FACULTY
OF INFORMATION
TECHNOLOGY
CTU IN PRAGUE

ASSIGNMENT OF BACHELOR’S THESIS

Title:	Distributed interface for cooperative game
Student:	Anna Moudrá
Supervisor:	Ing. Jan Buriánek
Study Programme:	Informatics
Study Branch:	Web and Software Engineering
Department:	Department of Software Engineering
Validity:	Until the end of summer semester 2018/19

Instructions

Design and realize prototype of distributed interface for cooperative multiplayer game. Focus on a game in a chosen immersive environment (movie theatre, conference hall, classroom, etc.), in which players will control the game via mobile devices and the output, course of the game and instructions will be displayed on a central screen (projection, CAVE, dome, etc.).

1) Study available literature concerning matters of cooperative games.
2) Design proof of concept for such distributed game; describe used mechanics and principles, ideally in a form of short Game Design Document.
3) Implement prototype of a chosen simplified game for multiple mobile devices and one central screen.
4) Test and assess the prototype in real operational environment.
5) Write down and analyze your observations.

References

Will be provided by the supervisor.

Ing. Michal Valenta, Ph.D. Head of Department	doc. RNDr. Ing. Marcel Jiřina, Ph.D. Dean

Prague January 31, 2018

FACULTY
OF INFORMATION
TECHNOLOGY
CTU IN PRAGUE

Bachelor's thesis

Distributed interface for cooperative game

Anna Moudrá

Department of Software Engineering
Supervisor: Ing. Jan Buriánek

May 14, 2018

Acknowledgements

I would like to express my sincere thanks to my supervisor Ing. Jan Buriánek for his valuable insight and advice. I would also like to thank my family for continuously supporting me throughout my studies.

Declaration

In Prague on May 14, 2018 . . . .

Citation of this thesis

Moudrá, Anna. *Distributed interface for cooperative game.* Bachelor's thesis. Czech Technical University in Prague, Faculty of Information Technology, 2018.

Abstrakt

Tato bakalářská práce se zaměřuje na lokální kooperativní hry a jejich použitelnost v různých imersivních prostředích, jako jsou kina, planetária či CAVE systémy. Důraz je kladen na využití mobilních zařízení a verbální mezilidský kontakt, který tyto typy her vyžadují. Praktická část této práce zahrnuje analýzu, návrh a popis následné implementace prototypu distribuovaného rozhraní pro týmovou kooperativní hru. Prototyp implementovaný pomocí herního enginu Unity3D je vytvořen pro specifickou promítací plochu a je koncipován pro ovládání přes webový prohlížeč na mobilních zařízení.

Klíčová slova návrh a implementace prototypu, lokální kooperativní hry, distribuované uživatelské rozhraní, imersivní prostředí, bring your own device gaming, real-time interakce, Unity herní engine

Abstract

This thesis focuses on local cooperative games and their usability in immersive environments such as movie theaters, planetaria or CAVE systems. The emphasis is put on encouraging verbal communication and the use of mobile devices. The practical part of this work incorporates the analysis, design and subsequent realisation of own prototype of distributed interface for local cooperative game. The prototype, implemented in Unity3D game engine, was developed for a specific projection surface and is controlled via web browser on mobile devices.

Keywords prototype implementation, local cooperative gaming, distributed interface, immersive environment, bring your own device gaming, real-time interaction management, Unity game engine

Contents

List of Figures

List of Tables

Introduction

In recent years local cooperative gaming has been gaining in popularity, and while every public screen or projection – in science centres, train stations or shopping malls – can be a potential playfield, solutions offering spontaneous, setup-free gaming experiences are sparse. While some museums and planetaria have their own interactive exhibits or voting systems, expensive equipment and lengthy setup are often in the way of creating a spontaneous experience. To address these complications, this paper presents a cooperative game with minimal setup and distributed interface, which allows for players use their own devices to connect.

Challenges

There are several technical and even artistic challenges to creating a distributed interface for cooperative game, as the main goal is to create a system, that is complementary to the experience of a spectator. In other words a system, that does not take away from the immersive experience by creating additional barrier between the audience and the screen. The substance of the interactive screening content should share some of the qualities of passive content, namely a visual appeal and an engaging element, which may be either a story or some of the primary motivations of gamers e.g. competition or completion of a task. As it is crucial to not excessively divert the attention from the main source of content, the form of interaction ought to be minimal and intuitive. Furthermore, the requirements on usability of the game's user interface will only grow with the number of players, moreover so, when they must communicate and cooperate effectively.

Goals

There are three main conceptual goals to this thesis. Firstly, the text aims to educate about the possibilities of interactivity in immersive environments. Readers should be able to recognise the advantages and limits of interactive content in immersive environment and gain knowledge of existing solutions. The second goal is the design and realization of own prototype of local cooperative multiplayer for chosen environment. Since the prototype is designed without given foundation, the author focuses on all major steps of the process of creating a distributed interface for a cooperative game, including the game logic and designing own graphical user interface. The final part of this text describes the testing and assessment of the prototype in real operative environment, pointing out users' reception and room for future improvement.

Chapter 1

State of the art

This chapter briefly resumes the state of the art of local collaborative gaming. The author lists several up to date solutions to the problem of distributed applications for local cooperative gaming, along with the advantages and limits of each solution. From the conducted research the author additionally outlines several universal principles applied to the cooperative games.

Before the text will delve into the analysis it is necessary to introduce several terms and concepts relevant to this thesis.

Immersive environment The terms immersion and immersive environment are most commonly linked to an artificial, computer-created scene or world within which the onlooker can immerse themselves. Nowadays, the phrase is mainly used in regards to Virtual Reality or Augmented reality systems using headsets. Even less artificial environments like planetaria, art exhibits or movie theaters can create an immersive experience, most commonly by providing a limited level of interactivity.

Event games These type of games often incorporate aspects of networking games or icebreakers as they are mostly taking place at large, often corporate, events and meetings. Event games are designed to connect large groups of strangers and initiate conversation, often by facing the group with a problem that can be solved only by joint discussion [1]. Other types of event games are created to convey commercial messages or raise awareness about a given issue and can be found in science centres or museums [2].

