multiplayer gaming experience.

Keywords: Online Multiplayer Game, Unity, Netcode for GameObjects, Unity Gaming Services, Tank Battle, Networking

Report's total word count: 12402

Acknowledgements

I want to express my sincere gratitude to Prof Shuang-Hua Yang, my supervisor and lecturer, for his invaluable guidance, and support.

I also would like to express my sincere gratitude to James Maslin and Ahmed Bedair for their invaluable contributions to the Tank Tactics project. Their assistance during the development and testing phases was instrumental in shaping the final product into an engaging and polished online multiplayer experience.

James and Ahmed dedicated their time and effort to participate in numerous demo sessions and playtesting iterations. Their keen insights and constructive feedback were pivotal in identifying areas for improvement, refining gameplay mechanics, and enhancing the overall user experience. Their willingness to explore every aspect of the game allowed Tank Tactics to uncover and rectify bugs, ensuring a smooth and enjoyable multiplayer environment for all players.

Contents

List of Abbreviations

NGO	Netcode for GameObjects
UGS	Unity Gaming Services
NAT	Network Address Translation
PLATO	Programming Logic for Automatic Tech Operations
MUDs	MultiUser Dungeons
LAN	Local Area Network
FPS	First-Person Shooter
TCP	Transmission Control Protocol
UDP	User Datagram Protocol
QoS	Quality of Service
UPnP	Universal Plug and Play
P2P	Peer-to-Peer
RPCs	Remote Procedure Calls
IDE	Integrated Development Environment

Chapter 1

Introduction

1.1 Background

Multiplayer online games have seen a significant rise in popularity and player engagement in recent years. However, in traditional multiplayer game development often faces challenges in such as then need for manual port-forwarding and IP address sharing, which can hinder accessibility and growth. Tank Tactics aims to address these limitations by leveraging Unity's Netcode for GameObjects (NGO) framework and Unity Gaming Services (UGS) Relay & Lobby, which enables seamless multiplayer connectivity and self-hosting.

1.1.1 Tank Tactics Game Overview

The project focuses on the development of an online multiplayer game called Tank Tactics. The gameplay of Tank Tactics is inspired by popular .io style games such as Agar.io and Slither.io, as well as classic tank-based games like Tank Trouble and Wii Tanks. The game features a top-down shooter mechanic, where the player controls tanks and engages in combat.

1.1.2 Unity Game Engine and Multiplayer Technologies

Tank Tactics is developed using Unity, a widely used game engine platform that provides a wide range of tools and features for game development. To enable seamless multiplayer functionality, the project leverages two key technologies: Unity's Netcode for GameObjects (NGO) framework and Unity Gaming Services (UGS) Relay & Lobby.

Netcode for GameObjects (NGO)

Netcode for GameObjects (NGO) is a high-level API provided by Unity that simplifies the process of building networked games. It abstracts away many of the low-level details of networking, allowing developers to focus on creating the gameplay logic and synchronising objects across the network, NGO handles tasks such as network communication, object spawning, and state synchronisation, making it easier to create multiplayer games.

Unity Gaming Services (UGS) Relay & Lobby

Unity Gaming Services (UGS) Relay & Lobby is a service provided by Unity that handles the hosting and matchmaking aspects of multiplayer games. UGS Relay provides a secure and scalable infrastructure for hosting multiplayer game sessions, eliminating the need for players to manually configure port-forward or share IP addresses. UGS Lobby, on the other hand, facilitates the matchmaking process allowing players to find and join available game sessions.

1.1.3 Key Feature of Tank Tactics

To keep the gameplay engaging and dynamic the game incorporates various features including:

- Leaderboard: A real-time leaderboard that tracks top-performing players, creating a sense of competitiveness among players.
- Mini-map: A miniaturised map that provides players situational awareness, allowing them to monitor the map and the position of other players.
- Healing Zone: Designated areas on the map where players can replenish their tank's health.
- Collecting Coins: Players can collect coins scattered around the map to earn points on the leaderboard. Collected coins can be used as currency to shoot other players or access healing zones.
- Bounty Coins: These coins are dropped when players have collected 100 or more regular coins which players can collect.

1.2 Problem statement

Traditional online multiplayer games often present technical challenges that can negatively impact the player experience and hinder the accessibility and growth of online games. These challenges include:

- The requirement for manual port-forwarding: To enable incoming connections for the game, players frequently need to change their router settings to open particular ports manually. For many players, this can be time-consuming and technically difficult Glazer and Madhav (2015).
- IP address sharing: Traditional online games frequently require sharing IP addresses to create direct connections between players. Players must provide their IP addresses to others, which presents privacy and security issues Parameswaran and Whinston (2007).
- Complex network configuration: Complex network settings may be necessary to set up and maintain a reliable multiplayer gaming environment. For those without extensive networking knowledge, navigating firewall settings, network address translation (NAT), and other technical issues may be challenging for players Tikhomirov (2023).

These technical hurdles create barriers to entry for players, limiting the accessibility of online multiplayer games Goguen (2023). As a result, many players might be prevented from participating in online gaming, which might hinder the growth and popularity of multiplayer games. The success and broad uptake of online games depend on resolving these issues and offering a smooth, intuitive, multiplayer gaming experience Smed et al. (2002).
By utilising NGO framework and UGS Relay & Lobby, Tank Tactics seeks to address these issues and provide users with a more engaging and accessible multiplayer gaming experience.

1.3 Aims and objectives

Aims: The primary aim of this project is to use NGO framework and UGS Relay & Lobby to build Tank Tactics. The game aims to provide an enjoyable multiplayer experience for up to 20 players simultaneously, addressing the common challenges of traditional online games, such as the need for manual port forwarding and IP address sharing.

Objectives:

1. Implement Unity NGO framework and UGS Relay & Lobby service to enable seamless self-hosting and real-time multiplayer connectivity, eliminating players' need to configure their networking settings or share IP addresses.
2. Develop the core gameplay mechanics of Tank Tactics, including intuitive tank movements, accurate shooting mechanics, and a dynamic coin collection system.
3. Incorporate advanced features that enhance the overall gameplay experience, such as:
   - Leaderboard
   - Min-map
   - Healing Zones
   - Bounty Coins
4. Ensure the game's overall design and mechanics are inspired by popular .io style games and classic tank-based titles, providing players an engaging and familiar gaming experience.

1.4 Solution approach

The Solution approach for developing Tank Tactics multiplayer game using Unity's Network for GameObject (NGO) framework and Unity Gaming Services (UGS) Relay & Lobby. Involves the following steps:

1. Implementing the NGO framework and UGS Relay & Lobby service to enable seamless multiplayer connectivity and self-hosting without the need for manual port-forwarding or IP address sharing. This eliminates common technical barriers and provides an accessible multiplayer experience for players.
2. Developing the core gameplay mechanics of Tank Tactics, including intuitive tank movements, accurate shooting mechanics, and a dynamic coin collection system. These mechanics will be designed to provide an engaging and enjoyable gameplay experience.
3. Incorporating advanced features such as a real-time leaderboard, mini-map, healing zones, and bounty coins to enhance the overall gameplay experience and keep players engaged. These features add depth and variety to the game, encouraging players to strategise and compete.
4. Ensuring the game's overall design and mechanics are inspired by popular .io style games and classic tank-based titles. This familiarity will help attract players and provide an intuitive gaming experience.
5. Conducting thorough testing and optimization to ensure a smooth, stable, and enjoyable multiplayer experience for up to 20 players simultaneously.

1.5 Organization of the report

This dissertation is structured as follows: Chapter 2, Literature Review, explores the existing research and solutions related to multiplayer game development, networking frameworks, and game design principles relevant to Tank Tactics. Chapter 3, Methodology, describes the design and implementation process of Tank Tactics, including the use of the NGO framework, UGS Relay & Lobby service, and the development of core gameplay mechanics and advanced features. Chapter 4, Results, presents the outcomes of the development process, showcasing the implemented features, multiplayer functionality, and overall gameplay experience. Chapter 5, Discussion and Analysis, evaluates the results, discussing the significance of the implemented features, the effectiveness of the solution approach, and any limitations encountered during the development process. Chapter 6, Conclusions and Future Work summarizes the key findings and contributions of the project and outlines potential areas for future development and improvement of Tank Tactics. Chapter 7, Reflection, offers a personal reflection on the learning experience gained throughout the project, discussing challenges faced, lessons learned, and the impact of the project on personal growth and future work in game development.

Chapter 2

Literature Review

This chapter provides a comprehensive literature review exploring the key aspects related to the development of Tank Tactics, an online multiplayer game built using the Unity game engine. The review is divided into four main sections, each focusing on a specific area of interest. Section 2.1 traces the evolution of multiplayer gaming, contextualizing Tank Tactics within the broader history of gaming and networking. Section 2.2 examines the challenges developers face when creating online multiplayer games and discusses how Tank Tactics addresses these challenges. Section 2.3 explores the choice of network topology, comparing client-server and peer-to-peer models, and explains how Tank Tactics employs a client-hosted listen server model. Finally, Section 2.4 provides an in-depth look at the Unity game engine and its multiplayer technologies, covering scripting, multiplayer solutions, Unity Gaming Services (UGS), limitations, future developments, and how Tank Tactics leverages these technologies to create a unique and engaging experience.

