Development of SudoDuel: A Competitive Online Multiplayer Sudoku Game

Thomas Tudor

4th Year Project Report
Computer Science
School of Informatics
University of Edinburgh

2024

Abstract

The popularity of Sudoku has endured over time, with the game evolving to include digital versions, themed variations, and online multiplayer modes. This project focuses on the development process of SudoDuel, a competitive online multiplayer Sudoku game. Key considerations in designing and implementing features for such a game are investigated and discussed. Multiplayer functionality is crucial, allowing players to compete or collaborate with others in real-time. To make sure the playing experience remains smooth, it is crucial that the game can handle high volumes of players and has good scalability.

When developing a game, creating an interface that is easy for users is also necessary. This includes things like different levels of difficulty and notes aimed at improving the actual playing experience. We also need to deal with issues in development, such as synchronising events or managing big numbers of players at once. The project aims to offer an understanding of the technical and design choices made for making SudoDuel. This interactive online Sudoku game allows players to compete against each other.

Research Ethics Approval

Instructions: This project was planned in accordance with the Informatics Research Ethics policy. It did not involve any aspects that required approval from the Informatics Research Ethics committee.

Declaration

I declare that this thesis was composed by myself, that the work contained herein is my own except where explicitly stated otherwise in the text, and that this work has not been submitted for any other degree or professional qualification except as specified.

(*Thomas Tudor*)

Acknowledgements

I would like to express gratitude towards my supervisor, Professor Nigel Topham, for his assistance throughout the duration of the project.

I would also like to thank my friends and family for supporting me during the writing of this paper.

Lastly, I extend my gratitude to the dedicated community and developers behind the Godot Engine, whose efforts and commitment to open-source game development have made this project possible.

Table of Contents

Chapter 1

Introduction

The evolution of game development, particularly within the realm of online multiplayer platforms, presents an intricate blend of technical prowess, creative vision, and user experience design. This field has expanded dramatically due to the progress in digital technology, creating an environment for game developers to make exciting and interactive experiences that bring together players worldwide. A competitive Sudoku game for playing online against people across the globe, created using the Godot engine's robust features, is a clear sign of this advancement. This project aims to break down the complex process involved in making a game that is accessible but also challenging and goes beyond typical Sudoku by including competitive multiplayer aspects.

This paper delves into the comprehensive process behind designing and developing a multiplayer Sudoku game. It goes into detail about the important choices in gameplay mechanics, network structure, user interface design, and compatibility between platforms. It gives extra attention to difficulties and answers found in transforming a puzzle game typically played alone into an interactive online multiplayer experience. The conversation also includes the decision to use Godot as a development framework, explaining its ability to handle the complicated networking needs of the game and make the development process smooth.

The culmination of this project is a fully functional online multiplayer Sudoku game compatible with Windows and Linux operating systems. It has a mode of quickplay that lets players easily join online games, giving a smooth blending of competitive playing inside the digital world. This version adds extra excitement to the regular Sudoku play. Now, you can use abilities that affect the state of play and have the potential to slow down your rival's advancement. This brings new strategies and involvement that are not seen in usual Sudoku games. Using these abilities together with matches against opponents happening at that moment changes the solo puzzle experience into an active and exciting mental fight where every game is about more than just solving puzzles but also beating your competitor. This fresh method of Sudoku emphasises the game's unique role in digital gaming, providing a new and thrilling competition for those who love Sudoku and competitive players too.

The following chapters of this paper analyse the project's development cycle. Chapter

2 investigates the origins of Sudoku, game design principles, and similar applications. This background information sets the stage for understanding the game's design rationale. Chapter 3 talks about the design process, which involves gameplay mechanics, multiplayer framework, and user interface. This chapter gives an understanding of the gameplay decisions and technical thoughts that affected how the game was made. Chapter 4 covers the technical implementation, including Godot's High-Level Multiplayer API, server management on DigitalOcean Droplets, and cross-platform compatibility. Chapter 5 evaluates the game's performance, limitations, and whether it met the requirements mentioned in Chapter 3. It provides beneficial information for possible game improvements later on.

Chapter 2

Background

2.1 The Sudoku Problem

2.1.1 Outlining The Problem

The Sudoku puzzle is a specific variant of the mathematical problem known as Latin Square theorised by Euler [6]. The modern problem started appearing in 1979 in newspapers. The puzzles at the time were called Number Squares. They only started gaining popularity in 1986 when Japanese puzzle company Nikoli published the same style of problem under the name of Sudoku, meaning Single Digit or Single Number.

The total number of valid Sudoku grids is 6,670,903,752,021,072,936,960 [8]. This number has been reduced to 5472730538, essentially different grids if you account for transformations of existing grids using some symmetry. [24]

When it comes to ranking the difficulty of a Sudoku problem there are a few criteria that determine it. The most obvious one is the number of clues in the grid. Logic assumes that the more clues available, the easier the given problem. Researchers Maji et al. [19] used Table 2.1 to compare difficulty to the number of clues provided, scaling from Extremely Easy to Evil. Figure 2.1 is an example of an easy Sudoku problem from Sudoku.com [4].

Take notice of the fact that evil difficulty Sudoku Problems have a minimum of 17 clues. The problem of minimum Sudoku problem was first proven in 2010 by Lin & Wu [18] using Artificial Intelligence. This was elaborated and confirmed by McGuire [20] in 2012.

Table 2.1: Number of clues for each difficulty level

Difficulty Level	# of Clues
1 (Extremely Easy)	> 46
2 (Easy)	$36 - 46$
3 (Medium)	$32 - 35$
4 (Difficult)	$28 - 31$
5 (Evil)	$17 - 27$

8				2		9	1	
2	3	4	5	1				7
7	1			8			5	4
6			1			3		5
1	8	5				7	2	
	4		6		2	8		
	6	8				4		
						1	6	2
			4		7	5	3	

Figure 2.1: Easy Sudoku Problem

2.1.2 Solving Sudoku Grids

Sudoku puzzles have been extensively researched. To the masses of readers of the newspapers, people often use popular techniques such as ”X-Wing” or ”Swordfish”[15] to solve complex grids. These are examples of propagation techniques that can be used in a constraint problem.

The general problem of $n^2 \times n^2$ boards of $n \times n$ blocks, however, has been proven to be NP-Complete by Yato and Seta in 2003[34]. Because of this, specific algorithms can be used to solve a Sudoku grid. The most straightforward algorithm to implement is a brute-force method known as backtracking.

Backtracking can be described as, for a given problem, exploring all the potential nodes of a particular branch and, if none are valid, going up one node, making a new branch, and exploring all the potential nodes there. This depth-first search strategy is guaranteed to find all solutions, given enough time, to a Sudoku puzzle given the finite number of valid grids. For a Sudoku problem, the algorithm will go through each cell and assign it a number from 1-9; then, it will check that the grid is valid at that point. If it is, it will proceed to the next cell; if it is not, it will try the next digit until it is either invalid for all numbers or proceeds. If the algorithm finds that none of the numbers 1-9 fit that particular cell, it will go to the previous cell and try the next digit. Eventually, if a solution is found, the algorithm will terminate. Otherwise, it will try every single possible solution in the search space. Figure 2.2 shows a simplified visual representation made by Pelánek of the backtracking algorithm.[21]

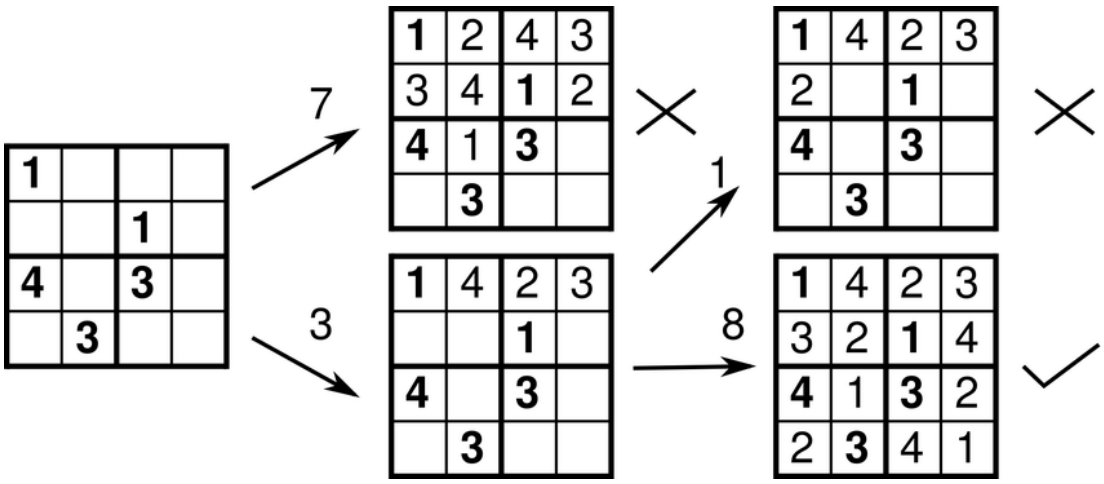

Figure 2.2: Simplified Representation of Sudoku Backtracking for a 4x4 grid. Numbers over arrows indicate number of steps without branching

This algorithm can be improved upon as, at the moment, the algorithm has a time complexity of $O(9^n)$ where n is the number of empty spaces. The most noticeable improvements are cell ordering and value elimination. Cell ordering is sorting the available spaces by their constraints and using the most constrained cells first. Value elimination is an improvement that removes any potential values for a given cell that would be invalid. Alternatively, values could be ordered to use the most common or used numbers first to speed up the backtracking process.

2.2 Game Design & Development

2.2.1 Game Engines

This section will outline the different choices of game engines I had access to in this project and each of their pros and cons with developing them.

2.2.1.1 Unity

The Unity Game Engine[30], made by Unity Technologies, is a widely used 2D & 3D game engine and IDE. The engine can produce applications for a wide range of platforms. In terms of puzzle games made on the platform, some notable names include Monument Valley[10], The Room Series[10] & Lara Croft Go[14].

In terms of features, Unity has a visual interface that allows developers to create and edit game scenes, manage assets, and tweak settings. It provides a WYSIWYG (What You See Is What You Get) environment where game elements can be dragged, dropped, and arranged as the developer sees fit.

Game scripts are written in Microsoft's C# programming language. Unity has its own "GameObject" class, where several different components can be assigned to the object to define its behaviour or actions.[31] These components include but are not limited

to rendering, physics, sound, and scripting. Scripting components are created using Unity's "MonoBehaviour" class[32] and allow for many different object behaviours.

2.2.1.2 Unreal Engine

The Unreal Engine, developed by Epic Games, is a highly regarded game engine and development environment. The engine boasts capabilities to craft games for a diverse array of platforms. Regarding puzzle games crafted using this engine, notable mentions are The Witness, Q.U.B.E. 2, and Tetris Effect.

As for its functionalities, Unreal Engine provides a visual workspace known as the Blueprint system that offers developers an intuitive way to design gameplay mechanics and events without the immediate need for code. It embodies a node-based interface where game logic can be visually pieced together, offering a direct insight into how events will unfold in the game.

If not done through Blueprints, gameplay scripts and logic are primarily crafted using C++ in Unreal, ensuring robustness and flexibility in the development process. The engine has a core "Actor" class, which is the foundation for all interactive objects. Similarly to Unity, various components, like mesh rendering, physics simulations, audio, and AI behaviours, can be attached to these actors. Using its "UObject" class and associated subclasses, developers can design intricate and diverse gameplay elements tailored to their vision.

If the Unreal Engine were to be used to make the Sudoku application, Unreal would feature a dedicated toolset for 2D games called Paper2D. It offers sprites, flip-books for animations, tile maps for grid-based level design, and specialised 2D physics. Integrated with Unreal's powerful Blueprint visual scripting system, Paper2D allows developers to create efficient and visually compelling 2D games within the Unreal environment. While not as popular as some other 2D-focused engines, it provides a robust platform for those seeking the capabilities of Unreal in a 2D space.