Game moderator Sometime also known as Game Master is a person who acts as organiser and administrator to local and online cooperative games. The responsibilities of a game moderator often include preparing the game session, directing the course of the game and, in some cases, managing possible conflicts between players [3].

1.1 Forms of cooperative gaming

The upcoming section briefly resumes the different concepts of local cooperative gaming.

1.1.1 Collaborative cinema

Historically, movie theaters and planetaria were first to introduce interactive storytelling to their spectators.

In 1967, Kinoautomat, the world's first interactive film has been released. The screening took place at a theater with built-in remote controls and was essentially divided into 9 parts. At the end of each part, a live actor appeared on stage and asked the audience to vote on which way the scene should continue [4].

While immersive and interactive screenings moved predominantly to 3D projections and synchronized physical effects such as wind blowing and temperature changes, interactive films are made even today [5].

Many of those interactive "choose-your-own-story" films in the past were created for planetaria and other theaters with built-in voting systems. But companies like CtrlMovie are creating interactive movies that can perform in a much less limiting setting – as is stated on their website: *"the audience's smart devices, the CtrlMovie app and a local WiFi enable decisions based on majority votes [6]."*

1.1.2 Console games and mobile cooperative games

Nowadays, local cooperative games are quite common in households and video game consoles like Nintendo Switch[1] or Sony's PlayStation 4[2] allow for multiple controller connections. While these games are very popular and engaging, they are usually limited to 4 or 8 players and require a specific video game console with a set of controller, that effectively restrict the game play to one location.

Another new trend in local multiplayer gaming has been mobile cooperative games, operated through smartphones. Most of these games are simple arcade or racing games that support multiplayer mode which lets the players compete with each other and does not require any cooperation. Some of these games, such as SpaceTeam[3] are focused strictly on players' verbal communication by supplying each player with only a subset of resources needed for solving the game [7]. The mobile devices are usually connected through a local Wi-Fi or a Bluetooth connection.

[1]https://www.nintendo.com/switch/
[2]https://www.playstation.com/en-us/explore/ps4/
[3]http://spaceteam.ca/

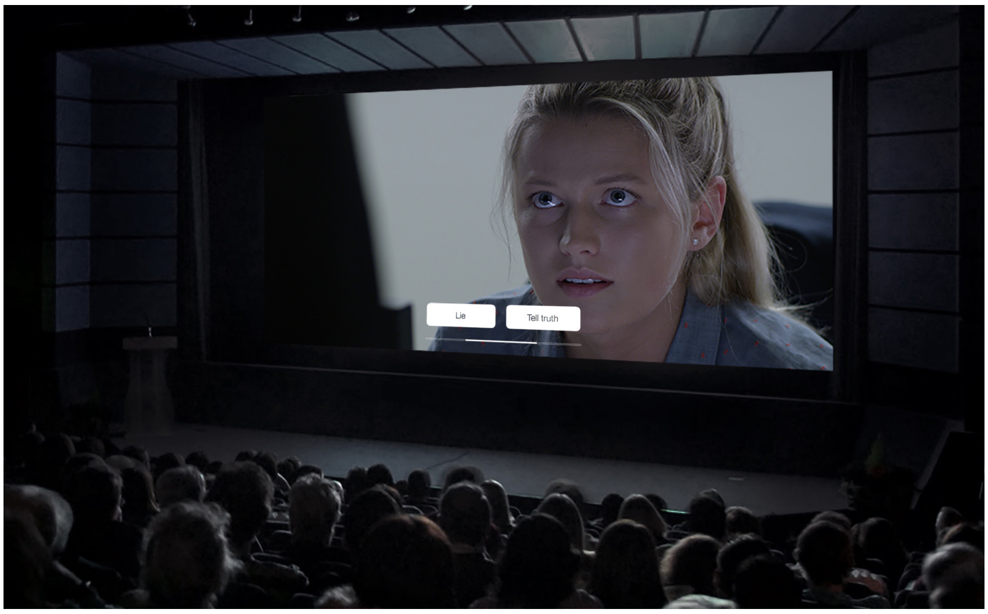


Figure 1.1: CtrlMovie: Photo from interactive movie *LateShift* where audience votes on the actions of the protagonist via their smartphones.
Source: [6]

As local smartphone multiplayer gaming effectively solves the issue of mobility, the games are designed for a limited number of players, usually 8 at maximum, as game designers must also take into consideration the limited screen space.

1.1.3 Collaborative interactive displays and special events games

Science centres and museums often look for ways to make their exhibitions more engaging to the public, and thus interactive exhibits are becoming more and more popular. According to Eric Klopfer and collective [8], simply equipping visitors with audio guides and pocket devices is counterproductive as the technology makes them feel more separated from the exhibition. Klopfer instead approaches the idea of using technology to encourage human interaction with other visitors. For example the China Science and Technology Museum in Beijing offers a collaborative VR experiences by creating a virtual scene for 5 participants. The participants, each equipped with VR goggles, must complete the task of transporting an aircraft carrier from a hangar and subsequently take-off [9]. Other science centers also include this idea by offering round table games, cooperative dance pads or music trackers to engage visitors [10]. For the same reasons of engaging groups of people together and encouraging networking, many corporate companies implement cooperative and round table games at their meetings, as it is an effective way to encourage the participants to engage in discussion.

1.2 Existing solutions

This section describes some existing solutions of local multiplayer games developed in recent years and lists their advantages and disadvantages. From those findings, the author then concludes in section 1.3 which features are worth keeping in the development of own prototype.