2.1 History of Gaming and Networking

Section 2.1 provides a comprehensive overview of the history of gaming and networking, tracing the evolution of multiplayer gaming from its early beginnings to the present day. By examining the key milestones and technological advancements that have shaped the multiplayer gaming landscape, this section aims to contextualise the development of Tank Tactics. The subsection in 2.1 covers various aspects of multiplayer gaming history, including early multiplayer games (2.1.1), the role of mainframe computers and early computer networks (2.1.2), the impact of MultiUser Dungeons (MUDs) and arcade multiplayer gaming (2.1.3), the impact of LAN-based (2.1.4) and Internet-based gaming (2.1.5), and recent trends and challenges in the gaming industry (2.1.6). The section, concludes by discussing how Tank Tactics builds upon this rich legacy while addressing contemporary challenges in multiplayer game development (2.1.7).

2.1.1 Early Multiplayer Games

The concept of multiplayer gaming dates back to the earliest days of electronic games. In 1958, William Higinbotham created Tennis for Two, which allowed two players to compete against each other using separate controllers connected to an oscilloscope Armitage et al. (2006). This groundbreaking game showcased the potential for social interaction through electronic gaming. A few years later, in 1961, students at the Massachusetts Institute of Technology (MIT) developed Spacewar, a multiplayer space combat game that ran on a PDP-1 computer

Armitage et al. (2006). Spacewar introduced concepts such as player-versus-player combat and shared-screen multiplayer.

2.1.2 Mainframe Computers and Early Computer Networks

The 1970s and 1980s witnessed the emergence of online gaming communities through the use of mainframe computers and early computer networks, PLATO (Programming Logic for Automatic Teach Operations) was a notable example of a mainframe-based system that supported multiplayer gaming Armitage et al. (2006). Developed at the Univeristy of Illinois, PLATO allowed users to log into the system remotely and interact through various applications, including multiplayer games like Empire and Airflight. These games followed a centralised architecture, with the mainframe handling the primary computation and communication while players connected using terminals with limited processing power Armitage et al. (2006). PLATO's multiplayer games demonstrated the feasibility of remote gaming and laid the foundation for future online gaming platforms.

2.1.3 MultiUser Dungeons (MUDs) and Arcade Multiplayer Gaming

The 1980s saw the rise of MultiUser Dungeons (MUDs), text-based virtual worlds where players could interact with the environment and each other through commands entered via a terminal Bartle (1990). MUDs were inspired by the popularity of tabletop role-playing games like Dungeons & Dragons and combined elements of storytelling, exploration, and social interaction Kushner (2004). Games like MUD1, developed by Richard Bartle and Roy Trubshaw at the Univerity of Essex, allowed players to connect to a central server using Telnet, a protocol for remote terminal communication Armitage et al. (2006). MUDs introduced concepts such as persistent worlds, player progression, and social interaction, which would later be expanded upon in graphical MMORPGs Bartle (2004).

MUDs utilised a client-server architecture, connecting players to the MUD server using a Telnet program. While the Telnet client itself was quite basic, the server-side game could be quite sophisticated, tracking the state of the virtual world, managing player's interactions and updating the game state in real time. The popularity of MUDs in academic circles helped spur innovations in game design, online communities, and server infrastructure.

Concurrent with the development of MUDs, the golden age of arcade gaming in the 1970s and 1980s brought multiplayer gaming experience to the masses. Arcade machines, which were coin-operated and located in public spaces like shopping malls and amusement parks, offered a social gaming experience where players could compete against each other in the same physical location Donovan and Garriott (2010). Games like Pong (1972) and Space Invaders (1978) popularised the concept of head-to-head competition, while titles like Atari Football (1978) and Gauntlet(1985) introduced cooperative multiplayer gameplay Armitage et al. (2006).

While the technology underpinning arcade games was relatively simple compared to modern standards, the games showcased various innovative game mechanics and design elements. From pattern-based gameplay of Space Invaders to the scrolling levels of Super Mario Bros., arcade games laid the foundation for many of the game genres and conventions that are granted for today.

The combination of MUDs and arcade games in the 1980s represented a significant step forward in the evolution of multiplayer gaming. MUDs showed the compelling multiplayer experience could be delivered over a network connection, while arcades demonstrated the social appeal and commercial viability of gaming as a shared activity.

2.1.4 LAN-based Multiplayer Gaming

The introduction of personal computers and the development of local area networks (LAN) technology in the 1990s revolutionised multiplayer gaming. LAN allowed multiple computers to be interconnected within a limited geographic area, enabling players to connect and play games. One of the most influential games of this era was Doom (1993), a first-person shooter (FPS) that introduced a networked multiplayer game Armitage et al. (2006). Doom allowed up to four players to compete against each other over a LAN using the IPX protocol, which was commonly used in Novell networks Kushner (2004). However, Doom's reliance on broadcast packets for communication led to network congestion issues, prompting the development of more efficient networking models, in subsequent games Armitage et al. (2006). LAN parties, where players gathered with their computers to play multiplayer games together, became increasingly popular.

2.1.5 Internet-based Multiplayer Gaming

The release of Quake in 1996 marked a turning point in multiplayer gaming. Developed by id Software, Quake introduced a client-server architecture that allowed it to connect to dedicated game servers over the Internet Armitage et al. (2006). This architecture addressed the limitation of peer-to-peer networking used in earlier games like Doom and enabled the creation of persistent online gaming communities Kushner (2004). Quake's popularity led to the emergence of organised clans, competitive tournaments, and the rise of professional gaming Armitage et al. (2006). The game's success also sparked the development of user-created modifications (mods), which expanded the game's content and gameplay possibilities Kushner (2004). Quake's impact on the gaming industry cannot be overstated, as it laid the groundwork for large-scale, Internet-based multiplayer games.

2.1.6 Recent and Future of Multiplayer Gaming

The rapid advancements in technology have significantly improved the online gaming industry, reshaping the landscape and paving the way for innovative gaming experiences. The increasing accessibility and affordability of high-speed internet connection, coupled with the expansion of powerful gaming devices, have filed the growth of multiplayer gaming on a global scale.

Mobile Gaming

One of the most notable trends in recent years has been the rise of mobile gaming. The widespread adoption of smartphones and tablets has made gaming more accessible than ever before, allowing players to engage in multiplayer experiences on the go. The integration of social features and multiplayer capabilities in mobile games has fostered a sense of community and competitiveness among players, driving engagement and further fueling the market's growth.

Virtual Reality (VR) and Augmented Reality (AR)

Another key development in the multiplayer gaming space is the advancement of virtual reality (VR) technologies. VR and AR have the potential to revolutionize the way players interact with virtual works and each other. With the development of more powerful hardware and sophisticated software, VR headsets have become more accessible and user-friendly offering players a highly immersive and engaging experience ogerman (2021). The integration of haptic feedback and motion tracking technologies further enhances the sense of presence and interaction within these virtual environments, AR games such as Pokemon Go, have also gained significant popularity, blurring the lines between the real world and the virtual world by superimposing digital elements onto the player's surroundings ogerman (2021).

Challenges

However, the rapid growth of online gaming has also brought forth new challenges that need to be addressed. One of the most pressing concerns is data privacy and security. With millions of players engaging in online gaming platforms the amount of personal data being collected and stored has reached unprecedented levels *The Future of Online Gaming: Innovations and Challenges* (2023). Players are increasingly worried about the potential for their personal information to be compromised or exploited by cybercriminals. Game developed and publishers must prioritise data protection and implement robust security measures to safeguard player information and maintain trust within the gaming community.

Another challenge facing the online gaming industry is the need for greater inclusivity and accessibility. Cross-platform compatibility has emerged as a crucial factor in promoting inclusivity, allowing players to connect and play with others regardless of the device they use *The Future of Online Gaming: Innovations and Challenges* (2023).

2.1.7 Tank Tactics: Building Upon the Legacy of Multiplayer Gaming

Tank Tactics builds upon the rich history of multiplayer gaming while incorporating modern technologies and design principles to create a unique and engaging experience. The game draws inspiration from early multiplayer titles such as Tennis for Two and Spacewar, which pioneered the concept of competitive gameplay between multiple players. However, Tank Tactics leverages the power of modern game engines, like Unity, and advanced networking frameworks such as NGO and UGS Relay & Lobby, to provide a seamless and accessible multiplayer experience.

In contrast to the early multiplayer games that were limited by the technology of their time, Tank Tactics takes advantage of the widespread availability of high-speed internet connections and the increase of gaming devices to reach global audience. The game's focus on real-time, fast-paced tank combat draws inspiration from classic titles like Tank Trouble and Wii Tanks while introducing modern features such as leaderboards, mini-map, and dynamic gameplay elements like healing zones and bounty coins.

Furthermore, Tank Tactics differentiated itself from traditional multiplayer games by eliminating the need for manual port-forwarding and IP address sharing, which were common challenges in the early days of online gaming. By utilising Unity's NGO framework and UGS Relay & Lobby, the game provides a seamless and user-friendly multiplayer experience, allowing players to easily connect and play.

In summary, Tank Tactics represents a modern take on the multiplayer gaming legacy, combining classic gameplay elements with cutting-edge technologies and design principles to deliver an engaging accessible experience for players.