2.2.1.3 Godot

The Godot engine is an open-source, cross-platform game development tool designed to provide developers with a powerful and user-friendly platform for creating games. It has been designed to facilitate the creation of both 2D and 3D games across a variety of platforms. When considering its use for puzzle games such as Sudoku, Godot's flexible scene system and easy-to-use script language, GDScript, make it a suitable choice for developers such as myself aiming to create games like online multiplayer Sudoku.

One of Godot's key features is its scene and node architecture, which allows developers to organize game elements in a modular and reusable manner. This system is particularly beneficial for the structured design required in a Sudoku application, where different game components can be managed and interacted with efficiently.

For gameplay logic, as previously mentioned, Godot offers GDScript, a Python-like scripting language designed to be accessible to newcomers while still powerful enough for advanced users. Additionally, the engine supports VisualScript, a node-based

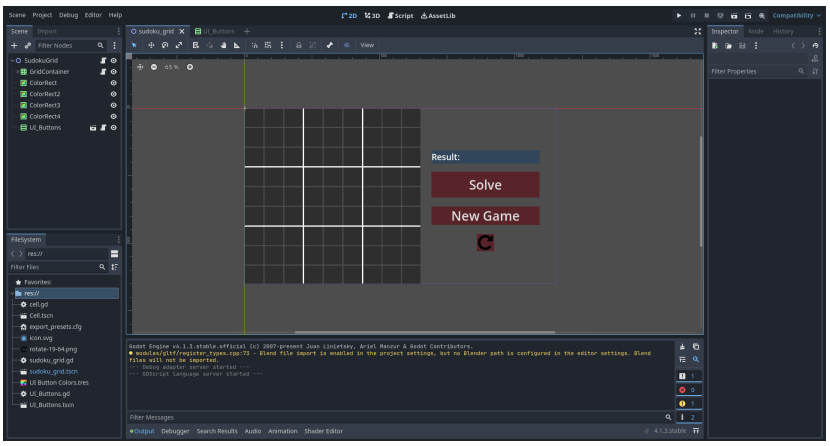

Figure 2.3: The Godot Development Environment

programming language, and C#, providing developers with multiple options to suit their preferences or project requirements.

With online multiplayer capabilities, Godot features a high-level multiplayer API to make the setup of multiplayer sessions more straightforward. In particular, Godot features a dedicated MultiplayerSynchronizer node, which can synchronise different objects between peers simply and intuitively.

Godot does have some limitations. It is a versatile tool for many games, but its performance may lag behind other engines for highly demanding 3D applications and games. Fortunately, this is less of a concern for a Sudoku game, which is not graphically intensive. Another drawback of the engine is that it has a much smaller community than Unity or Unreal. As a result, fewer third-party tools and assets are available to developers, requiring them to invest more time to create custom solutions.

However, despite these drawbacks, Godot's dedication to being open-source and free means there are no financial barriers to entry making it an appealing option for solo developers and small studios. The active community and comprehensive documentation further support developers through the development process.

For a project like an online multiplayer Sudoku game, Godot's strength in cross-platform development, flexible scripting languages, and solid networking capabilities make it a strong candidate. While the engine's limitations in terms of third-party add-ons and a relatively steep learning curve for networking might pose challenges, the comprehensive development environment and active community support made it the most compelling option for this project.

2.2.2 Game Design Elements

When developing any sort of game, the core gameplay loop and its mechanics are critical to get right to achieve an enjoyable game. In this context, a gameplay loop refers to a sequence of actions that players repeat over and over as the core of their interactive experience with a game. For example, in the case of solving a Sudoku puzzle, the gameplay loop consists of identifying empty cells, analyzing existing numbers in

the row, column or square that the cell is in, and filling in the cell with an appropriate number. Fabricatore explains the importance of core gameplay[7] which he defines as the activities the player will undertake most during a game. He also says that the mechanics that enable this core gameplay are the most critical things that should be designed.

Fabricatore also recommends minimizing the number of core mechanics and their complexity whilst also ensuring that these mechanics stay relevant and do not become redundant. Maintaining relevance allows players to gain a sense of mastery over these mechanics by repeating the core activities. Finally, he also suggests including ”satellite mechanics” to add variations and keep the gameplay fresh using variations on existing mechanics.

Adams and Dormans provide a framework for analyzing game mechanics in terms of their effect on the player's physical and mental skills.[1] They categorize core mechanics as those that are available to the player throughout the majority of the game and directly impact progress, aligning with Fabricatore's definition. Designing compelling core gameplay with balanced and engaging mechanics is crucial to a game's success.

In the context of Sudoku and similar puzzle games, Rollings and Adams point out[23] that the core gameplay loop in puzzle games typically revolves around the player attempting to solve a puzzle using logic, deduction, pattern recognition, or trial-and-error experimentation. Schell notes[26] that puzzles themselves are the main mechanic, and rules constraining the player's manipulation of the puzzle elements create the challenge. Sudoku is a well-known puzzle format, and thus, the underlying mechanic is standard; the difficulty in developing a competitive multiplayer variation on Sudoku is to modify the underlying mechanics in such a way that provides engagement and fun without drastically affecting the gameplay loop.

Unlike action games, progression in puzzle games such as Sudoku, is usually discrete and tied to solving each self-contained puzzle. Since the core logical rules of Sudoku tend to remain constant, the progression takes the form of the overall skill of the player when it comes to solving Sudoku boards. As a player does more puzzles, they should be able to complete them faster, as they become better suited at analysing the board and making logical deductions faster.

As Koster explains[16], puzzle games can create variety by changing the visual presentation or specific puzzle configurations between levels. But the underlying abstract mechanics, whether arranging shapes in Tetris or filling a grid in Sudoku, are often tightly constrained and elegant in their simplicity. This allows the player to focus on mentally engaging with the puzzle rather than manipulating complex game objects.

To summarise, the core gameplay in puzzle games like Sudoku revolves around the abstract mechanics of the puzzles themselves. Designers craft the specific puzzle instances and their difficulty to create an engaging solo or competitive experience. The minimal interface and constrained interactions focus on the mental challenge of puzzle-solving using logic and deduction.

2.2.3 Competitive Game Design

Competitive game design involves creating game systems and rulesets that facilitate enjoyable player-versus-player experiences. A key aspect is ensuring a level playing field where skill and strategy determine the winner, rather than luck or unfair advantages[29]. In a multiplayer Sudoku game, this means generating puzzles of equal difficulty for all players or generating the same board. And provide them both with the same interface, controls, and access to any extra or additional features.

Another important factor in competitive games is the depth of strategy and decision-making. Games with a wide possibility space for player choices, risks, and optimization usually provide greater strategic depth according to Burgun[2]. Sudoku's gameplay is inherently deep, requiring players to make logic-based decisions about which numbers to place constantly. Designers can further increase depth through mechanics like power-ups or abilities that can negatively affect an opponent.

In multiplayer Tetris games, clearing multiple lines at once will send ”garbage” rows to the opponent's playfield, which appear at the bottom and push their existing blocks upward[33] Garbage rows usually contain gaps, requiring the opponent to clear them strategically to avoid reaching the top of the playfield. The number of garbage rows sent depends on the line type clear - for example, a Tetris (4 lines) will send four rows of garbage.

Garbage adds an important competitive mechanic to player-versus-player Tetris. Players aim to send as much garbage as possible to their opponents while managing the garbage they receive. This creates an intense back-and-forth dynamic as players try to overwhelm each other with successive attacks. Quick reactions and strategic planning are vital to countering incoming garbage and launching effective counterattacks.

A similar mechanic could be adapted to a competitive multiplayer Sudoku game. Filling in enough cells or completing rows, columns or squares could let the player cause an effect on the opponent's grid that could hinder their performance, such as obscuring their view or deleting their notes. Players would need to adapt their problem-solving strategies to work around these obstacles while finding opportunities to attack their opponent back.

Over time, competitive games often give rise to a ”metagame” - the most popular strategies, tactics, and setups that emerge around a game[25]. Metagames arise as players discover dominant and practical techniques, counter-strategies, and cyclical patterns in high-end play. They make up ”the game outside the game”, which is the ever-evolving body of knowledge and techniques that shape how the game is played at the highest levels of competition as explained by Elias et al.[5]

In a competitive multiplayer Sudoku game, a metagame could emerge around the most efficient solving patterns, logic deduction techniques, or the most advantageous abilities to select and when to use them. For example, the community could develop strategies to identify and execute multi-cell patterns quickly. The player base could also figure out an optimal loadout of abilities to impact their opponent the most. Counter-strategies would likely arise as these strategies become widely adopted, creating an evolving metagame.

Elias et al. also explain that a healthy metagame can deepen the strategic complexity of a game and reward player ingenuity[5]. As players continually seek to outplay established strategies, they push the boundaries of the game and uncover new ways to play. This can extend the lifespan of a competitive game by keeping it dynamic and challenging long after the core mechanics have been mastered.

Metagames can also have downsides. If a metagame becomes too stagnant or if a particular strategy ”solves” the game, with only a narrow range of viable playstyles, it can lead to repetitive matches as discussed by Harper[12]. Overly dominant strategies can make a game feel imbalanced and limit the room for player creativity. In a Sudoku context, if an unbeatable solving pattern or technique is discovered, it could undermine the integrity of the competition.

To ensure a metagame stays fresh, developers can release balance patches, introduce new mechanics, or provide tools for players to customise the game. In a competitive Sudoku game, this could mean adding new abilities for players to use, modifying the way current abilities work, or introducing new win conditions that disrupt established patterns, such as limiting mistakes. By carefully monitoring the metagame and making informed interventions, designers can sustain a dynamic competitive environment.

Anticipating and shaping the metagame is a key consideration in competitive game design. For a multiplayer Sudoku game, the designer should aim to create mechanics that allow for a diverse range of viable playstyles and counter-play. By striking a balance between rewarding skill and leaving room for innovation, they can cultivate an engaging metagame that enhances the depth and longevity of the competitive experience.

2.2.4 Game Development Process

The method of making a video game is called the game development process. This process follows a planned structure, starting from ideas to final release. Game development goes through many stages for designing, building and testing the game so it works well and gives pleasure to players. The various stages of development are discussed below.

2.2.4.1 Pre-Production

Pre-production is the initial phase where the game concept is conceived and refined[26]. Game designers think up ideas, explore themes, and establish core gameplay mechanics. They also carry out market investigations to know about the people they are aiming at with their game, look into rivals' games, and find trends in gaming[3]. The pre-production stage ends with the making of a Game Design Document (GDD). It acts as a detailed plan for the whole project, explaining all aspects like the game's characteristics, narrative, persons, and general design as explained by Rogers[22].

During pre-production, developers also create prototypes to test and refine the core mechanics. It is a method that enables developers to test out multiple ideas and receive input at the beginning stages of development. Fullerton explains the importance of prototyping[9] as it assists in finding design mistakes and improving the play experience. This iterative process helps find out possible problems and guarantees that the game is enjoyable before too much time or resources are invested into complete production

2.2.4.2 Production

Once the pre-production phase is complete, the game enters the production phase of development. This is where the actual development of the game takes place. Assets such as models, textures, and audio are created while programmers implement gameplay mechanics and features.

The production phase is highly iterative; the game goes through many loops of development, testing, and polishing. As a solo developer, I'm expected to fulfill multiple roles including researcher, designer, artist, programmer, and tester. Effective project management during production is necessary to maintain the development process.[3] This involves task allocation, progress tracking, risk management, and ensuring the various parts of the game work cohesively to achieve the project goals. Poor project management can lead to delays, cost overruns, and a compromised final product.

2.2.4.3 Testing and Quality Assurance

To guarantee a stable and bug-free release, thorough testing is essential for the game. QA teams conduct different types of tests, such as functionality testing, compatibility testing, and playtesting, to evaluate the user experience and ensure a balanced game.[27]

Playtesting should produce ideas and feedback for improvement. It should also test the wide variety of platforms and configurations the game is supposed to be built for. The game should support each platform as laid out in the GDD and should be optimized for each platform. User experience research is another crucial aspect of QA, where unbiased feedback is gathered from players to make necessary adjustments to gameplay and features, ensuring that the game delivers the intended experience

2.2.4.4 Release and Post-Launch Support

Last but not least the release phase is when the game development process comes to its peak, and the game becomes accessible for everyone. But, it doesn't mean that work is over. Post-launch support and live operations are very important for keeping up with changes in market conditions and making improvements to the game as time goes on.