1.2.1 SpaceTeam

As already mentioned in section 1.1.2, SpaceTeam is a local mobile cross-platform multiplayer game developed in 2012. The game accommodates from 2 to 8 players, each participating on their own device. Each player is assigned a random control panel with buttons and their names, as seen in figure 1.2. Each player is then required to follow time-sensitive instructions to push a certain button. However, the instructions are being sent randomly to other teammates, which leads to the players shouting instructions at each other. While this application does not utilize a central screen, the game logic could

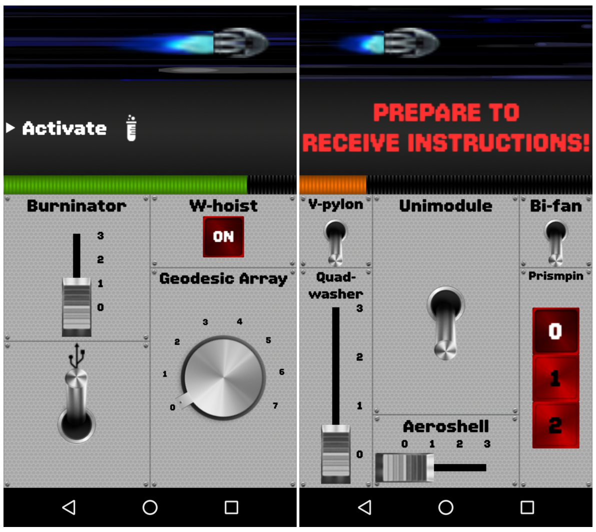


Figure 1.2: SpaceTeam: Two screens next to each other showcasing different control panels and instructions

be adapted well to an environment with multiple controllers - sets of control panels – and a central display. Displaying the game play on TV screen has been already been done as SpaceTeam has been the first game to launch on

AppleTV [11]. The game could be well adapted to a team based game as well, where the teams would compete against each other and track the results on a central display. The sci-fi sentiment makes the game suitable for science centers or planetaria.

Advantages

- Mobile controllers; this game can be played virtually anywhere as devices can be connected through Bluetooth.
- Clear directions and simple controller manipulation for the players.
- Communication between players is necessary for success.

Disadvantages

- Setup requires cooperation – as all players must simultaneously hold a button on their devices – even before the game starts.
- For 8 players at maximum.
- Players need own mobile devices to connect to the game.
- Very fast paced game. Players solve multiple problems at once which results in very loud environment[4].

1.2.2 Massive Mobile Multiuser framework

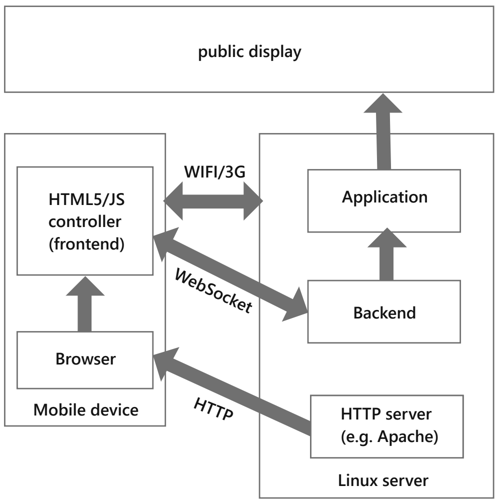
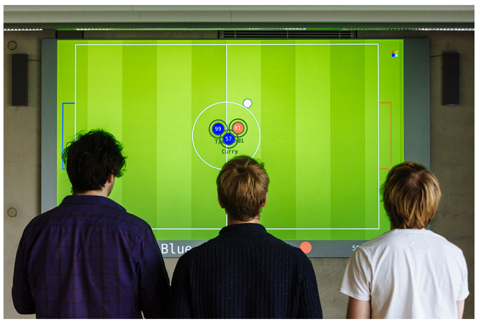


Figure 1.3: M^3 framework architecture.
Source: [12]

[4]This could, of course, be a positive trait as well, but the size of the playing crowd must be taken into consideration.

In 2015 a collective of authors from Bauhaus-Universitat Weimar created and tested the Massive Mobile Multiuser framework, hereafter referred to as the M^3 framework. Their work presents a software platform enabling a set-up free, real-time interaction for a larger number of players on a single public display. The M^3 framework consists of 3 modules; while the application and backend runs on a Linux server, the frontend runs on client devices and is accessible via mobile web-browser, as seen in figure 1.3[5].

The authors implemented a multiplayer soccer match that enables 17 concurrent users. The connection between the server and client is kept over a HTTP protocol using WebSockets. The players may connect and disconnect anytime during the 3 minutes long matches. The game itself is very minimal yet extremely fast-paced and sensitive to any amount of lagging. The team of authors assessed the prototype in real-time operative environment with over 8 hours of play. Their surveyed data shown positive feedback to perceived latency, easy set-up and some issues with lost users' input [12].



(a) M^3: MMMBall playfield on one central screen.

Source: [13]

(b) M^3: Controllers accessible via web browser on smartphones.

Source: [13]

Advantages

- Mobile game that can be easily transported.
- Players can spontaneously join the play during a game.
- Scalable architecture.

Disadvantages

- Players need own mobile devices to connect to the game.
- Players do not need to talk to each other and can just observe the screen.

[5]Figure recreated from the original diagram.

1.2.3 Fishing at VIDA! center in Brno

The learning center VIDA! in Brno introduced an interactive exhibit called Round Table in 2014. The table is consists of 12 separate client computers for players and one central computer with round screen in the middle for everyone to see. The first implemented cooperative game, called Fishing[6], simulates a pond where the central screen shows an available amount of fish and players can each gather a number of it. After each of the finite 10 rounds, the number of caught fish is added to the overall number and the rest of the fish in the pond can multiply. The collective goal is to gather as many fish as possible while not interrupting its ecosystem by overfishing. Leaving only certain number of fish in the pond leads to failure [10] [14]. While the system was prepared for hosting more types of games, only one was implemented. In 2017 David Savič designed and tested new central control system for the Round Table. This new system offers choice between two games but was designed to be easily extensible by specifying the API and requirements for future games [15].

Advantages

- Interesting algorithm for multiplying fish between rounds.
- Players must engage in conversation.
- Players do not need own any equipment to join the game.

Disadvantages

- Non-scalable to more that 12 players.
- Built-in system.
- Players cannot join the play during a game.
- The control system must have been reimplemented to accommodate a different type of game.

1.2.4 Dukovany Power Plant Information Centre

Another interactive interactive exhibit in the Czech Republic is at the Dukovany power plant information center. According to a personal conversation with a guide at the center [17], the exhibit is divided into 6 separate sections, each of which has some kind of display. Groups of visitors divided into teams are given small voting controllers, photographed in figure 1.6, and then proceed to collectively discuss and answer displayed questions in each section. After all groups have finished the course, the results of voting, teams order and correct answers are shown. Each exhibit is conducted by a guide who is in charge of distributing the controllers and setting up each section. The

[6]Rybolov in Czech original.