2.2 Online Multiplayer Game Development Challenges

Developing an online multiplayer game presents a unique set of challenges compared to single-player games. Dongen (2015) highlights that adding online multiplayer functionality to a game can double the amount of work required for programming. This section explores the various challenges developers face when creating online multiplayer games, including connecting players (2.2.1), manual port forwarding and IP sharing (2.2.2), complex network configurations (2.2.3) scalability and performance (2.2.4) and security (2.2.5). The section concludes by discussing how Tank Tactics addresses these challenges (2.2.6).

2.2.1 Connecting Players

Connecting players over the internet is a complex task that involves multiple layers of communication. At the most basic level, players' devices must establish a connection using a specific network protocol, such as Transmission Control Protocol (TCP) or User Datagram Protocol (UDP) Glazer and Madhav (2015). The choice of protocol depends on the game's requirement for speed, reliability, and error correction. Once a connection is established, the game must manage the flow of data between players, ensuring that updates are sent and received on time.

Organising the network topology is another crucial aspect of connecting players. Developers must choose between a client-server architecture, where a central server manages the game state and communication between players, or a peer-to-peer architecture, where players communicate directly with each other Tikhomirov (2023). Each approach has its advantages and disadvantages in terms of scalability, performance, and security.

Network Address Translation (NAT) is a common issue that complicates player connectivity. NAT is a technique used by routers to allow multiple devices on a local network to share a single public IP address. This can make it difficult for players to establish a direct connection, as incoming traffic must be correctly routed to the appropriate device Goguen (2023). Developers must implement techniques such as NAT traversal or relay servers to overcome this challenge.

2.2.2 Manual Port Forwarding and IP Sharing

Port forwarding is a technique used to allow incoming connection to a specific device on a local network. In the context of online gaming, players may need to manually configure their routers to forward incoming traffic on specific ports to their gaming devices Glazer and Madhav (2015). This process can be technically challenging, requiring the player to access their router's settings and correctly input the necessary information. Incorrect configuration can lead to connectivity issues or even expose the player's devices to security risks.

IP sharing is another common requirement in traditional online multiplayer games. To establish a direct connection between players, each player must know the IP address of the other player Smed et al. (2002). This can raise privacy and security concerns, as the player must share their IP addresses with others, potentially exposing themselves to malicious actors. Developers can mitigate these risks by implementing secure communication channels and using techniques like encryption and authentication.

2.2.3 Complex Network Configurations

Setting up a reliable multiplayer gaming environment often requires players to configure their network settings, including firewall rules, port forwarding, and Quality of Service (QoS) settings Tikhomirov (2023). Firewalls are designed to block unauthorised incoming traffic, which can interfere with the game connections if not properly configured. Players may need to create exceptions in their firewall settings to allow the game to communicate with other players and servers.

NAT can also complicate the setup process, as players may need to configure their routers to see specific NAT types (e.g. open, moderate, or strict) to ensure compatibility with the game's networking requirements Goguen (2023). Some games may require players to enable Universal Plug and Play (UPnP) or manually configure port forwarding to establish the connection.

Developers can simplify the setup process by providing clear instructions, automatic configuration tools, or implementing techniques like NAT traversal and relay servers to minimise the need for manual configuration Glazer and Madhav (2015). However, the wide variety of network setups and hardware configurations makes it challenging to create a one-size-fits-all solution.

2.2.4 Scalibility and Performance

As online multiplayer grows in popularity, they must be able to handle an increasing number of concurrent players without sacrificing performance. Scalability refers to a game's ability to accommodate a growing player base while maintaining a high level of service Smed et al. (2002). This requires careful planning and optimisation of the game's architecture, including server infrastructure, network protocol, and game logic.

Developers must choose the appropriate network topology (e.g. client-server or peer-to-peer) and design their server infrastructure to handle the expected player load Glazer and Madhav (2015). This may involve using techniques like load balancing, sharing, or cloud computing to distribute the workload across multiple servers, The game's network protocol must also be optimised to minimise latency and packet loss, ensuring a smooth and representative gaming experience.

Performance optimisation is crucial for maintaining player satisfaction and retention. Developers must continuously monitor and profile the game's performance, identifying and addressing bottlenecks in the game logic, network communication, and rendering pipeline Tikhomirov (2023). This may involve techniques like data compression, interpolation, or client-side prediction to minimise the impact of network latency on gameplay.

2.2.5 Security

Online multiplayer games must also protect against security threats, such as denial-of-service (DoS) attacks, which can overload servers and disrupt gameplay, and data breaches, which can expose player information and undermine trust in the game and its developers Smed et al. (2002). Developers must follow best practices for secure coding, regular patch vulnerabilities, and employ strong authentication and authorisation mechanisms to protect player accounts and data.

Balancing security and performance is an ongoing challenge, as security measures can introduce overhead and latency that may impact gameplay. Developers must carefully consider the trade-offs and implement security solutions that provide necessary protection without compromising the player experience Glazer and Madhav (2015).

2.2.6 Tank Tactics: Addressing Multiplayer Game Development Challenges

Tank Tactics addresses several of the challenges mentioned in the previous subsections by leveraging modern technologies and frameworks. By utilising the NGO framework and UGS Relay & Lobby, the game eliminates the need for manual port-forwarding and IP address sharing. This significantly reduces the technical barriers for players and improves accessibility, as players no longer need to configure their routers or share personal information to establish connections.

The use of UGS Relay & Lobby also simplifies the process of connecting players, as it handles the complexities of NAT traversal and provides a secure, scalable infrastructure for hosting multiplayer game sessions. This allows players to easily find and join available game sessions without the need for complex network configurations.

Furthermore, by leveraging the capabilities of the Unity game engine and its associated services, Tank Tactics can focus on optimising performance and scalability. The game can take advantage of Unity's built-in features for efficient network communication, object synchronisation, and game state management, reducing the development effort required to create a smooth and responsive multiplayer experience,

In terms of security, Tank Tactics benefits from the security measures and best practices implemented by Unity and its services The use of secure communication protocols, encryption, and authentication mechanisms helps protect player data and prevent unauthorised access to game servers.

2.3 Network Topologies in Online Multiplayer Games

The choice of network topology is a fundamental design decision when engineering a networked multiplayer game. The network topology describes how the computer and gaming devices are connected over the network Smed et al. (2002). The two main types of network topologies used in online multiplayer games are client-server and peer-to-peer.

2.3.1 Client-Server Topology

In a client-server topology, players connect to a central server that coordinates the game state and relays data between players. The server is the authoritative source of the game state. Clients send their input commands to the server, and the server validates these commands, simulates the game, and sends back the updated state to the clients Glazer and Madhav (2015).

Advantages of client-server topology include:

- Better security and cheat prevention since a trusted dedicated server has authority over the simulation

- Ability to handle a large number of players (scalability)
- Simplifed game state management and synchronisation

Disadvantages of client-server topology include:

- Cost of running dedicated servers
- Server being a single point of failure
- Potential for higher latency due to the extra hop between clients and server

Tribes Networking Model:

Starsiege: Tribes, released in 1998, is an example of a game using a client-server model Frohnmayer and Gift (2000). In Tribes, a single central server maintains the authoritative game state and communicates with up to 128 clients. The ghost manager system on the server determines which objects are relevant to each client and only sends those. This relevant filtering was key to reducing bandwidth sage and scaling support to 128 concurrent players.

2.3.2 Peer-to-Peer Topology

In a peer-to-peer (P2P) topology, each player's computer connects directly to every other player's computer without a central server. Game state and simulation are distributed across all participating machines Smed et al. (2002).

Advantages of P2P topology include:

- Reduced infrastructure and hosting costs
- Potentially lower latency due to direct connections between peers
- No single point of failure

Disadvantages of P2P topology include:

- Increased complexity in game state management and synchronisation
- Difficulty in preventing cheating and ensuring fair play
- Limited scalability due to exponential growth in the number of connections.

Deterministic Lockstep Model:

Age of Empires, released in 1997, used a peer-to-peer topology with a deterministic lockstep networking model Bettner and Terrano (2001). In Age of Empires, the game does not send the game state over the network - instead, it sends the player commands. The game simulation is deterministic, meaning that as long as every peer reviewers every command in the same order, the simulation will not diverge.

Techniques used in the deterministic lockstep model include:

- Command Queue: Each peer maintains a queue of commands to execute. Commands are only executed once all payers have acknowledged receiving them.
- Turn Timers: The game is divided into turns, with each turn lasting a fixed amount of time (e.g. 200ms). Peers send their commands for the turn to each other and then simulate the turn deterministically.
- Sync Checking: peers periodically compare their game state to ensure synchronisation and detect any divergence.

The deterministic lockstep model works well for games with a relatively small number of players and low update frequency, such as turn-based or slow-paced real-time strategy games. However, it can introduce latency and may not scale well to fast-paced games or a large number of players.

2.3.3 Hybrid Topologies

Glazer and Madhav (2015) note that modern online games often use a combination or hybrid of client-server and peer-to-peer topologies to balance the advantages and disadvantages of each.

Listen Server:

In a listen server mode, one player acts as the server, and the rest connects as clients to them. This avoids the infrastructure cost of dedicated servers while still providing some of the benefits of the client-server model, such as improved security and state management. However, the host player may have an unfair advantage in terms of latency, and the game is dependent on the host's connection quality.