Post-launch support is for handling any troubles that show up after the game's launch, like fixing bugs, improving performance, and balancing adjustments. Live operations are about maintaining the game's appeal by introducing fresh content and features for players.

This project focuses primarily on the pre-production and production phases of game development. But, the testing/QA and post-launch phases are also crucial for a game's success. So, they could be examined more in a subsequent project.

2.3 Existing Applications

Sudoku is an extremely popular puzzle and thus several applications allow users to generate and solve Sudoku puzzles in a competitive manner using timers and leaderboards and there are also other multiplayer versions of Sudoku. The version of Sudoku I have

made however is closer to other similar competitive puzzle games which I will detail below. That being said, there is not much literature on the topic of competitive Sudoku however there are some relevant and adjacent applications which are discussed below.

2.3.1 UsDoku

UsDoku is a website that offers a multiplayer Sudoku experience. Players can run games for others to join and then either can compete against each other to see who is faster or help each other solve a board more cooperatively. Games are run in real-time and provide an interactive way to enjoy Sudoku, adding a social element to the traditionally solo puzzle-solving activity.

Players can create either public or private rooms to run games and these games can be of easy, medium, or hard difficulty boards. Public rooms are available via a server browser and private rooms can be joined via a unique 4-letter code that the host can share.

2.3.2 Sudokill

Sudokill introduces an innovative competitive twist to a traditional Sudoku puzzle, turning it into a turn-based dynamic two-player game. It started as a class project in a heuristics course at New York University where students were tasked with making ”player bots” that real users could play against.

The game is hosted online, allowing players to connect from anywhere. To start playing, participants enter their player names on the start page and connect to the game. Once connected, players can either join a game and choose to compete against other online players or select a computer-controlled opponent from a list.

The gameplay involves taking turns to place numbers on the Sudoku board. The first move can be placed in any free square, but subsequent moves must be made in the same row or column as the opponent’s last move, provided there’s an open square available. If no such space exists, the player is free to choose any open square on the board for their next move. The core objective of Sudokill is to not only complete the Sudoku grid correctly but to strategically force your opponent to make an illegal move, one that violates the rules of Sudoku. This competitive edge and the additional rules add a layer of strategy and anticipation, as players must think several steps ahead to outmaneuver their opponents players can intentionally insert incorrect numbers into the grid provided they are legal moves, allowing players to set traps.

2.3.3 Tetr.io

Tetr.io, made by OSK, is a free-to-play online multiplayer and single-player version of the classic game called Tetris. It offers both competitive and casual gameplay options depending on the user’s preference. Players can choose to participate in ranked matches against people around the world through a ”Tetra League” mode, or they can play casually with quick play or custom rooms.

The game features mechanics such as a standard ”change on attack” attacking system, adjustable tuning settings, and a new attack table with a ”multiplier” system. In standard games of multiplayer Tetris, a very common strategy was to leave a four-wide gap in the board, either in the center or at one of the sides. In Tetr.io however, players are rewarded when executing several ”difficult” moves such as T-spins or quad clears. Building up a multiplier means more garbage will be sent to your opponent, meaning that if you risk tricker moves, you will be rewarded for it. With the inclusion of this ”multiplier” system, Tetr.io was able to take an existing game with a somewhat stale metagame and introduce a new mechanic that made the game more interesting in the higher-level competitive scene.

Most of the essential game mechanics and multiplayer aspects of an online Tetris game, like Tetr.io, can be applied to an online Sudoku game with no significant alterations. For instance: the scoring method in Tetris where players gain points for line clears, combos, etc., could be changed in Sudoku to give points for finishing squares, rows, or columns. The online multiplayer feature where Tetris players can compete in real-time could be transferred to enable Sudoku players to race against each other, trying to complete the same puzzle at the same time. Therefore, even though the rules for these games are not the same, we can reuse the basic technical setup and design patterns that make competitive multiplayer possible. This only needs a few changes to adjust for different game mechanics.

Chapter 3

Design

3.1 Gameplay

In this section, we discuss what a normal round of the game that has been developed looks like. The core gameplay of the competitive multiplayer Sudoku game is to combine the traditional puzzle-solving experience with fast-paced and competitive elements to create a continually active gaming experience. Players are tasked with filling a 9x9 grid with digits so that each column, each row, and each of the nine 3x3 subgrids contain all of the digits from 1 to 9, adhering to the classic Sudoku rules. The game is won by the first player to solve their board, or if time runs out, the player with the most points. The application brings extra competitive elements that we will talk about below.

3.1.1 Objective

The Sudoku game's main goal is to involve two players in a 1 versus 1 competition to solve a normal 9x9 Sudoku board. The board is randomly generated with a single solution. For quick games, the grids should be of easier difficulty having between 36-46 clues according to the table used by Maji et al[19]. This should allow players to solve the board in a short amount of time. The inclusion of a planned time limit of four minutes per puzzle helps reinforce the importance of quick thinking and decision-making, turning each session into an exciting competition against both time and your rival.

While the elements of competition can make the game more thrilling, we should always see if an individual is engaged in solving the board as the main goal. The aim is to fill in the entire grid correctly. Success should come from a mix of speed, accuracy, and logical deduction equally. Depending on the strategy or method they use to complete the task at hand (filling in each cell), someone who values quickness shouldn't necessarily outperform someone else with identical experience but different priorities such as focusing on accuracy instead. To keep the game fair and interesting each player is given the same randomly generated solvable board with the same available clues.

3.1.2 Competitive Twist

To make my game unique compared to similar applications, the game introduces a competitive twist. This is inspired by the lively and head-to-head feel found in other multiplayer puzzle games such as Tetris or Puyo Puyo. The twist brings an element of strategy and surprise which makes the experience more exciting. Players not only need to complete their grids within the time limit but also are engaged in a battle against their opponent where they can use different abilities to get ahead or interfere with other's progress by earning points by solving tiles correctly.

3.1.2.1 Points Acquisition

As a round of the game goes on, players will accumulate points as they correctly fill in tiles. When a cell is filled correctly by a player, it grants them 1 point. If someone fills out the cell without employing any jottings on that specific cell and answers it correct on their first attempt, they get 2 points. This encourages accurate puzzle-solving under the pressure of competition whilst also making it so that players need to use their mind and Sudoku-solving abilities, taking the game to a more advanced level. Finally, one last incentive is that completing a row, column, or square will earn the player 5 additional points.

This system of points is fair because it rewards various aspects of playing. The main point you get when solving a tile is that players are always involved and they know they will be rewarded for their attempts. The extra point received for filling in the tile without any notes creates a balance between risk and reward, allowing players to decide if they want more points but with the possibility of making mistakes and losing all earned points from that specific tile. Lastly, the bonus for finishing a row, column, or square improves this strategic aspect. It motivates players to not only solve separate tiles on their own but also consider the puzzle as a whole.

As an example take the Sudoku grid in Figure 3.1 with 38 clues. A beginner player who uses jottings has the potential to earn 178 points on this grid as there are 43 available tiles and a combined total of 27 rows, columns, and squares available, which is the maximum, totalling 135 points. At the same time, a skilled player who can solve tiles without making jottings would get 221 points because they earn twice as many from the no-jotting bonus.

8				2		9	1	
2	3	4	5	1				7
7	1			8			5	4
6			1			3		5
1	8	5				7	2	
	4		6		2	8		
	6	8				4		
						1	6	2
			4		7	5	3	

Figure 3.1: Easy Sudoku Problem with 38 Clues

To avoid cheating or exploiting, the game confirms if jottings were ever used for the tile. This stops players from simply deleting their jottings before filling in the tile.

3.1.2.2 Abilities

The accrued points can then be spent on abilities to affect the flow of the game. Abilities are mainly employed for two purposes - one is to influence positively on your board while the other is to impact negatively on your opponent's board. Using these abilities strategically may potentially alter a game's outcome significantly at crucial moments, rendering every match thrilling and unpredictable. In an ideal scenario, this could help Sudoku match-ups become less lopsided where less-experienced players have better chances against those who are more skilled or knowledgeable in playing Sudoku.

Depending on the strength of the effect each ability has a different point cost associated with it. The abilities are split into 3 tiers of point cost, with the weakest being 15 points, the middle tier being 30 points, and the strongest being 45 points. For every ability there is also an associated timeout period expressed in seconds that matches its point cost; this prevents abilities from being used repeatedly and enforces a need for strategic use.

Abilities that are to be implemented are outlined below

- Redact - 15 Points - Temporarily blocks a random row or column on the opponent's board, obscuring their view. It hampers their advancement by slowing down work in that area.
- Erase - 45 Points - Removes all notes or jottings from the opponent's board. Since notes are very important for strategizing in Sudoku, removing them can greatly disadvantage an opponent.
- Shield - 30 Points - This defensive move shields you from the next hostile ability your opponent uses. It helps to protect your progress and also makes the adversary use up their points without effect.
- Hint - 30 Points - Fills in 2 random tiles on the player's board correctly. This boost can speed up a player's progress, providing a significant advantage, especially when used strategically.

- Peek - 15 Points - This ability lets players look at their opponent's board for five seconds. It is an information-based power that can unblur the other player's board. So, you can not only copy filled-in tiles but also it gives hints about whether you should attack or defend.
- Rotate - 15 Points - Rotates the opponent's board by 90 degrees in a random direction. This can cause some initial confusion to an opponent as it disorientates them before they resume their puzzle-solving.
- Double Points - 30 Points - For the next 30 seconds, each tile solved accurately will give double points. This effect doesn't influence the bonus for completing a row/column/square, but it encourages fast and correct solving of single tiles.
- Blind - 45 Points - This powerful ability allows players to blind their opponent by applying the same blur effect used to obscure the opponent's board. For the next 15 seconds, the opponent's board is obscured heavily hindering their ability to make progress, allowing the player to potentially get ahead of their opponent.
- Lock - 45 Points - The player can put a lock on any number of their choice for the next 30 seconds. During this time, the opponent cannot place that particular number anywhere on their board. To use it well the player must have good timing and understand what numbers are important to their opponent's board state; if they succeed in blocking a key number this could potentially stall the other player's progress

3.1.3 Difficulty

In the subject of Sudoku, the challenge and appeal often lie in the puzzle's difficulty which can range from straightforward to extremely complex. In the design of this competitive multiplayer Sudoku game, the decision to classify all generated boards as "Easy" with 36-46 clues provided serves a specific purpose. This is done purposely to keep the speed fast and make it accessible for players of different skill levels. This method finds a good middle ground: it's not too hard for new players, yet the extra mechanics provide a satisfying challenge for experienced ones. It also ensures that matches are brief and engaging.

Each round of the game is restricted by a four-minute time limit, a design choice that injects a sense of urgency into the gameplay. This time constraint encourages quick thinking and decision-making, pushing players to refine their strategies under pressure. The inclusion of an "Easy" difficulty setting across the board does not lessen the game's challenge but rather ensures that the competition remains a key focal point, with players not only challenging themselves by solving the puzzle but also challenging each other and competing against their opponent's tactics. This difficulty is selected to maintain quick and lively matches, avoiding lengthy standoffs while keeping a constant stream of competitive play. It also permits more varied methods for solving boards; players may tackle different parts of the board in their own unique manner. Using a harder difficulty board could lead to a more linear solution which would hinder some of the abilities listed previously, affecting engagement.

Finally, the ”Easy” difficulty setting facilitates a more inclusive gaming experience, making it simpler for individuals who are not familiar or skilled with solving complicated puzzles to participate and enhance their abilities. The decision to choose an easier difficulty should offer an approachable entry point to the Sudoku community, where learning and competition go hand in hand. As players get to grips with solving Sudoku puzzles, the challenge shifts from solving the puzzle to mastering the strategic use of abilities to outmanoeuvre opponents within the time limit and figuring out which abilities fit their playstyle the best. This delicate way of handling difficulty makes the game not too hard but still interesting, giving pleasure to all players.