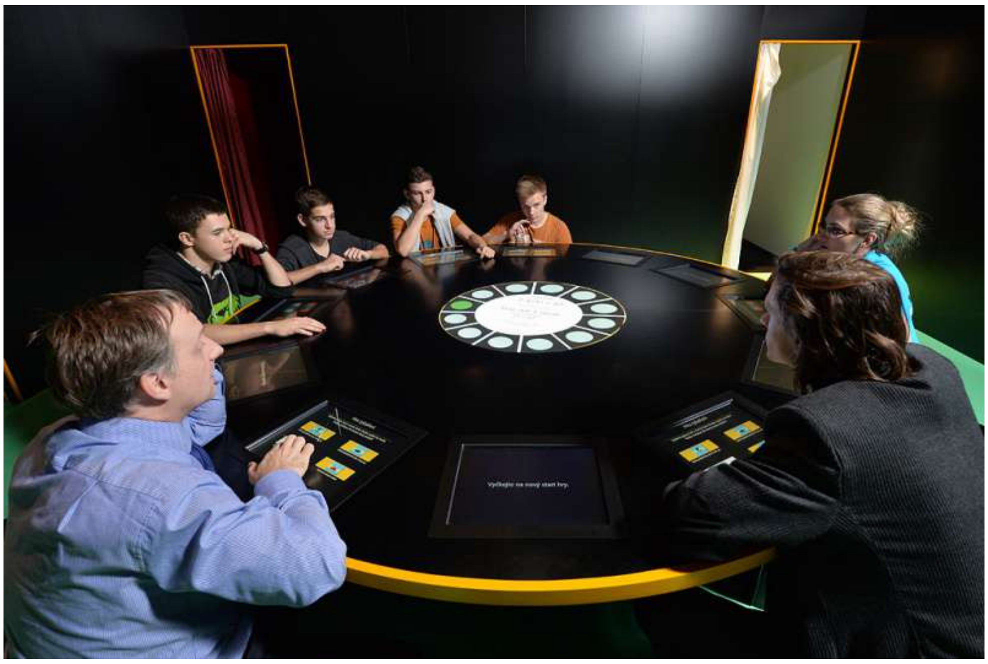


Figure 1.5: RoundTable: Interactive exhibit at VIDA! Science Center in Brno. Source: [16]

premise of such interactive exhibit is an added element of competition that motivates the visitors, mainly groups of pupils, to be attentive to the displayed information.

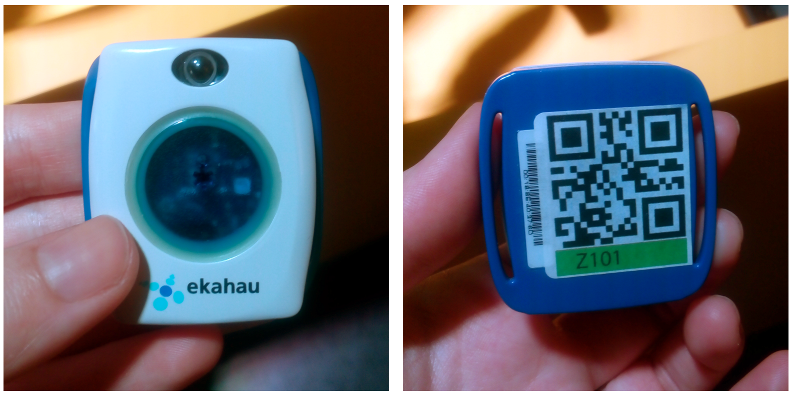


Figure 1.6: Controllers used at Dukovany Power Plant Information Centre

Advantages

- Moderator directs the course of the game and can adjust the speed to the players.
- Players do not need their own devices.
- Mobile controllers, at least to the extent of one building.
- Clear directions and simple controller manipulation for the players.
- Collaboration is encouraged.
- Competition between teams increases motivation to answer correctly.

Disadvantages

- Lengthy setup.
- System cannot be easily transported, number of players is dependent on the number of available controllers.
- Non-responsive controllers – players cannot see anywhere how they voted.
- Players cannot join nor leave during the game play.
- Players cannot change their answers.
- Players do not know the results or team score until after the course.

1.3 Which features should be used in own solution?

There are many different solutions to the problem of implementing a distributed interface for a cooperative game, and while all of the listed above are functional, most of them are a solution to a specific problem in a specific location.

This thesis aspire to present a more universal solution, with design that is easily extensible to accommodate different locations and types of games.

Such distributed interface should not be dependent on a pre-set type and amount of controllers, as drastically different amount of players is needed for different type of games. For that reason, the model of BYOD[7] implemented by the M^3 framework or the CtrlMovie company is the most location independent, easily scalable solution. The former solution is more user-friendly for the reason that it does not require to install an additional application on the device, but is accessible via web browser on any mobile device.

Another concept worth incorporating in own design of cooperative game interface is the fact that players need to verbally communicate with each other. As the number of players should be scalable to tens or even hundreds of people, depending on the type of game, the best way around this seems to be

[7]Bring-your-own-device is a policy allowing to work with own device. In this context the term refers to the players being able to connect their own device to the game interface and use it as a controller.

dividing the players into teams, which was also the solution at the Dukovany Power Plant information center. That way the communication should be kept manageable even with 10+ players.

On the other hand it is not hard to imagine that with that many people it could be difficult to clearly communicate the game settings, such as the game start, what game to choose if there is a choice and how to distribute the teams. For that reason the idea of having an extra person act as a moderator should be incorporated into the design for a game with more than 10 players.

1.4 Principles used in cooperative gaming

This section describes several other, more universal, aspects to cooperative multiplayer games than location and used technologies.

1.4.1 Cooperative game theory

Game theory is an important part of more complex gaming mechanisms as it deals with problems concerning behaviour between different agents with varying motivations [2].