Broadcast Server:

Bettner and Terrano (2001) discuss using a broadcast server to allow Age of Empires to scale to more than 4 players in a peer-to-peer topology. In this model, a server is used to broadcast game state updates to all peers, but the peers still simulate the game deterministically based on the received state updates. This reduces the number of connections each peer needs to maintain, improving scalability.

2.3.4 Overall

Client-server and peer-to-peer network topologies each have strengths and weaknesses. Client-server is the most common modern approach for online multiplayer games due to its ability to robustly scale to more players and provide strong security cheat prevention. However, peer-to-peer remains viable for smaller-scale games and those that wish to avoid the infrastructure costs of dedicated servers.

Hybrid topologies that combine elements of client-server and peer-to-peer can offer a balance of their respective benefits. However, careful engineering is required with any topology to build a game that plays smoothly online, considering factors such as bandwidth usage, latency, state synchronisation, and cheat prevention.

2.3.5 Tank Tactics: Client-Hosted Listen Server Model

Tank Tactics employs a client-hosted listen server model, which is a type of hybrid topology that combines aspects of client-server and peer-to-peer architectures. In this model, one of the players acts as both a client and a server, hosting the game for other players who connect as clients.

The listen server model used in Tank Tactics builds upon the traditional client-server topology by eliminating the need for dedicated server infrastructure. This reduces hosting costs and allows players to set up matches more easily. However, it still maintains many of the benefits of client-server architecture, such as improved security and simplified game state management compared to a pure peer-to-peer topology.

Unlike the deterministic lockstep model used in Age of Empires, Tank Tactics does not rely on deterministic simulation across all clients. Instead, the listen server acts as the authoritative source of the game state, similar to a dedicated server in a client-server topology. This allows Tank Tactics to support faster-paced, real-time gameplay without the latency issues that can arise in a deterministic lockstep model.

However, the client-hosted listen server model also has some disadvantages compared to a dedicated client-server architecture. The host player may have an advantage in terms of latency, as they do not need to spend updates to themselves. Additionally, the game's performance and stability are dependent on the host player's connection quality.

Despite these limitations, the client-hosted listen server model, is a good fit for Tank Tactics, as it allows for more accessible multiplayer gaming while still providing a reasonably secure and manageable network architecture. This model is particularly well-suited for smaller-scale multiplayer games like Tank Tactics, where the number of players per match is limited, and the costs of running dedicated servers may outweigh the benefits.

2.4 Unity Game Engine and Multiplayer Technologies

This section provides an in-depth exploration of the Unity game engine and its multiplayer technologies, which form the foundation for the development of Tank Tactics. Begins with an overview of the Unity game engine (2.4.1), highlighting its user-friendliness, versatility, and widespread adoption among game developers. The section then explores Unity's scripting capabilities (2.4.2). Next, it examines the multiplayer solution offered by Unity (2.4.3), particularly the Netcode for GameObject (NGO) framework. The section also covers Unity Gaming Service (UGS) (2.4.4). The limitations and future development of Unity's multiplayer technologies are discussed (2.4.5), followed by a conclusion summarising the key points (2.4.6). Finally, the section concludes by examining how Tank Tactics leverages Unity's multiplayer technologies to create a unique and engaging online gaming experience (2.4.7)

2.4.1 Overview of Unity Game Engine

Unity is a widely used cross-platform game engine developed by Unity Technologies. Since its initial release in 2005, Unity has gained significant popularity among game developers, particularly indie developers Dealessandri (2020). The engine's user-friendliness, versatility, and support for both 2D and 3D games across multiple platforms have contributed to its widespread adoption.

Unity offers a comprehensive set of tools and features to streamline the game development process. The Unity editor provides a user-friendly interface with customisable layouts and various windows for scene editing, game object hierarchy management, asset management, and more *Unity's Interface* (n.d.). The engine employs a component-based architecture, where game objects are composed of reusable components *GameObjects* (n.d.). This modular approach simplifies development and promotes code reusability, making it easier for developers to create and iterate on their games.

2.4.2 Scripting in Unity

One of Unity's key strengths is its scripting capabilities. Unity supports the C# programming language for writing game logic and behaviour. Developers can create custom scripts by deriving from the MonoBehaviour class, which provides access to Unity's event functions and lifecycle methods *Unity Scripts* (n.d.). This allows for fine-grained control over game objects and their interactions, enabling developers to implement complex gameplay mechanics and systems.

Unity's scripting systems are designed to be accessible to developers with varying levels of programming experience. The engine provides a rich set of APIs and documentation, making it easier for developers to learn and utilise the scripting features effectively.

2.4.3 Multiplayer Solutions in Unity

In recent years, Unity has been actively expanding its multiplayer offerings to meet the growing demand for online multiplayer experiences. Unity Technologies has introduced various networking solutions to facilitate the development of networked games, making it easier for developers to create multiplayer functionality within their projects.

One of Unity's primary multiplayer solutions is the Netcode for GameObjects (NGO) framework, NGO is a high-level networking library that abstracts away the complexities of networking, allowing developers to focus on creating gameplay logic and synchronising objects across the network *Netcode* (n.d.). It leverages the Unity Transport package, a low-level networking library that manages connection using UDP and WebSockets *Unity Transport* (n.d.). NGO supports a client-server topology and provides features such as object spawning, state synchronisation, and remote procedure calls (RPCs) for seamless multiplayer functionality.

2.4.4 Unity Gaming Services (UGS)

To further simplify the multiplayer development process, Unity offers Unity Gaming Services (UGS). UGS is a suite of services and tools that address various aspects of multiplayer game development including multiplayer services, diagnostics, analytics, and monetisation *Unity UGS* (n.d.). Two notable services within UGS that are particularly relevant for multiplayer games are Relay and Lobby.

The Relay service enables players to connect through Unity's proxy relay server, eliminating the need for manual port forwarding and resolving issues related to NAT and firewalls Monkey (2023*b*). It simplifies the process of establishing connections between players, making it easier for developers to create an accessible multiplayer experience without the complexities of direct peer-to-peer connections.

The Lobby service provides a framework for players to find and join game sessions Monkey (2023*a*). It allows developers to create lobbies where players can gather before starting a match, facilitating matchmaking and player grouping. The Lobby services integrate seamlessly with the Relay service, enabling players to connect through the relay once they have joined a lobby. This combination of services streamlines the multiplayer experience for both developers and players.

2.4.5 Limitation and Future Developments

While Unity's multiplayer solutions are powerful and user-friendly, they may have limitations in terms of scalability and performance. NGO is primarily suitable for small to medium-scale games and may not be optimal for large-scale, high-performance titles Lemay (2022). However, Unity Technologies is continuously improving and expanding its multiplayer offerings to address these limitations and cater to a wider range of multiplayer game requirements.

As Unity continues to evolve and enhance its multiplayer capabilities, it is expected to play a significant role in shaping the future of multiplayer game development. With the growing demand for online multiplayer experiences, Unity's commitment to providing accessible and efficient multiplayer solutions will empower more developers to create engaging and immersive multiplayer games.

2.4.6 Conclusion

Unity's comprehensive toolset, user-friendly interface and multiplayer technologies make it an attractive choice for game developers looking to create online multiplayer experiences. The combination of NGO and UGS provides a solid foundation for implementing multiplayer functionality in games like Tank Tactics. As Unity continues to evolve and enhance its multiplayer capabilities, it is expected to play a significant role in shaping the future of multiplayer game development.

2.4.7 Unity in Tank Tactics

The Tank Tactics project has leveraged Unity's NGO framework and UGS to implement online multiplayer functionality. By utilising NGO's networking capabilities, Tank Tactics can synchronise the game state and enable real-time multiplayer gameplay, ensuring a smooth and responsive experience for players. This aligns with the project's goal of creating an engaging dynamic multiplayer environment as outlined in Chapter 1 (section 1.1.1).

Tank Tactics also integrates the Relay and Lobby services provided by UGS to facilitate player connection and matchmaking. These services streamline the process of finding and joining game sessions, eliminating the need for manual port forwarding and IP address sharing, as discussed in Chapter 2 (section 2.2.2). By leveraging these services, Tank Tactics addresses the challenges associated with traditional multiplayer game development, providing an accessible and user-friendly experience for players.

Unlike the deterministic lockstep model used in Age of Empires in chapter 2 (section 2.3.2), Tank Tactics employs a client-hosted listen server model (section 2.3.5). This approach allows for faster-paced, real-time gameplay without the latency issues that can arise in a deterministic lockstep model. However, it also introduces some disadvantages, such as potential latency advantages for the host player and dependence on the host's connection quality. Despite these limitations, the client-hosted listen server model is well-suited for smaller-scale multiplayer games like Tank Tactics, where the costs of running dedicated servers may outweigh the benefits.

The use of Unity's multiplayer technologies in Tank Tactics demonstrates the engine's effectiveness in creating online multiplayer games. The combination of NGO and UGS has streamlined the development process, allowing the project to focus on gameplay mechanics and player interactions while relying on Unity's robust networking infrastructure. This aligns with the project's objectives in chapter 1 (section 1.3) of implementing seamless multiplayer connectivity and developing engaging gameplay features.