3.1.4 Game Balance

In my game, achieving game balance is tied to the design of the abilities system and the methodology behind point acquisition. This equilibrium becomes very important for making sure that games are fair and competitive, giving an interesting experience where good strategies and skilful playing get rewarded. The abilities system allows players to use points obtained from solving tiles to affect their board or interrupt their opponent’s progress. This introduces a tactical element that goes beyond just solving Sudoku problems traditionally. The costs assigned to these abilities, along with their cooldown periods, are adjusted to motivate careful and strategic activation rather than repetitive use. Abilities that have a big effect come with more costs and need a longer time to recharge; this guarantees that players must think carefully about when they want to use them based on their present situation.

The equal distribution of opportunities between opponents is essential to the game’s balance. Initially, they begin with the same puzzle setup and starting clues. This indicates that success is not based on chance, but more on skill and tactics. During a game round, both boards are shown on the screen all the time. But the opponent’s board is blurred; only showing where tiles are filled but not revealing their numbers. This brings in a strategic element of uncertainty while still letting players understand how far along others have come in their own progress—yet they cannot copy moves directly unless helped by an ability that keeps it fair and competitive.

For the abilities system, all players have access to the 9 abilities. But before a game starts, each player must select one ability from every point tier. This is done to make sure there’s balance in choices of abilities. This selection necessitates a good sense of strategy as it needs to take into account, not just personal strong points and weak points that could be aided by selecting specific abilities but also predict possible upcoming difficulties or chances for attacking and defending. However, points are also a precious resource. If the players run out of time, then the victory goes to the player with the most points. This forces players to determine if using abilities is worth the risk of potentially having a lower point score overall, possibly leading to defeat.

Through these carefully considered design elements, the game strives to foster an environment where good decision-making, adaptability, and Sudoku-solving prowess are equally rewarded. This approach promotes an environment where each match is a test of not only puzzle-solving ability but also wit; it’s possible for strategic choices to greatly influence the competition’s outcome. The game maintains a fair competitive

landscape by giving all players equal tools and the same challenge, allowing success to be determined by skill, strategy, and good tactics.

3.2 Multiplayer

The multiplayer component of the Sudoku game is designed around a real-time, server-client model to deliver a seamless competitive experience. This architecture is chosen to optimize performance, reliability, and scalability, accommodating a broad player base while maintaining game integrity and responsiveness. By employing dedicated servers for matchmaking and game management, players are afforded the convenience of quickly finding and entering games, ensuring a smooth and efficient path from the lobby to gameplay. For people who want a game that is more private or controllable, the option to host a server on one peer's machine allows for direct connections through IP addresses and ports. This choice gives players the flexibility to compete in the larger community or within their environment. The server instances are made to use resources efficiently, reducing delays and making sure the game stays in real-time. This method of multiplayer design was chosen to offer a reliable and easy-to-use platform for fans of Sudoku.

3.2.1 Matchmaking

The matchmaking system in the Sudoku application has been designed to accommodate both spontaneous and planned multiplayer encounters, ensuring that players can engage in competitive puzzles according to their preferences. This dual approach to matchmaking is one of the game's aspects that help its accessibility and community engagement, while also addressing the practical aspects of online connectivity and server management.

Through the matchmaking options below, I have aimed the game to cater to a wide range of player preferences, ensuring that every player can find the competitive Sudoku experience that they are looking for, whether through quick and easy random matchups or the aimed nature of direct connection.

3.2.1.1 Random Matchmaking

For players looking to just boot up the application and get right into a game, the random matchmaking option serves this purpose. When chosen, players begin connecting to a dedicated server that cycles through all game instances hosted across various ports, stopping at the first instance with an open player slot. This process is designed to be quick, painless, and, ideally, easily scalable. The matchmaking system prioritizes quickly putting players in a game, using many server instances for lower waiting times. When two players are connected to the same game instance, they start playing without any delay. Because each game is on its own instance, if one server crashes, then others are not impacted, and matchmaking can still cycle past the downed instance.

To summarize, this method offers a straightforward way for players to test their skills against a random opponent, making every match a new and exciting challenge provided

there is a diverse enough player base.

3.2.1.2 Direct Connection

For users seeking a more tailored multiplayer experience, the option to host a game lobby on their machine provides a perfect solution. Players can create their own game server, essentially allowing one of the players to act as both the host server and a player simultaneously, which others can join by entering the host's IP address and port number. This setup is ideal for friends looking to compete directly with one another or if games need to be played in a tournament-like environment. While this method requires the host to configure port forwarding on their router, the steps to do so are relatively simple and other popular multiplayer games such as Minecraft or Terraria require this to host servers as well.

3.2.2 Real-Time vs. Turn-Based

The decision between real-time and turn-based gameplay is important in crafting not just a Sudoku application but also for designing any multiplayer puzzle game. Each format has its own benefits and disadvantages, affecting player interaction with the game and other players.

In a turn-based system, players would take turns making moves, potentially following speed chess rules where players have a set amount of time to make all their moves. If someone uses up all their time before the game ends then they lose. This way of playing allows for more thought and strategy because people can take longer to plan out what they will do next in the game. It brings down the stress related to quick decision-making, converting the game into more of a puzzle-solving skill and less about swiftness. The main problem with turn-based playing is that it has the potential to slow down the game's speed significantly, leading to longer matches and reducing the sense of urgency. Additionally, it might not fully capitalize on the competitive, high-energy potential I have set out in the creation of this game, where part of enjoyment comes from direct and immediate competition.

Alternatively, opting for a real-time format introduces a dynamic and fast-paced environment where players act simultaneously. This approach heightens the sense of urgency and competition; not only must players hurry to solve puzzles, but they also compete against time and their opponent. Real-time gameplay emphasizes quick mental analysis, adaptability, and stress-handling skills which can decide victory or defeat within seconds of gameplay. There are some potential flaws with the real-time system. One could be that the need for fast action in a game could make players concentrate more on short-term moves rather than thinking about long-range tactics. In Sudoku, where thoughtful planning is crucial, the real-time style may not allow players to thoroughly interact with their puzzle and its complexity could be reduced. A faster pace could also make players who do not excel at thinking fast feel stressed, or even just those players in remote regions with slower internet connections are at risk of losing more often as their speed of play is inhibited. This accessibility barrier could limit the game's appeal which would not be ideal.

Yet despite its shortcomings, real-time gameplay was selected for my application. This choice was made because it best highlights the thrilling and captivating elements of competitive play within the Sudoku community. It aligns with the intention to create a lively, high-stakes environment where players are constantly engaged and where the thrill of competition is sought after. Real-time matches help maintain a quick tempo for gameplay, suiting well the brief yet intense multiplayer Sudoku experience. In terms of play style, turn-based games have their advantages, particularly when it comes to strategic complexity. However, for delivering the quick and lively competition that keeps online multiplayer games interesting to a broad range of players, real-time game-play is more suitable in this instance.

3.3 User Interface

The user interface of any game should be designed to be as inclusive and accessible as possible and the Sudoku application does not break this design philosophy. Recognizing that the quality of the user interface can significantly influence a player's engagement and satisfaction, I have aimed to create an interface that is both visually appealing and functionally seamless. The UI serves as the bridge between the player and the game's mechanics, ensuring that players of all skill levels can navigate through the game, understand its rules, and access all the features it offers without unnecessary complexity or confusion.

3.3.1 Layout

The game layout of the game is designed to provide an easy-to-access user experience, segmented into four key pages: the Main Menu, Ability Select Screen, Connection Screen, and the Game Screen. Each page is crafted to facilitate ease of navigation and ensure players can access all necessary features without confusion, enhancing the overall gameplay experience.

Firstly the Main Menu serves as the entry point to the application. It is designed to be simple and functional. Players are presented with clear options to either join a game or choose their abilities. Additionally, a dedicated "How to Play" button brings up a window that explains controls for both keyboard and controller users, ensuring all players, regardless of their preferred input method, can comfortably navigate and enjoy the game. Lastly players can enter an options menu to change audio levels and set controls to their liking.

Next, the Ability Select Screen, which is accessed from the Main Menu is a critical part of the pre-game strategy where players are introduced to the nine available abilities, categorized into Basic, Advanced, and Elite tiers. This tiered system prompts players to choose one ability from each category, fostering strategic thinking from the outset. Detailed explanations of each ability's effects are provided through pop-up bubbles when hovered over, ensuring players are well informed about the abilities, leading to a more fulfilling experience.

Also accessible from the Main Menu is the Connection Screen. This screen facilitates

the transition from solo preparation to multiplayer engagement. Players are prompted to input a username, adding a personal touch to their game presence. Following this, they are given the choice between random matchmaking or direct connection, depending on what sort of game they are looking for as previously detailed in the multiplayer section.

Lastly, there is the actual Game Screen. Arguably the most important screen in the application, it has been designed for clarity and competitive balance. The layout is split into two clear halves, red and blue, with the opponent's side blurred to obscure specific details while still indicating their progress. A prominent timer at the top tracks the remaining game time, ensuring players are always aware of the urgency of their puzzle-solving efforts. Abilities are displayed below each player's grid, with distinct icons and color coordination reflecting their tier and providing at-a-glance information on point cost and cooldown, enabling strategic ability use during the heat of competition.

Together these four pages form a coherent and user-friendly game layout, supporting the player's journey from initial engagement through strategic preparation and into the competitive fray ensuring an enjoyable Sudoku dueling experience.

3.3.2 Interactivity and Input Methods

The game's user interface has been designed to cater to different player preferences and input methods. By allowing players to choose between keyboard, mouse or a combination of both, the game can follow the golden rule of universal usability in user-interface design[28] as outlined by Shneiderman et al. This approach ensures that players can interact with the game in a way that feels most comfortable and intuitive to them, adding to their overall experience and satisfaction.

For keyboard users, the "WSAD" or arrow key navigation provides a familiar and efficient way to navigate the Sudoku grid. A looping mechanism when reaching an edge of the board ensures a seamless and uninterrupted flow, reducing the frustration of having to rapidly press a key if a player wants to get to the opposite side of the board. The use of number keys for cell input and specific key combinations for abilities (Shift + number for jottings, Ctrl + number for abilities) follows multiple golden rules of interface design including those of consistency, ease of control and lastly reducing short-term memory load. By keeping keys consistent but adding modifiers, users don't have to memorize confusing key bindings and this leads to a more enjoyable and efficient user experience.

Users who prefer to use a mouse have access to a central keypad and click-based cell selection, which provides a direct and intuitive way to interact with the game. The keypad mimics the familiar layout of a physical Sudoku puzzle, making it easy for players to input numbers without the need for excessive mouse movements. The toggle switch for jottings and actual numbers is a clear way to switch input modes, reducing the likelihood of errors and enhancing usability.

The combination of mouse and keyboard support gives players who prefer a hybrid approach to easily swap between the two based on their needs and preferences. This flexibility allows players to optimize their gameplay experience which hopefully can lead to better enjoyment.

Godot's InputMap and action system allows for easy remapping of keys providing flexibility for players to customize controls to their liking. Additionally, Godot's key mappings are based on the physical location of keys, not just their assigned letter. This means that players using international keyboards with layouts other than QWERTY will not have awkward key placements when playing games made in Godot.

There is one unfortunate caveat to the input method that I have used in the design of this application. The current control scheme makes it difficult for two players to share one keyboard for local multiplayer. Since the keyboard controls are mapped to allow a single player to fully control the game and their Sudoku grid using WASD/arrow keys for navigation, number keys for inputting numbers, and modifier keys like Shift and Ctrl for abilities and notes, there are no remaining keys for a second player to use simultaneously on the same keyboard. This would limit a theoretical local multiplayer mode to be turn-based, as players take turns using a single keyboard. This hinders the intended fast-paced nature of the game and also limits the accessibility of local multiplayer.