Tragedy of the commons is one of the most popular game theory principles used in local cooperative games. The term is used to describe the problem of sustaining a public resource that everybody is free to overuse, which leads to a social dilemma. Each party wants to act in their current best interest, and take as much from the shared pool of resources as they can, which is contradictory to the common good of all users and eventually leads to resource depletion. In reality the best collective good can be only reached by consistent collaboration between all parties [18].

The tragedy of the commons is in many ways application of the Prisoner's dilemma, another game theory principle, on a larger scale [2].

The described game Fishing or the WarmGame [2], focusing on the issue of climate change, are great examples of incorporating game theory principles into local cooperative gaming.

1.4.2 Group dynamics

There are different dynamics between players that know each other and complete strangers. A study from 2014 found out that players must figure out the right way to communicate and coordinate every time they become a part of a new team. Moreover, they must maintain a positive atmosphere as it, according to the study, directly translates to the team's well-being. [19]

Individual players Generally speaking, when developing a game for spontaneous, large scale gaming, there should not be a reason to force players into teams, unless socialisation is the primary motivation behind the game. Instead of forcing a single player in a cooperation with a group of strangers and then expecting them to work well together, it is better to let a players join the game individually and then reward them for cooperation. The best way to motivate individual players to collaborate on a problem is by setting a goal that cannot be reached otherwise. There are two types of goals for players in a cooperative game:

1. Shared goals – for example in the soccer match implemented in the M^3 framework the shared goal was to move the ball across the playfield and score on the other team.
2. Interlaced goals – players have different goals, however, one player in the process of completing their goal simultaneously creates an environment where second player can also succeed [20].

Playing in teams Playing in teams is the base for most cooperative games that require verbal communication between players.

1. Preexisting groups – oftentimes the players are already in well formed groups and join a cooperative game with the intention to play together. Forcibly separating these groups of people into different groups can lead to a complete loss of motivation to play [21]. For that reason it's important to consider when it's best to let players choose their own teams. It is important to note that in many educational games the act of separating good friend, and thus forcing players to come out of their comfort zone is desirable.
2. Groups of strangers – dividing strangers or not fully formed groups into teams is common for games focused on collaborative learning or networking

1.4.3 Time and speed

If timing plays an important role in a single player game, timing in collaborative games is crucial, as the players need some room to communicate, but having too much time can make the game slow. Having too much time in cooperative gaming can often lead to one player taking over the process of decision making, which defies the concept of solving a problem by joint discussion. Depending on the implemented game, short time limit resulting in forced simultaneous choice is a good way of preventing this issue. On the other hand, too little time can cause the game to become very difficult and frustrating [22] [23]. To find the right balance, the final product should be thoroughly tested and adjusted accordingly.

1.4.4 Game difficulty

The environment and used technology are detrimental to the level of cooperation that should be required from the players. According to Reuter et al both age and limits in communication should be further considered in developing a cooperative game. Age of the target group dictates the complexity of the game in respect to required communication and timing difficulty. Additionally, any factors limiting human interaction (e.g. loud environment, players cannot see each other, ...) should be taken into consideration and further lower the amount of required cooperation between players [23].

Chapter 2

Analysis

The analytical part of this thesis describes the initial process of creating a local cooperative game and its functional and non-functional requirements. The process goes from simple analysis of target audience and choosing an appropriate game to a more thorough process of defining roles and use-cases. The chapter ends with a deduced state machine diagram that will serve as a base structure for the following application design.

2.1 Initial Game Idea

When creating a game, no matter the final platform, it is crucial not to underestimate the basic motivations that attract and engage players. There are certain concepts and steps that need to be followed to make the game enjoyable. What is needed first is a basic game idea that can initiate the development process. The game idea may be completely new or, as it is in our case, can be inspired by other successful game. The initial game idea settles mainly these following issues:

- Who is the target player (single person or group) and what are their motivations for playing.
- What are the primary expectations of these people, what will attract and engage them.
- What are the main motivations for creating the game and what possible outcomes are the creators hoping for.

All these issues (the whats and whys) should be addressed before even considering the initial development requirements (the hows).

2.2 Target audience

The target player for a local cooperative game, that implements the BYOD gaming policy, is someone who owns a smartphone and is comfortable with performing basic tasks such as connecting to wifi or installing and running a mobile application.

Other than these requirements the game should not be excessively excluding considering that a group of 40 individuals, even in a similar age group, can be very diverse.

Another crucial part to consider is that the players, unless they will have exterior motivations, will not want to be bothered with a lengthy setup or complicated game rules. Therefore, the goal is to create a simple game accessible without prior installation via mobile browser. The game should not be targeted at any specific age group, since it could possibly drive away other potential players.

2.3 Applicable games

Below the author lists several local cooperative games applicable to the target group.

Spaceteam The game is described in section 1.1.2. A group of 40 players would have to be divided into smaller teams. The teams would then compete against each other, having their score be displayed on the central projection.

Fishing The game is described in section 1.2.3.

Snake or slither.io The setting with 40 players and one central screen lends itself to a game like slither.io[8] where each player could operate one snake and compete with other players for who lasts the longest. But because this game would be more competitive than cooperative, another simple modification of the game snake could be implemented. Several team consisting of 4 players could play this game, when each player would control one snake on the central screen. However, each player on the team would have only one way to move the snake – left, right, speed up, slow down. The team would inevitably have to verbally communicate to move the snake.

Numbers The players are divided into several teams and each player has a set of numbers on their controllers. Numbers for each team are randomly appearing on the main screen and the players in one team must collectively match the sum on screen to get points.

Catch the Egg This game is mostly used as an icebreaker or a team building game. Each player controls movement of a particle, from the top of the central screen are slowly falling objects (eggs). The players have to form a basket shape from their particles and catch the falling object before it reaches the bottom of the screen. Players may be divided into teams but since there

[8]http://slither.io/

should be a minimal number of particles to create the figurative basket, the minimal number of players per team could be 5–10. Very technically similar to this game is the soccer match realized in the M^3 framework.

2.3.1 Pace and game choice

The type of audience plays an important role in choosing the pace of the final game. For a fast-paced game, such as SpaceTeam, the emotions rise rather quickly, with players shouting at each other with vigor. While the pace keeps the game engaging, at higher numbers of players it becomes increasingly more difficult to keep the volume at a reasonable level. On the other hand, a slow-paced game can easily become uninteresting and players will find themselves either unattached or waiting at their co-players.