2.5 Summary

The literature review conducted in this chapter provides a comprehensive overview of the key aspects related to the development of Tank Tactics, an online multiplayer game built using the Unity game engine. The review begins tracing the history of gaming and networking, highlighting the evolution of multiplayer gaming from early games like Tennis for Two and Spacewar to the rise of MUDs, arcade gaming, LAN-based gaming, and eventually Internet-based multiplayer gaming. This historical context sets the stage for understanding the challenges and opportunities faced by modern multiplayer game developers.

The review then goes into the specific challenges associated with developing online multiplayer games, such as connecting players, dealing with manual port forwarding and IP sharing, navigating complex network configurations, ensuring scalability and performance, and addressing security concerns. By examining these challenges, the review underscores the importance of leveraging modern technologies and frameworks, such as Unity's NGO framework and UGS Relay & Lobby, to overcome these obstacles and provide a seamless and accessible multiplayer experience for players.

The choice of network topology is identified as a fundamental design decision in multiplayer game development. The analysis compares the two main types of network topologies, client-server and peer-to-peer, discussing their respective advantages and disadvantages. It also explores hybrid topologies and explains how Tank Tactics employs a client-hosted listen server model, which combines aspects of both client-server and peer-to-peer architectures to balance the benefits and drawbacks of each approach.

The literature review provides an in-depth exploration of the Unity game engine and its multiplayer technologies, which form the foundation for the development of Tank Tactics. It highlights Unity's user-friendliness, versatility, and widespread adoption among game developers, as well as its powerful scripting capabilities using C#. The study examines Unity's primary multiplayer solution, the Netcode for GameObjects (NGO) framework, which abstracts away the complexities of networking and enables developers to focus on creating gameplay logic and synchronizing objects across the network. It also covers Unity Gaming Services (UGS), particularly the Relay and Lobby services, which facilitate player connection and matchmaking, streamlining the multiplayer experience for both developers and players.

While acknowledging the limitations of Unity's multiplayer solutions in terms of scalability and performance, the analysis emphasizes Unity Technologies' continuous efforts to improve and expand its multiplayer offerings. As Unity evolves and enhances its multiplayer capabilities, it is expected to play a significant role in shaping the future of multiplayer game development.

Finally, the literature review examines how Tank Tactics leverages Unity's multiplayer technologies to create a unique and engaging online gaming experience. By utilizing the NGO's networking capabilities and integrating UGS Relay & Lobby services, Tank Tactics addresses the challenges associated with traditional multiplayer game development, providing an accessible and user-friendly experience for players. The project's use of a client-hosted listen server model allows for faster-paced, real-time gameplay while minimizing the costs and complexities of running dedicated servers.

In conclusion, the literature review provides a solid foundation for understanding the context, challenges, and opportunities involved in developing an online multiplayer game like Tank Tactics using the Unity game engine. By leveraging the insights gained from this study, the Tank Tactics project can effectively navigate the complexities of multiplayer game development, and create an engaging and accessible gaming experience.

Chapter 3

Methodology

Text Here

3.1 Problem Description and Requirements

The primary problem that the Tank Tactics project aims to address is the development of an online multiplayer tank battle game that overcomes the limitations of traditional multiplayer games, such as the need for manual port forwarding and IP address sharing. This task presents several challenges, including the need for seamless multiplayer connectivity, the creation of engaging gameplay mechanics, and the incorporation of advanced features to enhance the overall gaming experience.

Developing an online multiplayer game like Tank Tactics requires a fast and efficient networking solution. The game must support multiple players interacting in real-time, which necessitates a secure, efficient, and low-level latency network. Any delay or lag in the game can significantly impact the user experience, making seamless networking crucial for the success of the project.

Another significant challenge is the creation of engaging gameplay mechanics. Tank Tactics should feature intuitive tank controls, precise shooting mechanics, and a dynamic coin collection system. These core gameplay elements must be designed to provide an enjoyable and engaging experience for players.

The incorporation of advanced features is also a key requirement for Tank Tactics. The game should include elements such as a real-time leaderboard, mini-map, healing zones, and bounty coins to add depth and variety to the gameplay. The specific context of the project also plays a vital role. Tank Tactics is designed to cater to fans of .io-style games and classic tank-based titles. The game should draw inspiration from these genres to create a familiar and intuitive experience for players. The target platform for the game is the Unity game engine, which influences the development process and the choice of networking technologies.

Moving on to the requirements of the project, the functional requirements include seamless multiplayer connectivity, engaging gameplay mechanics, and the incorporation of advanced features. Seamless multiplayer connectivity is essential for allowing up to 20 players to interact in the same game environment without the need for manual port forwarding or IP address sharing. Engaging gameplay mechanics, such as intuitive tank controls and precise shooting are necessary and advanced features, like leaderboards, and mini-maps enhance the overall gaming experience.

The non-functional requirements of the project include performance optimisation accessibility, and scalability. Performance optimisation is crucial for maintaining a smooth and responsive gaming experience for all players, even as the number of concurrent users increases. The game should also be accessible, with a user-friendly interface and minimal technical barriers to entry.

In summary, the development of Tank Tactics presents several challenges and requirements that need to be addressed. The aim is to create an engaging, competitive, and accessible online multiplayer tank battle game by leveraging the power of Unity's multiplayer technologies and carefully considering these factors.

3.2 Technologies

This section explores deeper into the key technologies employed in the development of Tank Tactics, providing a comprehensive overview of the Unity game engine, Visual Studio, C# programming language, Unity Netcode, and Unity Gaming Services. By leveraging these powerful tools and frameworks, the project aims to create a seamless, engaging, and accessible online multiplayer gaming experience that effectively addresses the challenges and requirements outlined in section 3.1.

3.2.1 Unity Game Engine

Unity, as discussed in Chapter 2 (section 2.4.1), is a widely adopted cross-platform game engine that offers an extensive set of tools and features for game development. Its user-friendliness, versatility, and support for both 2D and 3D games across multiple platforms make it an ideal choice for the Tank Tactics project.

One of the key advantages of using Unity for Tank Tactics is its component-based architecture and modular approach to game object composition. This design philosophy simplifies the development process and promotes code reusability by allowing developers to create and modify individual components without affecting the entire game object. This modular approach enables the Tank Tactics project to be efficiently created and iterate on gameplay mechanics, visual assets, and multiplayer functionality, streamlining the development workflow and reducing the time required to implement new features or make changes to existing ones.

The Unity editor's intuitive interface and customisable layout further enhance the development experience. The editor provides a wide range of built-in tools for scene editing, asset management, and debugging, allowing developers to work more efficiently and effectively. For example, the scene view enables developers to visually design and manipulate game objects, while the hierarchy window provides a clear overview of the game object structure. The inspector window allows developers to modify the properties and components of selected game objects, and the project window serves as a central hub for managing and organizing game assets.

In addition to these core features, Unity offers a vast ecosystem of plugins, extensions, and asset packages through the Unity Asset Store. These resources can significantly accelerate the development process by providing pre-built solutions, art assets, and tools that can be easily integrated into the project. Tank Tactics project can leverage these assets to enhance the game's visual quality, implement complex gameplay mechanics, or extend the functionality of the Unity editor to suit the project's specific needs.

By leveraging the power and flexibility of the Unity game engine, the Tank Tactics project can benefit from a streamlined development process, a modular and reusable codebase, and a rich ecosystem of tools and assets. These advantages ultimately contribute to the creation of a polished, engaging, and accessible multiplayer gaming experience.

3.2.2 Visual Studio and C#

Unity's primary scripting language is C#, a powerful and versatile object-oriented programming language that is well-suited for game development. To write and debug C# scripts efficiently, Visual Studio, a feature-rich integrated development environment (IDE), is the recommended choice for Unity projects.

Visual Studio offers a wide range of productivity-enhancing features that streamline the coding process and help developers write clean, efficient, and maintainable code. One of the most notable features is IntelliSense, an intelligent code completion system that provides real-time suggestions and auto-completion for variables, methods, and classes as developers type. This feature significantly reduces the time required to write code and minimizes the risk of syntax errors, allowing developers to focus on implementing gameplay logic and mechanics.

In addition to IntelliSense, Visual Studio provides real-time error detection and highlighting, which helps developers identify and resolve issues quickly. The IDE continuously analyzes the code as it is written, underlining potential errors and offering suggestions for fixes. This feature is particularly valuable in a complex project like Tank Tactics, where maintaining code quality and minimizing bugs is crucial for a smooth and enjoyable multiplayer experience.

Visual Studio also offers powerful debugging tools that allow developers to investigate and resolve issues in their code. The debugger enables developers to pause the execution of the game at specific points, inspect variable values, and step through the code line by line to identify the source of problems. The IDE also provides advanced debugging features, such as conditional breakpoints and the ability to modify variable values during runtime, which can be invaluable when troubleshooting complex gameplay mechanics or multiplayer interactions.

C#'s object-oriented nature and extensive class library make it an ideal language for implementing intricate gameplay mechanics, and multiplayer network logic. The language supports essential object-oriented programming concepts, such as encapsulation, inheritance, and polymorphism, which allow developers to create modular, reusable, and maintainable code structures. By leveraging these features, the Tank Tactics project can be made with a codebase that is easy to understand, modify, and extend throughout the development process.

Furthermore, C# offers a wide range of built-in classes and libraries that simplify common programming tasks, such as string manipulation, file I/O, and collections. The language also provides support for advanced features like LINQ (Language Integrated Query), which enables developers to write expressive and concise queries for data manipulation and filtering. These features can significantly reduce the amount of boilerplate code required and allow developers to focus on implementing game-specific logic.