Despite this inconvenience, the game's user interface design follows the golden rules of universal usability by supporting multiple input methods including keyboard, mouse, and a combination of both allowing players to interact with the game in the way that feels most accessible to them leading to a more enjoyable user experience. The consistency of key bindings and a familiar layout reduces the likelihood of errors and provides a consistent and familiar experience. The game's input system, built using Godot's InputMap and action system, provides flexibility for players to customize controls and supports international keyboard layouts, further enhancing its accessibility.

3.3.3 Accessibility

The design of the game has incorporated several accessibility features to ensure a wider range of players can enjoy the game. The support for both keyboard and mouse input, as well as the option for a hybrid of both, allows players with different preferences or physical abilities to interact with the game in a way that suits them best. This kind of flexibility in input methods is crucial for accessibility, as it can accommodate players with motor disabilities who may find one input method easier to use than another as explained by Yuan et al.[35].

Auto-scaling the game window to fit monitors of different sizes is a crucial accessibility trait. It guarantees that the user interface of the game stays readable and functional, no matter what screen resolution or size it is viewed on - a feature very significant for players who have sight problems.

The inclusion of a simple and understandable 'How To Play' page in the main menu is a useful tool for assisting with cognitive accessibility. It helps new players, especially those with cognitive disabilities, to learn the game mechanics at their own pace, reducing the cognitive load and making the game more approachable.[11]

Volume control for music and sound effects in the options menu is a standard but essential feature for auditory accessibility. It can be altered by players who have trouble hearing, so they can set it according to their requirements or likes. This feature also

benefits those playing games that may produce some sounds they are sensitive to, along with people who play in surroundings where high audio is not suitable.

Another important design decision is to give players the ability to rebind keys in the settings menu. This is a critical accessibility feature for players with motor disabilities. It allows them to customize the control scheme to fit their range of motion and comfort, which can make the difference between being able to play the game or not.

When in a match, high-contrast colours are used to differentiate between each player's Sudoku grid, the numbers on the grid and the background it is all on. Abilities are colour-coded based on their point cost using a standard green-blue-red colour scale. All of these features assist individuals with visual impairments in distinguishing game elements at a glance as mentioned by Jarmillo-Alc et Al. in their case study[13]. The use of distinct shapes and icons, such as the power-up buttons with unique symbols, allows for easier recognition and can be helpful for players with colour vision deficiency. The different point tiers are consistently colour-coded to assist in this recognition and are ordered from least to most expensive to ensure that the gestalt principles of continuity, symmetry and similarity[17] are followed throughout the game.

Additionally, the game interface includes a large, easily readable timer at the top of the screen, which benefits players with cognitive disabilities by providing a clear indication of the time remaining. The points are shown big and they update in real time, this is very important to make the game accessible because it gives the player a fast response. The option for "jottings," or small notes inside the Sudoku cells, could assist those with memory impairments by helping them remember possible number positions.

To sum up, the game's user interface is made with accessibility in mind. This includes visual cues, high contrast and supportive features to suit players who have varying abilities. These design decisions are compatible with the recommended methods for making online games accessible and enhancing an all-inclusive gaming experience.

3.3.4 Aesthetics

The main objective of the aesthetics of the user interface was to avoid clutter and keep it simple to achieve a good user experience. The game utilizes a blackboard backdrop to invoke nostalgia for a classroom setting where Sudoku puzzles might be enjoyed, such as a Maths class. The chalk-like typography shown prominently in the game adds to the educational theme. The colour palette consists of calm colours with white text to provide a focused environment suited to a game like Sudoku that requires concentration. Menus are kept clutter-free to add an emphasis that the game is built on its gameplay rather than any impressive graphics.

I named the game Sudoduel to convey to users what the game is from a glance. The game is a literal Sudoku duel, where players go head-to-head of wits and deduction within the familiar framework of Sudoku puzzles. During a round of the game, the two Sudoku grids side by side help signify the game's duelling nature. The game utilizes red and blue tones for the opposing sides to represent the competitive nature of the game.

3.4 Requirements

This section lists the particular conditions that must be met by each distinct part of the game to be considered successful.

3.4.1 Gameplay Requirements

- **Objective** - The main goal is a 1v1 competition to solve a 9x9 Sudoku board with a single solution that can be solved within a 4-minute time limit. Success should come from a mix of speed, accuracy, and logical deduction.
- **Fairness** - To keep the game fair and interesting, each player should be given the same randomly generated solvable board with the same available clues. Players with different strategies (e.g. focusing on speed vs accuracy) should have equal chances to perform well.
- **Points** - The game should have a functioning point acquisition system in place that rewards players more when lines are filled with no jottings than players who fill single cells with notes.
- **Abilities** - The game should have multiple abilities implemented that positively affect a player's board or negatively impact their opponent's. They should be of different point tiers based on strength and feature a cooldown time to prevent overuse.
- **Difficulty** - The generated board should have 36-46 clues for quicker, faster-paced games to maintain engagement

3.4.2 Multiplayer Requirements

- **Architecture** - The game should feature a real-time, server-client model that can be scaled easily
- **Flexibility** - Players should have the option of joining matches randomly, or by hosting private games that can be joined via a direct connection
- **Efficiency** - Server instances should be designed to use minimal resources and minimize delays ensuring real-time gameplay.

3.4.3 User Interface Requirements

- **Intuitive** - The game should feature an interface which users can navigate easily and should not get confused by.
- **Accessible** - The game must have a user interface that can be usable by all users regardless of input method.
- **Functionality** - The game's interface should be fully capable of navigating through menus and altering options. In a round, the user should be able to navigate the puzzle without issue

- **Responsiveness** - The game's interface must respond to the user's actions and adjust appropriately if the screen size is altered.

3.5 Unimplemented Design Concepts

This section outlines three design elements which were designed in further detail but were unable to be implemented due to time constraints or technical limitations.

3.5.1 Local Multiplayer

Playing together in the same room can sometimes make gaming more fun, turning it from a single activity into a social gathering where players can be physically present and compete with each other. The idea of a side-by-side local multiplayer option in SudoDuel could introduce a new level of personal interaction and real-time strategy, providing an immediate, shared experience that online play is unable to replicate. However, the way the game is designed now, with a mouse and keyboard setup, makes it quite hard to add a local multiplayer option. The game's interface and control method are made for one user in mind only. To make room for two players at the same time, I would need to change the control mechanics greatly. This not only means disrupting how things work in playing but also needing a new design of player actions with game elements - making local multiplayer an impossible addition without rethinking basic ways to input data at its present stage.

3.5.2 Team Based Competition

In SudoDuel, if a 2v2 competition mode was added it would change the game from being just a duel of wits to an area where strategy and working together are very important. In this setting, every team member would handle their grid but with a variation: their puzzle matches up to one of the opponent's, making it an indirect contest and possibly allowing for sabotage. The grids become battlegrounds where every play has consequences not only for players but also for their teammates. This interaction makes the game more complex because participants need to manage their puzzle-solving with chances to help their partner. This could be from strategic play or utilizing abilities that delay the other side.

In this joint framework, the gathering of points is a team effort that highlights the significance of working together and achieving victory as one. People who are playing have to communicate well, discussing when and who they will attack with their abilities. This adds an element of strategy; it isn't only about being quick at solving but also about which group can handle their gathered points cleverly, plan out attacks together and adjust to changes in both their and rival's grid situation. Also, this mode could bring about a more diverse competition environment. Teams might start to create their own methods and tactics. Players could take on roles similar to offence or defence, depending on what they are good at solving in Sudoku and the powers they select for themselves.

3.5.3 Bot Detection and Anti-Cheat

Anti-cheat systems in a game such as SudoDuel, where skill and strategy are very important, need to be complex and have many facets. To find bots or false players, they would use multiple methods of detection. The first method is by examining the behavior inside the game for certain patterns that indicate non-human play like doing repetitive movements, deciding too quickly at an unnatural speed, and maintaining perfect gameplay over a long period which isn't probable for even skilled human players. Also, the system can follow the path of mouse movements and the pattern of clicks. This is because bots usually show straight or simple movements with their cursors, unlike the complex and less predictable patterns of a human player.

However, crafting an anti-cheat system that is both effective and non-intrusive is a complex challenge. Cheaters are continually evolving their methods, and to stay ahead, developers need real-world data on how these individuals are exploiting the game. If the game is not accessible to a large number of users, it becomes hard to collect this data and understand how widespread the use of cheats is.

Additionally, sometimes anti-cheat methods unintentionally impact real users too - they require careful implementation. It's a sensitive equilibrium between protection against fraud in games and ensuring an amusing experience for every player. Anti-cheat systems that are good usually need a 'learning' time from real user actions. This implies the game must be allowed to face possible cheating, so it can improve how well and successfully its detection methods work.

Chapter 4

Implementation

4.1 Application Architecture

The game architecture is split into two main nodes: the Game Node and the Player Node. This separation of concerns allows for a clear division of responsibilities and encapsulation of related functionality.

Upon the creation, the game node adds two instances of the player node, one for each user. Figure 4.1 shows a round of the match in play. The area contained inside the green box is the game node. It features the central keypad, jotting switch and in-game timer. By keeping these elements in the Game Node, they can be easily accessed and managed from a central location.

Figure 4.1: Screenshot of a match from the perspective of Player 1/ the Red Player

The area contained within the red box in figure 4.1 is a player node. It includes the 81 cells in a grid container, and the three abilities below the grid, which have been implemented as custom button nodes with an Enum attribute. It also includes the

player's name and the point label. This encapsulation guarantees that every player's data and functionality are kept within their entities and can be handled separately.

Both the game node and the player node access a global game controller, which is created using Godot's autoload feature when the application is first run. The game controller stores player information such as name, multiplayer ID, the amount of points, and the player's selected abilities. By using an autoload, this data can be easily accessed from any part of the game including in the pre-game matchmaking section without the need for passing references or complex communication between parent and child nodes.

Communication between nodes is handled through direct method calls and signals. The game node can directly control the player nodes using functions of the player node since they are the child nodes of the game node. If a player node needs to communicate with the game node to notify it an ability is being used for example, then the game node can use Godot's signals. This allows for loose coupling and event-driven communication. If the game node needs to communicate with both player nodes at once, on both separate clients, then the game node can call functions using a Remote Procedure Call.

Object-oriented design is the main philosophy behind Godot, making it possible to compose game elements in a modular way. This design is supported by scenes, nodes and a scene tree structure that forms a hierarchy. The architecture of Godot also involves an effective signalling system for communication among game objects.

This is important as it has led to an implementation that provides several overall advantages such as there's a distinct separation of responsibilities between things related to the entire game and those specific only to players; player data and functions are contained inside Player Node, providing good encapsulation; game-wide elements can be managed centrally within Game Node; accessing shared player information is made simple through autoloaded game controller, and nodes can communicate flexibly using direct method calls along with signals plus RPC.

4.2 Puzzle Generation

For the creation of the Sudoku puzzle, I first make a 2D array having size 9x9. Then I fill in the top left cube, middle cube and bottom right cube with numbers from 1 to 9 mixed randomly as shown in Figure 4.2. This is done so that these three cubes don't impact each other because Sudoku only looks at vertical and horizontal constraints, not diagonal ones. Filling in these cubes first makes it easier for the generator to fill in the rest of the tiles.

8	5	6						
2	3	4						
7	1	9						
			1	4	3			
			8	9	7			
			6	5	2			
						4	9	8
						1	6	2
						5	3	7

Figure 4.2: Sudoku puzzle partially generated

After filling these initial cubes, a solving mechanism is used to fill in the remainder of the board. This involves recursively trying random numbers in each empty cell, checking if the number is valid in that position, and backtracking if a solution cannot be found. The solving mechanism continues until the entire board is filled with a valid Sudoku solution. Once the board is filled, a duplicate is created and stored as the solved board. This will serve as the reference for the final puzzle.

To create the actual puzzle, the original filled board is taken and some of its cells are cleared. The numbers 1-81 are shuffled in a list, and then, in a loop that runs between 36-46 times (for easy difficulty), items are popped from the shuffled list one by one. Each popped number corresponds to a cell in the Sudoku grid, and that cell is cleared. The number of cells cleared determines the difficulty of the puzzle, with more cells cleared resulting in a harder puzzle. As discussed in the design chapter, an easier difficulty was chosen for a better gaming experience.