Out of the above listed the game of Numbers, hereafter referred to as *NUMBERS*, became the most viable option for the design and prototype for a few reasons. The game is easily scalable from a small number of players to accommodating even a larger crowd (100+ players), furthermore the pace can remain flexible in relation to the crowd size. Yet the greatest advantage the game has over the other options is the simplicity of the rules.

Regardless of the chosen cooperative game the distributed interface should be designed as universally as possible, to accommodate, possible future extensions.

2.4 Initial game Design: Idea Refinement

The game *NUMBERS* is a simple cooperative game for roughly 40 players playing on their own mobile devices. *NUMBERS* will be local multiplayer game as all players are required to have access to one main projector screen (4K resolution[9] projection screen at Techmania Science Center in Pilsen) and engage in inter-personal communication.

2.4.1 Gameplay

The game will divide players into teams in which they are required to cooperate to reach the final goal within limited time. The essence of a problem will be displayed on one central screen and each player will have a set of resources on their device.

2.4.2 Mindset

The game should be complex enough to rise an emotion and start a discussion but rather simple in design and accessibility to prevent confusion, embarrassment and frustration. The problems will most likely be extremely simple to

[9]4K, also known as Ultra High Definition, is a resolution of 3840x2016 pixels [10]

solve (i.e. simple tap on an object on mobile screen) but the game itself fast-paced to get players under pressure.

2.4.3 Proof of Concept

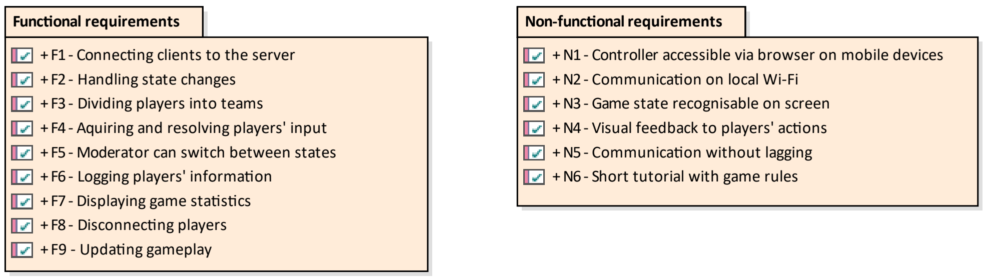
To demonstrate functionality a basic prototype of the game *NUMBERS* will be implemented. Such prototype will showcase the basic mechanics of cooperative multiplayer game such as connecting players together and dividing them into teams, starting a game and handling gameplay for a limited amount of time. Prototype will be required to undergo basic functionality testing in real functional environment by roughly 5 – 40 players.

2.5 Requirements

Each game that will be able to use the central control mechanics must fulfill certain requirements. These requirements are:

- Game has strictly set maximum and minimum player count. If a current game has maximum number of players connected, another player cannot join the game.
- Suitable graphical user interface.
- Only one instance of a game is playable at one time. If one game has been started, another one cannot start until first game is cancelled or finished.
- Moderator can cancel game at any given point.
- Players cannot dynamically join a game after it has started. Even when maximum player count has not been reached.
- Game should react to a player disconnection.

These initial demands were further analysed and subdivided into functional and non-functional requirements, as seen in figure 2.1.



Functional requirements

- ✓ + F1 - Connecting clients to the server
- ✓ + F2 - Handling state changes
- ✓ + F3 - Dividing players into teams
- ✓ + F4 - Aquiring and resolving players' input
- ✓ + F5 - Moderator can switch between states
- ✓ + F6 - Logging players' information
- ✓ + F7 - Displaying game statistics
- ✓ + F8 - Disconnecting players
- ✓ + F9 - Updating gameplay

Non-functional requirements

- ✓ + N1 - Controller accessible via browser on mobile devices
- ✓ + N2 - Communication on local Wi-Fi
- ✓ + N3 - Game state recognisable on screen
- ✓ + N4 - Visual feedback to players' actions
- ✓ + N5 - Communication without lagging
- ✓ + N6 - Short tutorial with game rules

Figure 2.1: Requirements

2.5.1 Functional requirements specification

To ensure basic functionality of the game, there are several requirements that must be fulfilled. The design ought to solve these following problems:

1. **F1 – Connecting players to the game.** Players can connect to the game via their own mobile device.
2. **F2 – Handling state changes.** The system will recognise a change of state (i.e. when player leaves the game) and will update the state on all related devices.
3. **F3 – Dividing players into teams.** The system will divide connected players into one or more teams and will consistently keep them in their teams.
4. **F4 – Acquiring and resolving players input.** The system will react to player's input and will update state on all related devices.
5. **F5 – Moderator can switch between scenes.** Moderator will have the option to switch between scenes and reset or cancel a game.
6. **F6 – Logging players' information.** The system will log information about the user devices and behaviour. This logging will be performed once right after connection to the game. Each new game will have it's own log file.
7. **F7 – Updating the game play.** Updates will appear on all related devices.
8. **F8 – Displaying game statistics.** Team statistics will be on display at all times during the game play.
9. **F9 – Disconnecting players.** After the game ends, all players will be disconnected.

2.5.2 Non-functional requirements specification

Along with the functional requirements, there is another set of requirements that must be accomplished to make the game usable. These requirements are not focused on the specific functions as much as on the used technologies and the graphical user interface design. These non-functional requirements ensure the prototype usability:

1. **N1 – Controller accessible via web browser.** All players can connect to the game and further control the game on their own mobile devices via web browser.
2. **N2 – Communication will be implemented on local Wi-Fi.** All devices will be connected through local network.
3. **N3 – Players recognise the game play states.** A change of a state will be recognisable for the player on either the controller, the central screen, or both.
4. **N4 – Feedback to players' actions.** Players will be visually (at the least) notified of their actions.
5. **N5 – Instant feedback.** Communication between devices will be without lengthy lag if possible.
6. **N6 – Game has a simple tutorial.** The tutorial will briefly demonstrate the game rules.