By utilizing C# and Visual Studio, the Tank Tactics project can benefit from a robust and efficient development environment that promotes code quality, maintainability, and extensibility. The combination of a powerful programming language and a feature-rich IDE empowers developers to create engaging gameplay mechanics, optimize performance, and ensure a smooth and enjoyable multiplayer experience for players.

3.2.3 Unity Netcode and Unity Gaming Service

Tank Tactics makes use of Unity's Netcode for GameObjects (NGO) framework and Unity Gaming Services (UGS) Relay & Lobby to solve the multiplayer connectivity issues mentioned in Chapter 1 (section 1.2) and Chapter 2 (section 2.2).

Unity Netcode for GameObjects (NGO) is a high-level networking library that abstracts the complexities of low-level network communication, allowing developers to focus on creating gameplay logic and synchronizing objects across the network. NGO is built on top of the Unity Transport Package, a low-level networking library that manages connections using UDP (User Datagram Protocol) and handles reliable and unreliable message sending.

One of the key advantages of using NGO in Tank Tactics is its simplicity and ease of use. The framework provides a set of components and APIs that seamlessly integrate with Unity's existing GameObject and component system, making it easier for developers to add multiplayer functionality to their games. For example, the NetworkManager component acts as the core orchestrator for multiplayer sessions, handling tasks such as server startup, client connection, and scene management. The NetworkBehaviour component, on the other hand, enables developers to define which game objects and properties should be synchronized across the network.

NGO also provides a range of built-in features that simplify common multiplayer tasks. These include object spawning, state synchronization, and remote procedure calls (RPCs). Object spawning allows the server to create game objects on all connected clients, ensuring that all players see the same objects in the game world. State synchronization keeps the game state consistent across all clients by automatically sending updates for synchronized variables whenever their values change. RPCs enable developers to define methods that can be invoked remotely by other clients or the server, facilitating communication and interaction between players.

To further enhance the multiplayer experience and address the challenges associated with manual port forwarding and IP address sharing, Tank Tactics integrates Unity Gaming Services (UGS) Relay & Lobby. UGS Relay is a managed service that provides a secure and scalable infrastructure for hosting multiplayer game sessions. By leveraging the Relay service, Tank Tactics can establish connections between players without requiring them to configure their routers or firewalls, eliminating the need for manual port forwarding and ensuring a seamless and accessible multiplayer experience.

UGS Lobby, on the other hand, offers a convenient way for players to find and join multiplayer game sessions. The Lobby service allows developers to create and manage game lobbies programmatically, enabling players to browse available sessions, join existing lobbies, or create new ones. This feature streamlines the matchmaking process and provides a centralized hub for players to connect and interact with each other before starting a game.

The combination of Unity Netcode for GameObjects and Unity Gaming Services Relay & Lobby enables Tank Tactics to implement a client-hosted listen server model, as described in Chapter 2 (section 2.3.5). In this hybrid network topology, one of the players acts as both a client and a server, hosting the game for other players who connect as clients. This approach offers several benefits, such as reduced hosting costs and improved accessibility, while still maintaining the advantages of a client-server architecture, such as better security and simplified game state management. By utilising the power of Unity Netcode and Unity Gaming Services, Tank Tactics can deliver a smooth, responsive, and seamless multiplayer experience that effectively addresses the challenges associated with traditional multiplayer game development. These technologies align with the project's objectives, as outlined in Chapter 1 (section 1.3), and provide a robust foundation for implementing the game's core multiplayer features and mechanics.

3.3 Implementations

This section provides a comprehensive overview of the implementation of Tank Tactics, covering various aspects such as game design, project setup, core gameplay mechanics, multiplayer architecture, and UI/game management.

3.3.1 Game Design

Tank Tactics is designed as an engaging, combat multiplayer tank battle game. The core gameplay resolves around players controlling tanks, collecting coins, and using the designated healing zones around the map, killing other players to collect bounty coins that are dropped. The game incorporates various other features to enhance the overall experience, including a real-time leaderboard, and a mini-map.

The game design aims to provide an enjoyable experience for up to 20 players simultaneously. It draws inspiration from popular .io style games and classic tank-based titles.

3.3.2 Project Setup

Tank Tactics is developed using the Unity game engine, taking advantage of its extensive features and tools for game development. The project is structured in a modular and organized manner, with various scripts, prefabs, and assets carefully categorized based on their functionality. This approach ensures maintainability, ease of debugging, and flexibility for future expansion.

The scripts are categorized into different functionalities, such as core player mechanics, combat systems, coin management, networking, and UI/game management. This modular approach allows for easy maintenance, debugging, and expansion of the game's features.

3.3.3 Core Gameplay Mechanics

The core gameplay mechanics of Tank Tactics are implemented through various scripts that handle different aspects of player functionality, combat, coin management, and game events. The tank movement and aiming mechanisms are designed to be intuitive and responsive, allowing players to navigate the game world and engage in combat effectively. The shooting mechanics are fine-tuned to ensure a satisfying and balanced gameplay experience.

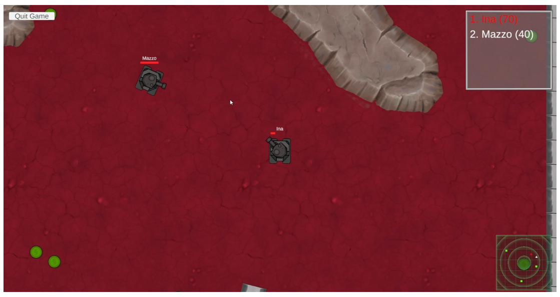

Figure 3.1: Gameplay

The game features a coin collection system, where players can gather coins scattered throughout the map. These coins serve as a form of in-game ammo and currency, enabling players to utilise the designated healing zones and enhance their gameplay. The coin management system keeps track of each player's coin balance and handles the spawning and collection of coins seamlessly.

Combat mechanics, such as damage calculation and health management, are implemented to create a challenging and rewarding experience. Players can engage in battles with other tanks, with the outcome determined by factors such as weapon accuracy, damage output, and strategic positioning. Healing zones and player respawning mechanics add an extra layer of strategy and ensure fair gameplay.

3.3.4 Network Architecture

Tank Tactics employs a client-hosted listen server model, which combines elements of client-server and peer-to-peer architectures. This approach allows for efficient network communication and synchronization between players while minimizing the need for dedicated server infrastructure.

The game leverages Unity's Netcode for GameObjects (NGO) framework to handle the underlying network communication and synchronization. NGO simplifies the process of implementing multiplayer functionality by providing a high-level API for managing network objects, synchronizing game states, and handling client-server interactions.

To enhance the multiplayer experience further, Tank Tactics integrates Unity Gaming Services (UGS). UGS provides a suite of tools and services for features such as player authentication, matchmaking, and lobby management. By leveraging UGS, the game ensures a seamless and user-friendly multiplayer experience, allowing players to easily connect, join games, and interact with each other.

The multiplayer architecture is designed to handle various aspects of network communication efficiently, such as object spawning and despawning, position synchronization, and event propagation. The implementation ensures a smooth and responsive gameplay experience, minimizing the impact of network latency and optimizing bandwidth usage.

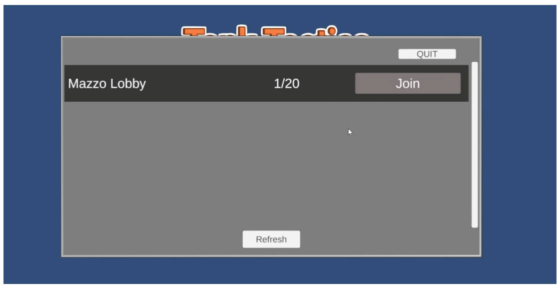

Figure 3.2: Player Lobby

3.3.5 UI and Game Management

The user interface (UI) and game management aspects of Tank Tactics are implemented to provide players with a polished and intuitive experience. The UI is designed to be clean, informative, and easy to navigate, allowing players to access essential information and perform actions effortlessly.

Players can browse available lobbies, create their own lobbies, and join existing ones. The lobby management system handles player connections, disconnections, and game session initialization.

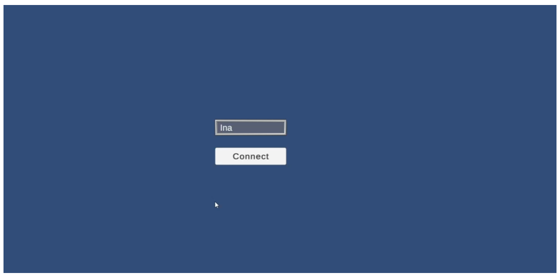

Figure 3.3: Player Connection

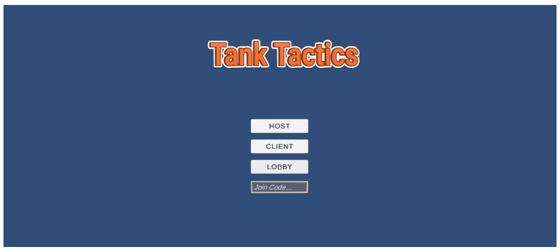

Figure 3.4: Menu Screen

Game management scripts handle various aspects of the game's flow and progression. This includes managing game states, triggering events, and handling player actions. The implementation ensures a consistent and synchronized experience for all players, regardless of their role as a host or client.