After clearing the cells, the remaining cells that still have numbers are marked as disabled on a separate 2D array. This array will be used to keep track of the pre-filled cells during gameplay and can be used as a reference in case the board is affected, such as if it is rotated.

4.3 User Interface

4.4 Abilities

There have been nine abilities implemented in this application at the time of submission, with three per tier. Their implementations are explained below.

4.4.1 Peek

When triggered, the player node sends a signal to the game node indicating a hostile attack. The game node then runs a function on the opposing player, but not via RPC as this ability's effect is client-side only. The function called simply toggles all the cells

between their blurred state and their normal state. A timer is then started for 10 seconds, and upon timeout, the cells revert to their blurred state.

4.4.2 Rotate

Similar to the above, on being triggered, the player node sends a ”hostile action” signal to the game node. However, unlike peek, the rotate effect should appear on both clients and thus the game node calls the function on the opponent via RPC.

Because the Sudoku grid is represented as a 2D matrix, rotating it 90 degrees is simple. First, the matrix is transposed, so that rows become columns and columns become rows. Then each row is reversed. This produces a matrix that is rotated 90 degrees clockwise.

This rotation is applied to the current board state, the solved board, the disabled cell array, and lastly the array keeping track of jottings.

The jotting array is a special case as each cell is given a 9-bit integer, and when represented as a binary, the correlating jottings would be shown. For example if we take the number 147 when represented as a binary it produces 010010011. From this, we can gather that the jottings 1,2,5 and 8 are active on that cell.

With all of these matrices rotated, the game can continue as normal, and the board remains solvable.

4.4.3 Redact

When triggered, the game node receives the ”hostile action” signal and calls the redact function on the opponent via RPC as the effect should be shown on both clients.

This ability activates an animation player node that Godot uses. Firstly via a random number generator, it is decided which row or column is chosen to be redacted. At that point, the player node moves a simple coloured rectangle to the start of the row or column based on relative coordinates to its parent node. The rectangle starts with one of its dimensions set to zero, depending on if it is vertical or horizontal. The animation player then starts its appropriate animation. Over the next two seconds, the animation player changes the size of the rectangle such that it covers the whole line, and then after 10 seconds, it reverses back to its original size.

The size and position of the shape are synchronised via the player node’s Multiplayer Synchroniser so that the rectangle appears in the same place on both screens.

4.4.4 Shield

When this ability is triggered, the player node sends a signal to the game node indicating that the player is using a buffing ability. The game node calls the shield function via RPC to ensure that the player is shielded for both clients, otherwise it could lead to desync.

The shield function simply sets a boolean from false to true. For any hostile abilities used on a shielded player, after points are spent but before the ability is triggered, the

shield prevents the ability from activating but sets the shield flag to false in the process.

4.4.5 Double Points

In the point calculation section of the code, points awarded are multiplied by a modifier. Typically this modifier is set to 1, but when this ability is triggered, the modifier is set to 2 for the next 30 seconds.

4.4.6 Hint

When triggered, the player node randomly goes through the current board and selects 2 empty cells, and fills it with the correct number from the solved board array.

4.4.7 Eraser

When triggered, the hostile signal is sent to the game node, which then calls a function via RPC that forces the opponent to erase all of their jottings on each cell.

4.4.8 Blur

This ability is very similarly implemented to the peek ability, but instead of unblocking the opponent's board for the player that used the ability, it simply blurs the board of the opponent on their screen. The screen of the player who used it remains unchanged. After 15 seconds, the blur effect wears off.

4.4.9 Lock

This ability sends the hostile signal to the game node, but instead of calling a function on the opponent's board, it sets the player node to a state of waiting for an input in the form of a number either via the central keypad or just the player's keyboard. After a number is chosen, the player sends a signal to the game node with the chosen number attached. This number is then sent to the opponent player via RPC and locks the number. For the next 30 seconds, that number cannot be used to fill in cells or add jottings of that number anywhere on the board.

4.5 Multiplayer

The multiplayer component of the online Sudoku game is a cornerstone feature that allows players to engage in real-time duels against each other. The multiplayer framework is powered by Godot's High-Level Multiplayer API. This API makes the process of managing network state synchronization, peer-to-peer communication, and RPCs (Remote Procedure Calls) less complicated. The use of this API guarantees that all networked events like placing numbers on the Sudoku grid or activating abilities are effectively spread to every connected client in real time. This helps in keeping gameplay integrity and responsiveness intact.

The dedicated servers, where the quickplay sessions are hosted, are run on a DigitalOcean Droplet. DigitalOcean offers cloud servers which are dependable and can be easily scaled up, making them suitable for an online multiplayer scenario. The game utilizes screen sessions, which are virtual terminal sessions that can be detached and reattached, allowing for persistent server processes. This setup enables the game servers to run continuously in the background, even when no active administrative session is connected, ensuring players can connect to a quickplay server at any time.

Upon selecting quickplay, the game initiates a connection attempt to the designated server IP on port 9980. To provide a seamless player experience, a timeout threshold of 3 seconds is implemented to prevent prolonged waiting periods during server connection attempts. If the first server is at capacity or otherwise unavailable, the client automatically proceeds to attempt connections on subsequent ports, incrementing by one each time (9981, 9982, etc.). This approach not only allows for efficient server load distribution but also ensures that players can quickly find and connect to an available game session without manual retries.

In the event that all servers from port 9980 to 9990 are full, the system is designed to loop back and retry from port 9980. This cycling mechanism acts as a dynamic load balancer, redirecting players to the first available server slot. Automating the process of finding open ports helps to remove players from the possible annoyance of servers not being available. The variety and ability of ports, along with DigitalOcean's adjustable structure, guarantee that game can handle more people playing at same time - an important feature for any online multiplayer game that is growing. This orderly and player-centered method of managing multiplayer servers highlights the dedication of the game towards offering a strong and easy-to-use competitive stage for Sudoku lovers worldwide.

Chapter 5

Evaluation

5.1 Game Performance

In evaluating the performance of SudoDuel, it is essential to assess the various performance metrics that directly impact the user experience and system efficiency. Measuring performance in online gaming is complex. It includes not only how fast and reactive the game feels to a player, but also the way resources are used for creating this experience. In SudoDuel, metrics like storage space needed, memory usage (both RAM and VRAM), CPU use, and network bandwidth are important parts that help make sure the game runs well with good response time for players while providing an interactive multiplayer atmosphere without putting too much pressure on their hardware or network system's capacity. Below is an analysis of the game's performance to give an all-around view of how SudoDuel stands in terms of efficiency.

5.1.1 Storage Space Utilization

The export system of the Godot Engine works well for getting games like SudoDuel ready on different platforms. It compiles game code, assets and configurations into an executable to run along with a .pck file - this compressed package contains all the resources needed by the game and has been optimized specifically for the devices it targets, these being Windows and Linux systems. The asset pipeline is efficient at converting resources into formats that allow them to be efficiently used across various hardware, showing how adaptable Godot is in deployment.

For Windows and Linux, the sizes of SudoDuel without compression are 107.88 MB and 109.21 MB respectively. These values match with what is normal in the industry as we see similar Godot games like "Godoku,"[1] which has a size of 105.66MB. These numbers show that SudoDuel uses storage effectively because it manages assets very well and can give a high-quality multiplayer experience even while making good use of space for saving data. The size difference between these two platforms could be due to specific dependencies related to each operating system, highlighting the careful optimization aspect during the development process for this game.

[1] https://templewulf.itch.io/godoku

5.1.2 Memory Usage

5.1.2.1 RAM

Figure 5.1 displays the overall RAM usage in SudoDuel. The graph is split into three clear sections. The first is the initial starting of the application and the menu navigation. The second is when the client enters actual gameplay, and lastly, the final section is when the client returns to the main menu after a game is completed.

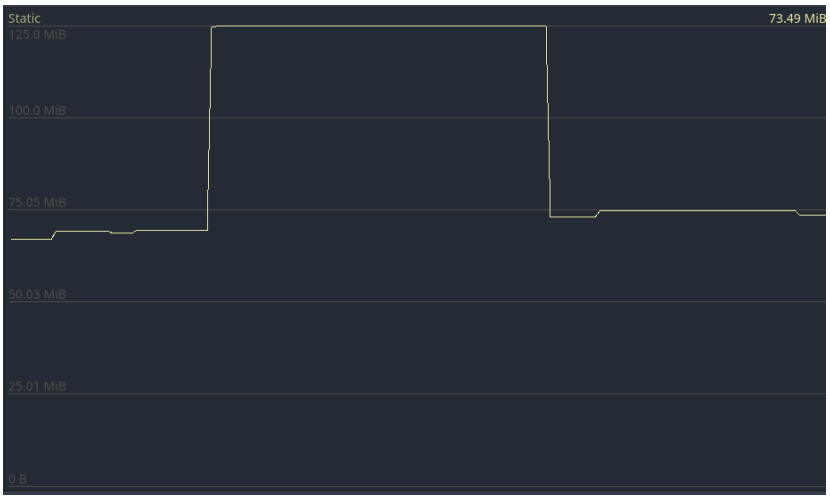

Figure 5.1: Graph detailing the RAM usage of the game over the course of 1 match

During menu navigation, the game maintains a low and stable RAM footprint, which shows good menu asset optimization and no memory leak problems. This stable condition makes sure that the starting interface of the game is active and provides a strong base for the gaming experience.

Upon entering gameplay, there is a significant yet expected rise in RAM usage. This peak correlates with the game's transition from simple menu screens to the more resource-intensive gameplay, where additional assets are loaded, and game logic becomes active. However, even with the rise in RAM usage, the highest point doesn't show an overuse of memory. This implies that game assets and scripts are efficient in terms of memory use.

After the gameplay concludes and the game returns to the main menu, the RAM usage drops back down close to its initial state, demonstrating that the game effectively deallocates memory that was used during gameplay. This coming back to almost a baseline level of memory usage is a good sign because it means that "SudoDuel" is effective in clearing up resources, not allowing unnecessary memory growth across many games. In general, the RAM usage graph suggests that this game has been fine-tuned to control its memory – guaranteeing steadiness and an uninterrupted gaming experience for players.

Figure 5.2: Graph detailing the VRAM usage of the game over the course of 1 match

5.1.2.2 Video Memory

Figure 5.2 demonstrates the game's effective VRAM management, showing low usage at the start when we navigate through menus. This suggests that menu graphics are well-optimized since they can handle different GPU capabilities. When a match is started, there's a sharp rise in VRAM use because the game needs more complex graphical assets for competitive gameplay; this shows the necessary boost to render a competitive play environment without overloading typical GPU capacities.

The VRAM consumption decreasing back to almost the starting point at the end of gameplay indicates good resource cleanup and deallocation by the game. This handling efficiency with VRAM guarantees stable performance, even during long playing periods, and shows promise for smooth operation on different systems—a crucial factor in accessibility and player experience. The maximum VRAM being under 512MB also indicates that the game should run without issue on a CPU's integrated graphics, thus not requiring a dedicated video card expanding the potential player base further.

5.1.3 CPU Usage

Appendix A features screenshots taken of three frames in which critical actions are taking place. The first image, shown in Figure A.1 shows when a player hosts a game. The "*host game*" and "*on host button down*" functions take 2.3 ms and 1.88 ms respectively. Compared to the other script functions, they take a slightly longer amount of time. This can be expected since hosting the game requires the initializing of network services and preparing the lobby. These tasks typically demand more of the CPU. Despite this, however, the script takes less time than the time it takes to process a frame.

Joining a game is shown in Figure A.2 and the "*on join button down*" function takes 4.21 ms indicating that the process of connecting to an existing game server and synchronising the player's initial game state with the new client is well within a reasonable time frame.

The most notable CPU time is observed when a game is started as shown by Figure A.3. Overall, it takes 622.74 ms to run the function that starts the game for the client. This suggests a considerable workload during this phase. This is because at this point the game is setting up logic, loading assets, and generating the initial Sudoku Puzzle. These are one-time costs at the game's start and are not recurring per frame which is critical to ensuring the game's performance.