Initial Architecture

The requirements imply that the architecture will consist of at least three different independent agents – the players, the moderator and the control system – communicating with each other through a local network. See the initial architecture in figure 2.2.

2.6 Detailed gameplay description

This section describes the rules of the game *NUMBERS* in greater detail.

The central screen will display one number, sometimes referred to as problem in the text, per team, moving across the screen from one edge towards the other. Each player in team has a set of numbers on their controllers from which they can choose just one. When the sum of chosen numbers by players matches the number on screen, the number will disappear and the team will acquire points. Once a number moves off screen, or is completed by the team, a new problem is generated. The level of difficulty can be set by the moderator. The highest difficulty deduces points from team for each failed problem. The lowest difficulty will not deduce any points and the problem will remain on screen until completion. The game will have three rounds, each lasting a given amount of time, set by the moderator. The players' answers are visible

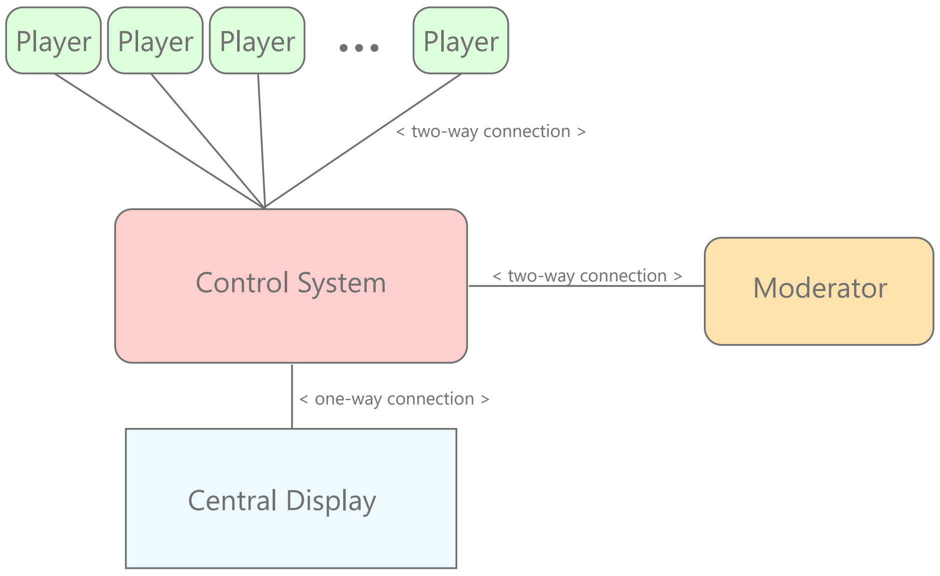


Figure 2.2: Initial architecture

on the main screen, along with the current team sum. After each round a leaderboard will be displayed, showing which team is in the lead. After all rounds are finished, a winning team is announced.

2.7 Actors and Roles

This section describes the type of users, shown in figure 2.3, who will interact with the game's distributed user interface.

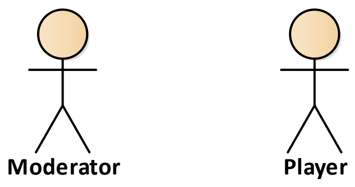


Figure 2.3: Actors

2.7.1 Moderator

The role of a moderator has shown to be crucial for local multiplayer games with a larger number of players. A good game design offers intuitive GUI that does not need a further explanation, and while having an additional person at hand to explain all the rules is beneficial, it is not a primary role of the moderator to be a support character.

Responsibilities: The moderator is responsible for the application setup. He or she must run all the required programs and set the environment for each new game. It is the moderator who decides and then sets the number of players that can join the game and the number of teams the players will be divided into. Moderator also must assess the group of players and set the game difficulty and duration for each round. In addition the moderator switches between game states that are not necessarily consecutive and would require the players to all communicate with each other.

2.7.2 Player

Player is a person, who has a conscious interest in joining the cooperative game. They can join and participate in the game, or leave it.

Responsibilities: Since the game *NUMBERS* requires players to have their own mobile devices (smartphones, tablets or even laptops), the players must each connect their device to the game on their own. Once the player is connected to the game, they go through the process of being divided into teams and learning the game rules. Once the game play has started, the players sorted to one team must cooperate and each choose a number on their devices. Once a game is finished the player should disconnect their device from the local network.

2.8 Use Cases

This section contains a detailed description of processes the actors have to complete in order to participate in the game. The described use cases are illustrated in figure 2.4 in relation to the actors responsible for their completion.

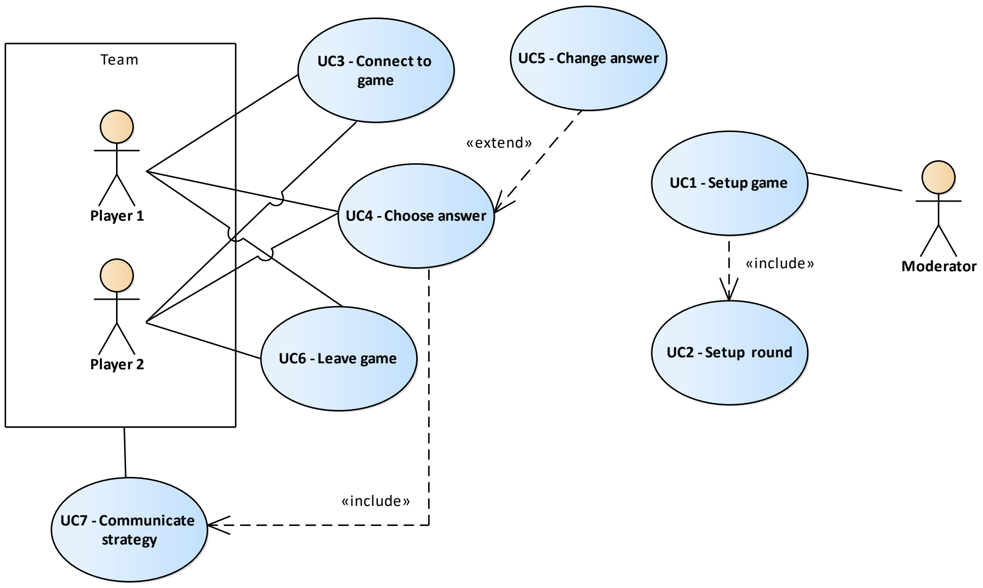


Figure 2.4: Actor's use-cases related utilizing the distributed game

2.8.1 UC1 – Setup game

Basic Path: Setup game

1. Moderator connects to the game via web browser.
2. Moderator sets the welcome message for players and instructions for connection.
3. Moderator sets the maximum number of players.
4. Moderator sets the number of teams.
5. Moderator starts the game.
6. If players connect to the game, then: <include> UC2 – Setup round.
7. After each round have ended and leader board is shown, moderator chooses to switch to a new round setting or proceed to the game finish.