Overall, the UI and game management components of Tank Tactics are designed to enhance the player experience, providing a seamless and intuitive interface for navigating the game world, managing multiplayer sessions, and interacting with other players.

3.4 Summary

In summary, the implementation of Tank Tactics showcases a modular and well-structured approach to game development. The game design prioritizes player engagement, and competition, while the project setup ensures maintainability and extensibility. The core gameplay mechanics are implemented to provide a dynamic and rewarding experience, with smooth tank controls, strategic coin collection, and intense combat. The multiplayer architecture leverages

Unity's NGO framework and UGS to enable efficient network communication and a seamless multiplayer experience. The UI and game management aspects are designed to be intuitive, informative, and user-friendly, enhancing the overall player experience. By combining these elements, Tank Tactics delivers an immersive and exciting multiplayer tank battle game that encourages, strategy, and skill.

Chapter 4

Results

4.1 Overview

This chapter presents the outcomes and results achieved through the development and implementation of Tank Tactics, the online multiplayer tank battle game. It showcases the effectiveness of the solution approach and the impact of the features and gameplay mechanics incorporated into the game. Additionally, this chapter leverages the insights gained from two surveys conducted at different stages of the project, providing valuable feedback and data to evaluate the game's performance and player experience.

4.2 Multiplayer Connectivity and Accessibility

One of the primary objectives of Tank Tactics, as outlined in Chapter 1 (section 1.3), was to implement seamless multiplayer connectivity and eliminate the need for manual port forwarding and IP address sharing, which often hinder accessibility in traditional online multiplayer games. The successful integration of Unity's Netcode for GameObjects (NGO) framework and Unity Gaming Services (UGS) Relay & Lobby service has effectively addressed this challenge.

Players can now join and engage in exciting tank battles without the need for complex network configurations or sharing personal information. The game provides a streamlined and user-friendly experience, allowing players to easily connect and participate in multiplayer sessions, as envisioned in Chapter 2 (section 2.2.6).

4.3 Core Gameplay Mechanics and Features

The core gameplay mechanics of Tank Tactics, including intuitive tank controls, precise shooting mechanics, and a dynamic coin system, were meticulously fine-tuned to provide an engaging and rewarding experience for players. The implementation of these mechanics, as described in Chapter 3 (section 3.3.3), has successfully delivered the project's objectives of creating an enjoyable multiplayer tank battle game.

Furthermore, the incorporation of advanced features such as the real-time leaderboard, minimap, healing zones, and bounty coins has added depth and variety to the gameplay experience. These features, as outlined in Chapter 1 (section 1.1.3), have encouraged players to strategies and compete, fostering a sense of competitiveness and immersion.

4.4 Survey Results and Player Feedback

To assess the effectiveness of Tank Tactics and gather valuable feedback, two surveys were conducted at different stages of the project's development. The results of these surveys provide insights into the game's performance, player experience, and areas for improvement.

4.5 Initial Survey Results

The first survey was carried out during the early stages of the project's development. The results revealed a significant number of participants encountered bugs and did not find the game enjoyable. Specifically, 71% of the participants reported encountering bugs, while 57.1% indicated that the game was not fun.

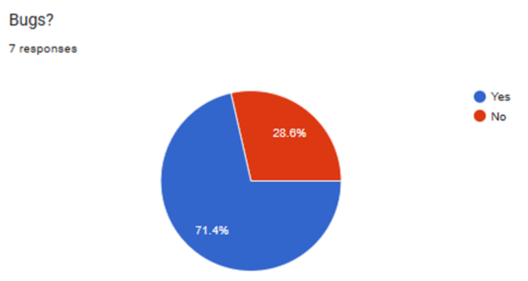

Figure 4.1: Survey for Bugs

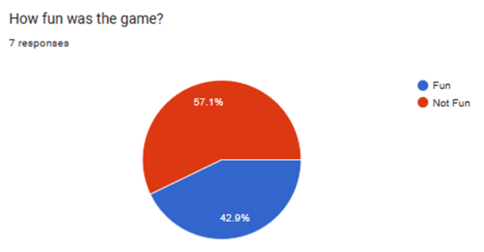

Figure 4.2: Players Experience

These initial survey results highlighted the need for immediate attention and improvement in various aspects of the game, including bug fixing, gameplay mechanics refinement, and overall user experience enhancement.

4.6 Post-Improvement Survey Results

After implementing significant improvements and refinements based on the feedback from the initial survey, a second survey was conducted to assess the impact of the changes. The results were remarkably positive, validating the effectiveness of the development efforts.

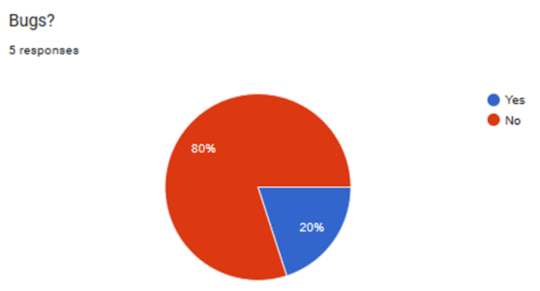

Figure 4.3: Survey for Bugs

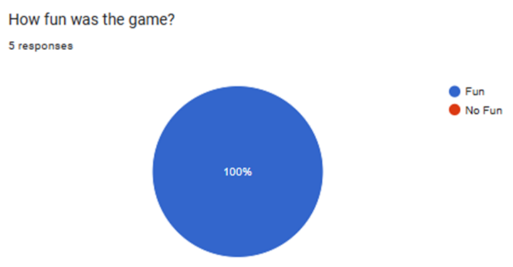

Figure 4.4: Players Experience

An astounding 100% of the participants found the game to be fun and engaging, a testament to the successful enhancements made to the gameplay mechanics and overall experience. Moreover, 80% of the participants reported a significant reduction in bugs, indicating that the extensive testing and optimisation efforts had paid off.

These survey results demonstrate the project's ability to address and overcome the initial challenges, ultimately delivering an engaging and accessible online multiplayer tank battle game.

4.7 Summary

This chapter presented the results achieved through the development and implementation of Tank Tactics, showcasing the successful integration of seamless multiplayer connectivity, engaging gameplay mechanics, and advanced features. The survey results and player feedback provided valuable insights into the game's performance and user experience, highlighting areas for improvement and validating the effectiveness of the implemented enhancements.

The positive outcomes and survey results demonstrate the project's success in achieving its primary objectives, as outlined in Chapter 1, and addressing the challenges associated with traditional online multiplayer games, as discussed in Chapter 2.

Chapter 5

Discussion and Analysis

5.1 Summary

This chapter provides an in-depth discussion and analysis of the results obtained from the development and implementation of Tank Tactics, the online multiplayer tank battle game. It evaluates the significance of the findings, highlights the key achievements, and examines the limitations encountered during the project. The chapter aims to enhance the reader's understanding of the project's outcomes and their implications within the context of the objectives outlined in Chapter 1.

5.2 Significance of the Findings

The successful implementation of Unity's Netcode for GameObjects (NGO) framework and Unity Gaming Services (UGS) Relay & Lobby in Tank Tactics represents a significant achievement in addressing the challenges associated with traditional online multiplayer games. By eliminating the need for manual port forwarding and IP address sharing, as discussed in Chapter 2 (section 2.2.2), the game provides a seamless and accessible multiplayer experience for players.

The engaging and dynamic gameplay mechanics, including intuitive tank controls, strategic coin collection, and intense combat, contribute to an immersive and rewarding experience for players. The incorporation of advanced features such as the real-time leaderboard, minimap, healing zones, and bounty coins adds depth and variety to the gameplay, encouraging players to strategies and compete, as envisioned in Chapter 1 (section 1.1.3).

The modular and well-structured approach to development, as described in Chapter 3 (section 3.3.2), ensured the maintainability and extensibility of the codebase. The separation of concerns through categorized scripts and prefabs facilitated easy debugging and future expansion of the game's features, contributing to the project's long-term sustainability.

5.3 Evaluation of the Solution Approach

The client-hosted listen server model employed in Tank Tactics, as discussed in Chapter 2 (section 2.3.5), proved to be an effective solution for the development of a smaller-scale multiplayer game. This hybrid approach combined aspects of client-server and peer-to-peer architectures, providing improved security and simplified game state management compared to a pure peer-to-peer topology, while minimizing the costs and complexities associated with dedicated server infrastructure.

The integration of Unity's NGO and UGS streamlined the development process, allowing the project to focus on gameplay mechanics and player interactions while relying on Unity's robust networking infrastructure. This aligns with the project's objectives outlined in Chapter 1 (section 1.3) and demonstrates the effectiveness of leveraging modern technologies and frameworks to create an engaging and accessible online multiplayer gaming experience.

5.4 Impact of Survey Results and Player Feedback

The results of the two surveys conducted during the project's development, as presented in Chapter 4 (sections 4.4 and 4.5), provided valuable insights into the game's performance, player experience, and areas for improvement. The initial survey results highlighted the presence of bugs and a lack of enjoyment among players, serving as a crucial wake-up call for the development team.

In response to these findings, significant efforts were made to refine the gameplay mechanics, squash bugs, and enhance the overall user experience. The positive outcomes of the post-improvement survey, with 100% of participants finding the game fun and engaging, and a significant reduction in reported bugs, validated the effectiveness of these development efforts.