In general, the usage of the CPU is in a good range for hosting and joining matches. This shows that tasks related to the network are optimized well. The increase in CPU usage when starting a game is significant, but because it matches with an event happening only one time compared to continuous frame-by-frame cost, it does not necessarily point towards a performance problem. The one-time cost could be reduced if the initialization of the game is optimised, such as using a more efficient Sudoku-solving algorithm. However, considering this takes less than a second, it should not be overly noticeable to the player. It could be disguised by adding a short loading screen to the game.

5.1.4 Network Bandwidth

In SudoDuel, direct player connections during hosted games exhibit commendable network efficiency, with both upload and download speeds averaging around 15 KiBps. The fact that the game only requires a small amount of bandwidth indicates that it uses a lightweight networking protocol. This kind of protocol is designed to efficiently transfer data, making sure players can play smoothly even if they have slower internet connections and minimizing lag.

Conversely, when players connect to a dedicated server, the upload requirement increases to 30 KiBps, a needed increase to handle more communication load between the server and its clients. Although there is a rise, Despite this increase, the upload speed remains modest, indicating that SudoDuel maintains network efficiency even in more complex server-client interactions. This is essential to sustain the integrity of the game and ensure that all players have a synchronized and fair gameplay experience that is as lag-free as possible.

The low bandwidth in SudoDuel can be attributed to the game's implementation of multiplayer synchronizers, which avoid syncing every frame. Instead, they employ a consistent replication rate, updating game states at regular intervals that balance real-time interaction with network resource conservation. This approach helps reduce the overall network load and preserves bandwidth

5.2 Issues and Limitations

The current server infrastructure of "SudoDuel" shows a significant constraint in its demand for separate ports for every game server. This one-port-per-server design might create problems of expansion as the game becomes more popular. It needs a wide range of open ports to handle many games happening at once, making network settings complex and possibly risky from a safety perspective. In a shared hosting setting, the maximum number of open ports can also limit how many games can be played at one

time. This might prevent the game from growing its player community without moving towards a more intricate and potentially expensive server arrangement.

Additionally, the requirement for players to set up port forwarding to host a game introduces a complexity that might discourage those who are less interested in technical details. When it comes to setting up port forwarding, this is a task that differs depending on the type of router being used and normally demands a particular level of networking know-how which regular players may lack. A matchmaking server could make this procedure more straightforward, resulting in an easier hosting experience for users. With a matchmaking server, players could effortlessly connect to game servers without the need to tinker with their network settings, thus enhancing the accessibility of game hosting and potentially increasing the game's adoption.

A third aspect where SudoDuel is lacking is controller support. The absence of these features affects the game's accessibility as many players prefer or are accustomed to using a gamepad for different reasons such as personal comfort, disability access and device type being utilized (keyboard/mouse vs joystick). Although it is usual to use a keyboard and mouse for puzzle games, adding controller support could make the game more appealing and accessible. This way, people who like playing with a controller can also enjoy SudoDuel, making it an inclusive experience for everyone involved! Looking at the future updates of "SudoDuel", it may be a big enhancement to offer controller support as the gaming sector is shifting towards varied input methods.

These issues highlight the importance of playtesting and iterative development in game creation. It is through these processes that developers can identify practical challenges and areas for enhancement, making the game better in successive versions, so it is more enjoyable and easier for users to play. Regularly improving and responding to user feedback is very important for turning a good game idea into an excellent final product that people love.

5.3 Requirements Evaluation

Personally, I believe that the game fulfils almost all of the requirements laid out in Section 3.4. In particular, all of the gameplay elements have been implemented as intended. The finished game is fair and features a working points system. All abilities are fun and work as designed. Lastly, the generated boards are often able to be completed within the time limit laid out.

Whilst the multiplayer architecture is currently quite complex, it does fulfil the criteria of being a server-client model. Players can quickly connect to a dedicated server or host servers themselves, provided they have port forwarding configured. The bandwidth used during a match is considerably low suggesting that the game runs efficiently, allowing for a smooth real-time experience.

Finally, the user interface of the game is certainly functional and responsive as the game uses minimal resources to operate the game's interface. Players can very easily interact with the game through mouse or keyboard inputs with the only limitation being the lack of controller support.

Chapter 6

Conclusions

6.1 Achievements

The following is a list of achievements of this project:

- Comprehensive literature review on the evolution of Sudoku, including various solving techniques, the development and design principles behind successful games, and an in-depth analysis of game engines with a special focus on the justification for selecting Godot as the development platform.
- Design and implementation of an efficient Sudoku board generator for easier difficulty puzzles
- Design of a user interface that is accessible to a range of users and is intuitive to use and understand.
- Implementation of a unique abilities system that adds to the classic Sudoku game, giving a new difficulty for players and encouraging strategic thoughts in a competitive multiplayer environment.
- Successful integration of multiplayer features, utilizing Godot's high-level multiplayer API to ensure seamless online interactions among players, including real-time puzzle solving and competition.
- Utilizing Godot's profiling tools to evaluate the game's performance identifying and addressing potential bottlenecks to optimize the overall gaming experience.
- Critical assessment of the game's limitations and areas for improvement, helping identify areas for future enhancements and iterations of the multiplayer Sudoku game.

6.2 Improvements

As discussed in the evaluation of the application, the biggest limitation of the game is how multiplayer servers are hosted and run. Using a port multiplexer or proxy server

structure can simplify the one-port-per-server model in SudoDuel. With a proxy server, all players are able to connect through one entry point that is commonly on standard ports like 80 or 443 which are usually open in many networks and can pass through firewalls without difficulty. The proxy server then smartly directs these connections toward the correct game server instance, managing network resources effectively and lessening the number of ports required for handling.

This setup is enhanced further when paired with a matchmaking server, which may assign players to their game sessions without revealing the intricacies of server selection and network setups.

Lastly, one more improvement that could be made is introducing a leaderboard or ranking system to SudoDuel to create a stronger competitive advantage by promoting progression and competition among players. This characteristic would give clear objectives and acknowledgement for players' abilities and accomplishments, motivating them to keep playing and participating as they compete for better rankings and superior status within the game's community.

6.3 Future Extensions

If this project were to be continued further the following extensions could be made:

- Incorporate User Feedback Mechanisms: Program in-game systems that permit users to give direct feedback on aspects like difficulty levels, user interface design, and overall game satisfaction.
- Player Technique Analysis: Develop discrete and non-invasive analytical tools to study how players interact with the game. This data could then be used to refine game mechanics and balance difficulty.
- Addition of a Matchmaking System: Create a matchmaking system that pairs players based on skill level, ensuring fair and challenging matches. This system could use a hidden rating or visible ranks to improve the competitive aspect of the game.
- Social Features and Community Building: Integrate social features such as friend lists, private messaging, and community forums. These features would foster a sense of community, encourage engagement, and make it easier for players to connect and organize matches.

6.4 Final Remarks

This report has explored the process of designing and developing ”SudoDuel,” an online multiplayer Sudoku game, highlighting its requirements and innovative solutions that shaped its creation. Through performance evaluations and the implementation of key features, the project fully explored the area of game development to produce a finished product. Despite some of the limitations found in the evaluation, all goals were achieved and requirements fulfilled, thus I consider this project a success.

Bibliography

[1] Ernest Adams and Joris Dormans. *Game mechanics: advanced game design*. New Riders, 2012.

[2] Keith Burgun. *Game design theory: A new philosophy for understanding games*. CRC Press, 2012.

[3] Heather Maxwell Chandler. *The game production handbook*. Jones & Bartlett Publishers, 2009.

[4] Easybrain. Play free sudoku now! `https://sudoku.com/`, 2018.

[5] George Skaff Elias, Richard Garfield, and K Robert Gutschera. *Characteristics of games*. MIT Press, 2012.

[6] Leonhard Euler. Recherches sur un nouvelle espéce de quarrés magiques. *Verhandelingen uitgegeven door het zeeuwsch Genootschap der Wetenschappen te Vlissingen*, pages 85–239, 1782.

[7] Carlo Fabricatore. Gameplay and game mechanics: a key to quality in videogames. 2007.

[8] Bertram Felgenhauer and Frazer Jarvis. Enumerating possible sudoku grids. *Preprint available at http://www. afjarvis. staff. shef. ac. uk/sudoku/sudoku. pdf*, 2005.

[9] Tracy Fullerton. *Game design workshop: a playcentric approach to creating innovative games*. CRC press, 2014.

[10] 2018 Game Developer. How the room devs succeeded on mobile, 'the only option left to, Jan 2018.

[11] Lee Garber. Game accessibility: enabling everyone to play. *Computer*, 46(06):14–18, 2013.

[12] Todd L Harper. *The art of war: Fighting games, performativity, and social game play*. Ohio University, 2010.

[13] Angel Jaramillo-Alcázar, Paz Cortez-Silva, Marco Galarza-Castillo, and Sergio Luján-Mora. A method to develop accessible online serious games for people with disabilities: A case study. *Sustainability*, 12(22):9584, 2020.

[14] Alysia Judge. An athletic aesthetic: The making of lara croft go, Sep 2015.

[15] Zubair Khan, Ashish Kumar, Sunny Kumar, and Uttar Pradesh-India. Sudoku puzzles by using x-wing techniques.

[16] Raph Koster. *Theory of fun for game design*. " O'Reilly Media, Inc.", 2013.

[17] Steve Krug et al. Don't make me think, revisited. *A Common Sense Approach to Web and Mobile Usability*, 2014.

[18] Hung-Hsuan Lin and I-Chen Wu. Solving the minimum sudoku poblem. In *2010 International Conference on Technologies and Applications of Artificial Intelligence*, pages 456–461. IEEE, 2010.

[19] Arnab Kumar Maji, Sunanda Jana, and Rajat Kumar Pal. An algorithm for generating only desired permutations for solving sudoku puzzle. *Procedia Technology*, 10:392–399, 2013.

[20] Gary McGuire, Bastian Tugemann, and Gilles Civario. There is no 16-clue sudoku: Solving the sudoku minimum number of clues problem via hitting set enumeration. *Experimental Mathematics*, 23(2):190–217, 2012.

[21] Radek Pelánek. Difficulty rating of sudoku puzzles: An overview and evaluation. *arXiv preprint arXiv:1403.7373*, 2014.

[22] Scott Rogers. *Level Up! The guide to great video game design*. John Wiley & Sons, 2014.

[23] Andrew Rollings and Ernest Adams. *Andrew Rollings and Ernest Adams on game design*. New Riders, 2003.

[24] Ed Russell and Frazer Jarvis. Mathematics of sudoku ii. *Mathematical Spectrum*, 39(2):54–58, 2006.

[25] Katie Salen and Eric Zimmerman. *Rules of play: Game design fundamentals*. MIT press, 2003.

[26] Jesse Schell. *The Art of Game Design: A book of lenses*. CRC press, 2008.

[27] Charles P Schultz and Robert Denton Bryant. *Game testing: All in one*. Mercury Learning and Information, 2016.

[28] Ben Shneiderman and Catherine Plaisant. *Designing the user interface: strategies for effective human-computer interaction*. Pearson Education India, 2010.

[29] David Sirlin. Balancing multiplayer competitive games. Sirlin.Net – Game Design, Dec 2009.

[30] Unity Technologies. Unity real time development platform — 3d,2d,vr,ar engine. `https://unity.com/`.

[31] Unity Technologies. Gameobject. `https://docs.unity3d.com/2023.3/Documentation/ScriptReference/GameObject.html`, 2023.

[32] Unity Technologies. Monobehaviour. `https://docs.unity3d.com/2023.3/Documentation/ScriptReference/MonoBehaviour.html`, 2023.

[33] Tetris Wiki. Garbage — Tetris Wiki, The Free Encyclopedia. `https://tetris.wiki/Garbage`, 2023.

[34] Takayuki Yato and Takahiro Seta. Complexity and completeness of finding another solution and its application to puzzles. *IEICE transactions on fundamentals of electronics, communications and computer sciences*, 86(5):1052–1060, 2003.

[35] Bei Yuan, Eelke Folmer, and Frederick C Harris. Game accessibility: a survey. *Universal Access in the information Society*, 10:81–100, 2011.

Appendix A

Profiler Results

This section features screenshots of the Godot 4 debug profiler discussed in section 5.1.3

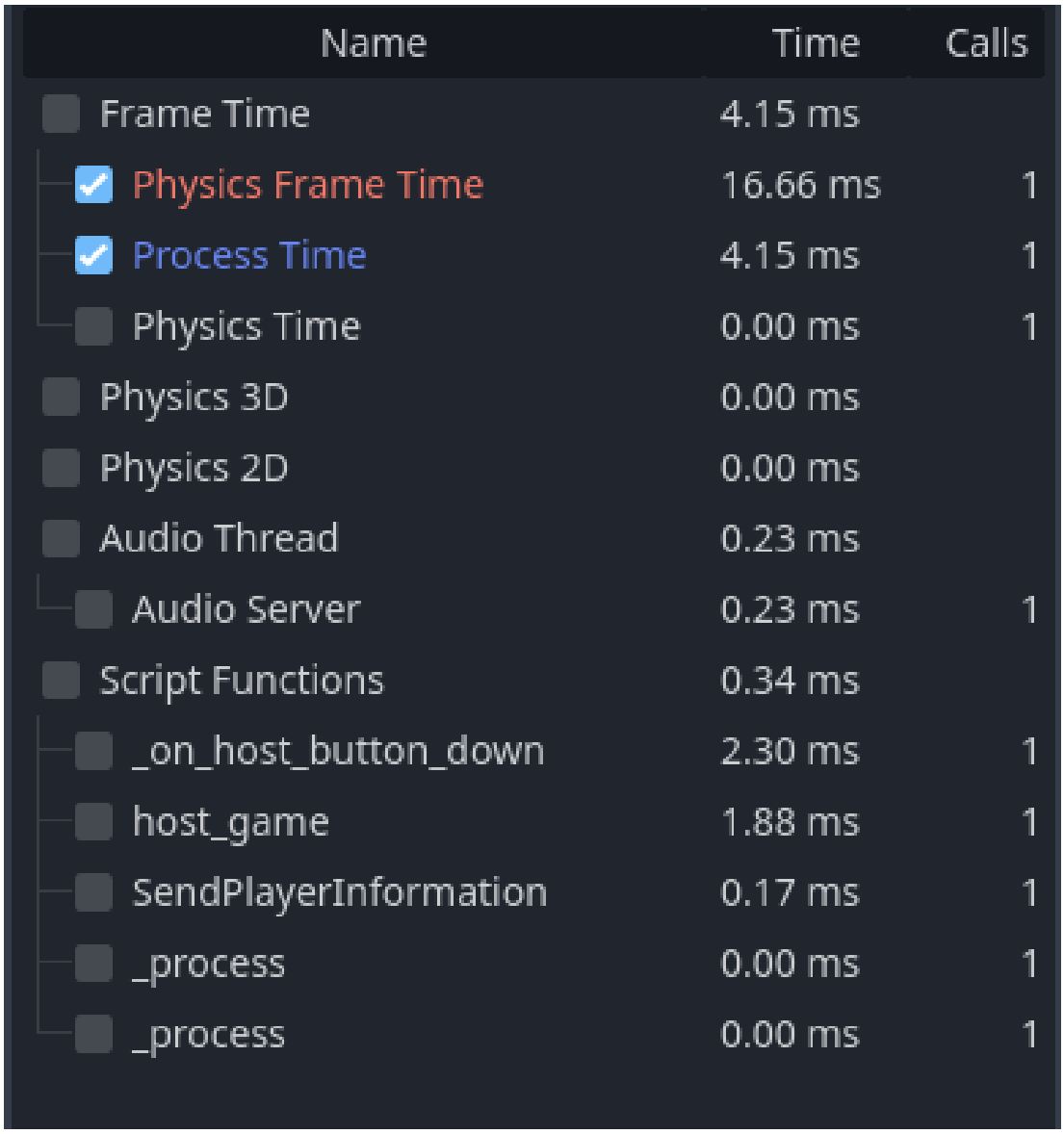

Figure A.1: Screenshot of Godot profiler during the frame a game is hosted

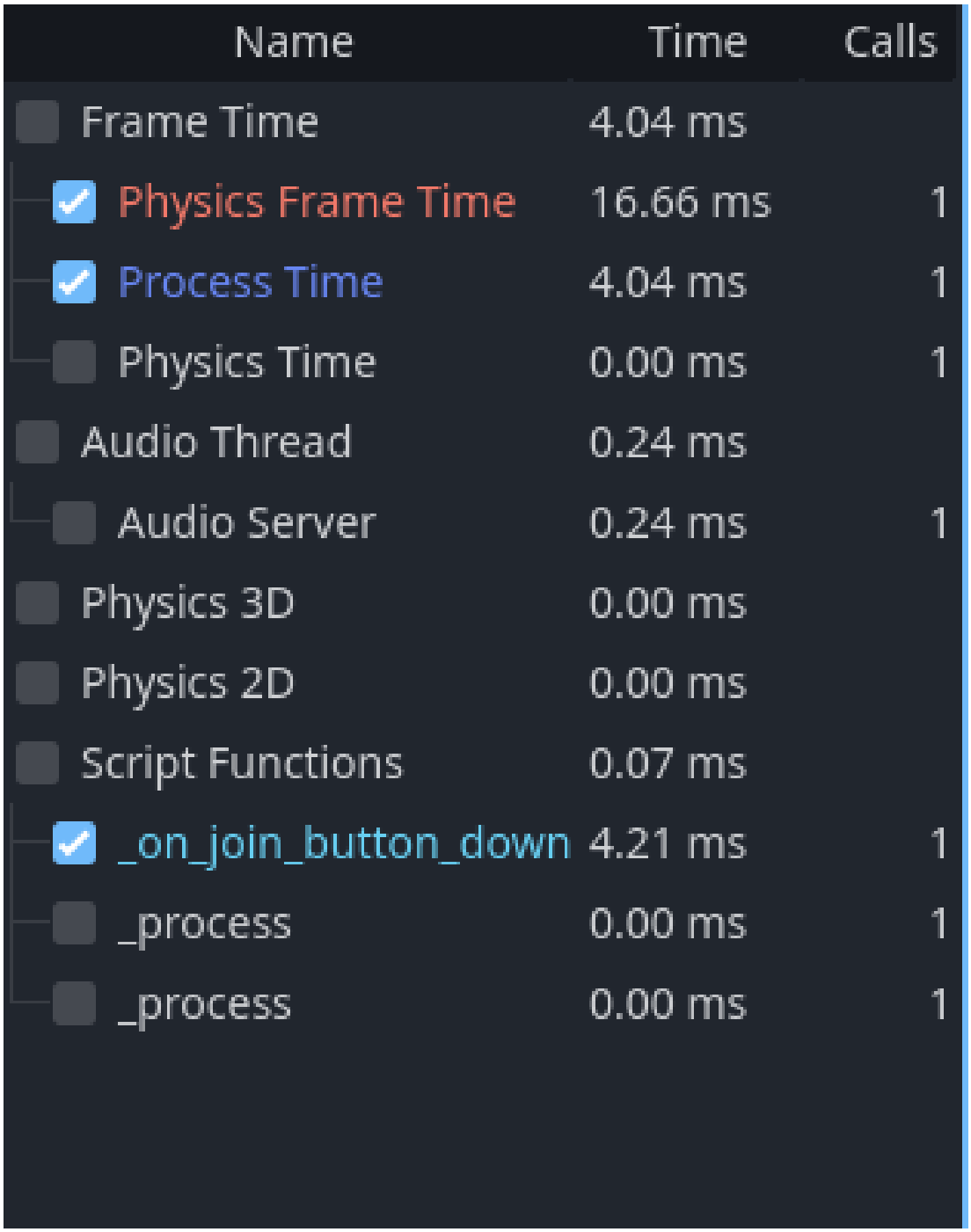

Figure A.2: Screenshot of Godot profiler during the frame a game is joined by a player

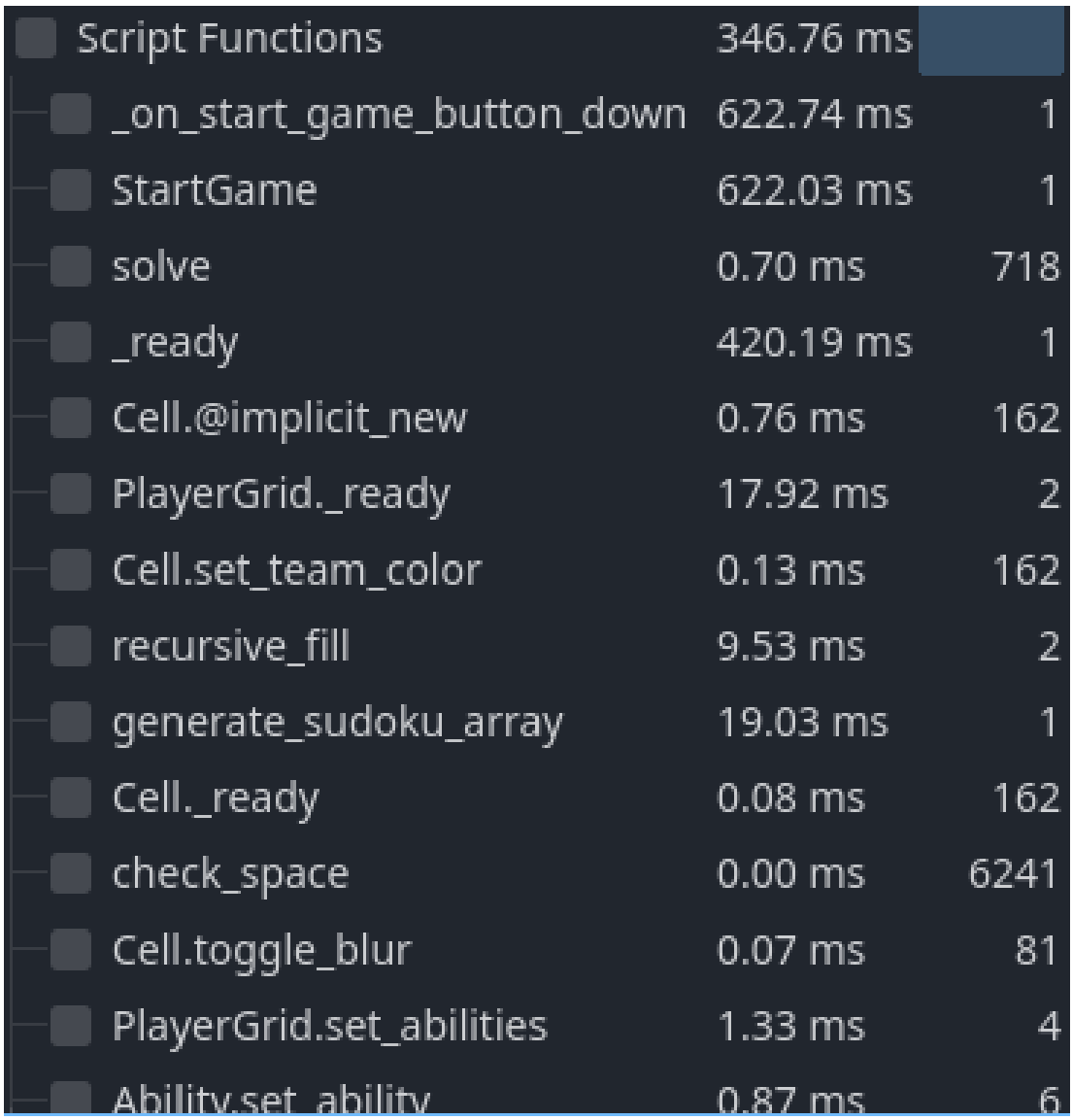

Figure A.3: Screenshot of Godot profiler during the frame a match is started