The survey results played a pivotal role in guiding the project's direction and ensuring that the final product met the expectations and preferences of the target audience. By actively incorporating player feedback and iterating on the game's design and implementation, Tank Tactics was able to deliver an engaging and polished multiplayer experience.

5.5 Summary

This chapter discussed the significance of the findings obtained from the development and implementation of Tank Tactics, highlighting the key achievements and their implications within the context of the project's objectives. It evaluated the effectiveness of the solution approach, particularly the client-hosted listen server model and the integration of Unity's multiplayer technologies.

Furthermore, the chapter analysed the impact of the survey results and player feedback, emphasising their crucial role in guiding the project's direction and ensuring the delivery of an engaging and polished multiplayer experience. By addressing these limitations and implementing the suggested improvements, Tank Tactics could evolve into a more robust, feature-rich, and engaging online multiplayer experience game.

Chapter 6

Conclusions and Future Work

6.1 Conclusions

The Tank Tactics project successfully addressed the challenges associated with traditional online multiplayer games by leveraging Unity's Netcode for GameObjects (NGO) framework and Unity Gaming Services (UGS) Relay & Lobby. The implementation of these technologies eliminated the need for manual port forwarding and IP address sharing, providing players with a seamless and accessible multiplayer experience.

The project met its primary objective of developing an engaging and dynamic online multiplayer tank battle game for up to 20 players simultaneously. The core gameplay mechanics, including intuitive tank controls, strategic coin collection, and intense combat, were carefully designed to create an immersive and rewarding experience for players.

The incorporation of advanced features, such as the real-time leaderboard, mini-map, healing zones, and bounty coins, added depth and variety to the gameplay, encouraging players to strategize and compete. The game's design drew inspiration from popular .io-style games and classic tank-based titles, providing a familiar and intuitive experience for players.

The modular and well-structured approach to development ensured the maintainability and extensibility of the codebase. The separation of concerns through categorized scripts and prefabs facilitated easy debugging and future expansion of the game's features.

Overall, Tank Tactics successfully demonstrated the power and versatility of Unity's multiplayer technologies in creating an accessible and engaging online multiplayer gaming experience. By addressing the common challenges faced by traditional multiplayer games.

6.2 Future Work

While Tank Tactics has achieved success in creating an engaging and accessible online multiplayer tank battle game, there are numerous avenues for future development and improvement.

One exciting direction could be the introduction of multiplayer co-op team battles, allowing players to form squads and strategically collaborate against opposing teams. This game mode could foster teamwork, communication, and coordination among allies, injecting a fresh layer of depth and social interaction into the gameplay experience.

The implementation of private lobbies would cater to players seeking more controlled and exclusive multiplayer sessions. This feature could enable friends, clans, or competitive groups to create private matches, fostering a tight-knit community and facilitating organized tournaments or events within the Tank Tactics ecosystem.

Expanding the game's customization options through a tank customization system could prove highly appealing to players. The ability to personalize the appearance, colour schemes, and potentially even performance attributes of their tanks could heighten the sense of ownership and individuality, allowing players to express their unique styles while engaging in battles.

Furthermore, the introduction of custom power-ups could dramatically alter the strategic landscape of Tank Tactics. Imagine players acquiring temporary abilities such as enhanced speed boosts, defensive shields, or even special weaponry. This mechanic could inject an element of unpredictability and dynamism into matches, forcing players to adapt their tactics on the fly and adding another layer of depth to the gameplay experience.

The potential integration of dedicated server infrastructure could address scalability and performance concerns as the game's popularity grows. Dedicated servers would provide a robust architecture capable of supporting larger concurrent player counts while maintaining a smooth and consistent gameplay experience.

Finally, performance optimization through code profiling and refinements should remain an ongoing priority, ensuring a consistently smooth and responsive experience as new features are introduced and the player population expands.

Chapter 7

Reflection

The development of Tank Tactics has been a critical learning experience, providing me with the opportunity to apply and expand my technical skills while also enhancing my problem-solving and decision-making abilities. Throughout the project, I developed a deeper understanding of game development, multiplayer networking, and engaging gameplay design.

One of the most significant areas of growth was my proficiency with the Unity game engine and its multiplayer technologies. I learned how to leverage Unity's Netcode for GameObjects (NGO) framework and Unity Gaming Services (UGS) Relay & Lobby to implement seamless online multiplayer functionality. Adopting and integrating these frameworks into the project allowed me to expand my knowledge of networking principles and real-time multiplayer interactions.

Designing engaging gameplay mechanics and features for Tank Tactics was both challenging and rewarding. Creating intuitive tank controls, precise shooting mechanics, and a dynamic coin system required a deep understanding of game design principles and player experience. While developing these core mechanics was initially complex, the process of iterating and refining them to create an enjoyable experience was immensely satisfying.

One of the major challenges I faced was the implementation of efficient networking solutions to ensure smooth and responsive multiplayer gameplay for up to 20 concurrent players. Ensuring real-time interactions without significant lag or delay was a complex task that required me to expand my knowledge of networking principles and optimization techniques. While I was able to implement a solution that met the project's requirements, I believe there is still room for improvement in this area, particularly in terms of scalability and performance for larger player counts.

If I were to approach a similar problem in the future, I would allocate more time to the initial planning and design phase. A more detailed plan and thorough architecture design could have helped mitigate some of the challenges I encountered during the development process, especially with client-server communication and synchronization. Reflecting on the project's aims and objectives outlined in Chapter 1, while Tank Tactics successfully met the core requirements of delivering an engaging online multiplayer tank battle game, there were some deviations from the initial plan. For instance, the complexity and scope of certain advanced features, such as leaderboards and mini-maps, were simplified over time as I gained a better understanding of the intricacies involved in their implementation.

These changes, though not initially planned, ultimately contributed to the success of the game by allowing me to focus on delivering a polished and stable core experience. The lessons learned from navigating these challenges and making informed decisions will undoubtedly be valuable in my future game development endeavours. Overall, the Tank Tactics project had a significant impact on my learning experience and future work. It not only allowed me to apply and expand my technical skills in game development and multiplayer networking but also helped me develop crucial problem-solving, decision-making, and project management abilities. The knowledge and expertise gained from this project will serve as a solid foundation for my continued growth and success in the field of game development.

References

Armitage, G., Claypool, M. and Branch, P. (2006), *Networking and online games: understanding and engineering multiplayer Internet games*, John Wiley & Sons.

Bartle, R. (1990), 'Early mud history'.

Bartle, R. A. (2004), *Designing virtual worlds*, New Riders.

Bettner, P. and Terrano, M. (2001), 1500 archers on a 28.8: Network programming in age of empires and beyond, *in* 'GDC', Vol. 2, p. 30.

Dealessandri, M. (2020), 'What is the best game engine: is unity right for you?'.

Dongen, J. V. (2015), 'Why adding multiplayer makes game coding twice as much work', *joostdevblog* .

Donovan, T. and Garriott, R. (2010), 'Replay: The history of video games', *(No Title)* .

Frohnmayer, M. and Gift, T. (2000), The tribes engine networking model, *in* 'Proceedings of the Game Developers Conference'.

GameObjects (n.d.).
URL: *https://docs.unity3d.com/Manual/GameObjects.html*

Glazer, J. and Madhav, S. (2015), *Multiplayer game programming: Architecting networked games*, Addison-Wesley Professional.

Goguen, N. (2023), 'Port forwarding explained: The ultimate guide to how it works and why you need it', *No-IP Blog* .

Kushner, D. (2004), *Masters of Doom: How Two Guys Created an empire and Transformed Pop Culture*, Random House Trade Paperbacks.

Lemay, S. (2022), 'Ngo limit'.
URL: *https://forum.unity.com/threads/what-makes-ngo-limited-to-small-scale-games-only.1338971/post-8461043*

Monkey, C. (2023*a*), *Unity Lobby*.
URL: *https://www.youtube.com/watch?v=-KDlEBfCBiU*

Monkey, C. (2023*b*), *Unity Relay*.
URL: *https://www.youtube.com/watch?v=msPNJ2cxWfw*

Netcode (n.d.).
URL: *https://docs-multiplayer.unity3d.com/netcode/current/about/*

ogerman (2021), ‘Technological advancement shapes the future of online gaming’, *GameOgre* .

Parameswaran, M. and Whinston, A. B. (2007), ‘Research issues in social computing’, *Journal of the Association for Information Systems* **8**(6), 22.

Smed, J., Kaukoranta, T. and Hakonen, H. (2002), ‘Aspects of networking in multiplayer computer games’, *The Electronic Library* **20**(2), 87–97.

The Future of Online Gaming: Innovations and Challenges (2023), *EBR* .

Tikhomirov, A. (2023), ‘Developing an online multiplayer game in unity’.

Unity Scripts (n.d.).
URL: *https://docs.unity3d.com/Manual/CreatingAndUsingScripts.html*

Unity Transport (n.d.).
URL: *https://docs-multiplayer.unity3d.com/transport/current/about/*

Unity UGS (n.d.).
URL: *https://docs.unity.com/ugs/manual/overview/manual/unity-gaming-services-home*

Unity’s Interface (n.d.).
URL: *https://docs.unity3d.com/Manual/UsingTheEditor.html*