2.8.2 UC2 – Setup round

Basic Path: Setup round

1. Moderator sets round duration.
2. Moderator sets round difficulty.
3. Optional: Moderator launches the tutorial.
4. Moderator starts round.

2.8.3 UC3 – Connect to game

Basic Path: Connect to game

1. Player identifies the instruction on central screen to connect his device to the game.
2. Player connects the mobile to local network.
3. Player opens the controller via web browser.
4. Player chooses to join the game and confirms the action on the device.
5. Player identifies that they have been connected to the game on the central screen.
6. Player is now connected to the game and can wait to be assigned to a team.
7. Moderator sends request to allocate the teams.
8. Player is now a part of a team and is waiting for the round to start.

Alternate: Connect to game

1. If the game has not been setup or is already in progress, this script starts in between steps 3 and 4 of the Basic Path.
2. Player is not allowed to join the game and is notified to try later.
3. Player's device is disconnected from the game.

2.8.4 UC4 – Choose answer

Basic Path: Choose answer

1. Player identifies problem belonging to his team on central screen.
2. Player identifies own set of possible answers on device screen.
3. If player cannot or does not want to solve problem alone then: <include> UC7 – Communicate strategy.
4. Player chooses an answer and confirms it on the device screen.
5. Player waits for other teammates to choose their answers.
6. Player identifies on the screen if the team was successful in solving the problem.

7. If the team has not been successful then: <extend> UC5 – Change answer.
8. If the team has been successful, the player can see updated team score on the screen.

2.8.5 UC5 – Change answer

Basic Path: Change answer

1. Player identifies that their answer needs to be changed.
2. Player chooses a different answer and confirms it on device screen.

2.8.6 UC6 – Leave game

A player may disconnect from the game at any point. Basic Path: Leave game

1. Player closes web browser window with the game on the device.
2. Player disconnects from the local network.

2.8.7 UC7 – Communicate strategy

Basic Path: Communicate strategy

1. Player identifies his teammates.
2. Player communicates the strategy to solve a problem with the team.

2.9 Game Flow

This section describes the overall game flow for actors as well as for the control system, along with defining the behavior for non-standard situations such as player leaving a game during game play or moderator resetting a round or the whole game. The use cases provided a set of actions coming from the actor, that the control system must assess and then change its state accordingly – which means the game flow is by design very action-response driven.

Standard successful game

Connecting the individual standard use–case paths together results in a base for one complete game.

After a game is started by the moderator, players are allowed to connect to the game. First player to connect will cause a change of state, and the game will shift from the welcome board to scene indicating the waiting for other players to connect. Each connected player can choose to stop waiting on other players, in which case he will not be joining the game. Each connection is logged by the control system as well as shown on the moderator's controller, that way the moderator can decide if he should create the teams or wait for more players to connect. Once enough players are connected and waiting, the moderator will give a command for the control system to sort players into teams. Alternatively, players who earlier indicated their team preference will be sorted accordingly. After the players are sorted, the moderator can set the round. The round set-up consists of duration and difficulty settings. Players are meanwhile waiting for the round to start. The moderator then has a choice to either play a tutorial or start the round.

During the game play, each player has a choice between multiple answers – in the case of the game *NUMBERS* a choice between numbers from 0 to 9. As a part of a team, the players must realise what is the complete anwer they are looking for, in this case, the team number appearing on the central screen. Subsequently the players in each team should communicate, how they will reach the complete answer and choose their answers accordingly. Their complete answer as well as their individual answers are shown on the central screen. If the team succeeds in matching their complete answer to their number on screen, the team will score points and the problem will disappear. After each success the players' answers will be reset to the default value, which is 0. If the team did not succeed, the players can change their answers. Players may leave the game, but cannot join again until a new game is started. After a round is finished, the control system changes state either back to waiting for moderator's setting or, if all rounds have been completed, to the final state, which displays the winning team and subsequently disconnects all players. After the game is finished, the control system resets all saved data and settings and returns to its inital state, waiting for the moderator to set a new game with a new set of teams and players. The game processes are best shown in figure 2.5 on page 28.

Round reset

At any point during the game play, the moderator can choose to restart the round. This situation may come up if the players were not prepared for the round to start or the time limit or difficulty need to be re-adjusted. This event will reset all team scores gain in the round as well as the players answer. The control system will set itself and the players back to waiting for the moderator to setup and start a new round.

Game reset

The moderator may choose to cancel the game at any point after the game has been set. This action will disconnect all players and reset all of the game settings except for the control system - moderator connection. The control system will then wait again for the moderator to send new game settings.

Player disconnection

A player may disconnect from the game at any point.

- **Player leaves a game before the teams were assigned** The control system and moderator are notified, but no other action is needed.
- **Player leaves a game after the teams were assigned** The control system and moderator are notified. Team size is decreased by one. A game can only be played if all team sizes are greater than zero, if that is not the case, the game must be cancelled as the number of teams cannot be changed dynamically and is preset for each game. If a team player disconnects during a game play, the other teammates will have their answers and their collective answer reset. Their curent problem on the central screen will be replaced with a new one with no penalisation.

2.10 Domain model

There are several independent entities in the game *NUMBERS*, some of which contain attributes that need to be synchronized between several independent applications. Below is a list of the most significant entities and their roles:

- **Moderator** The moderator is the main driving force behind most of the actions leading up to a successful game. He or she must first configure the game and then each round. Moderator also oversees the action of each player.

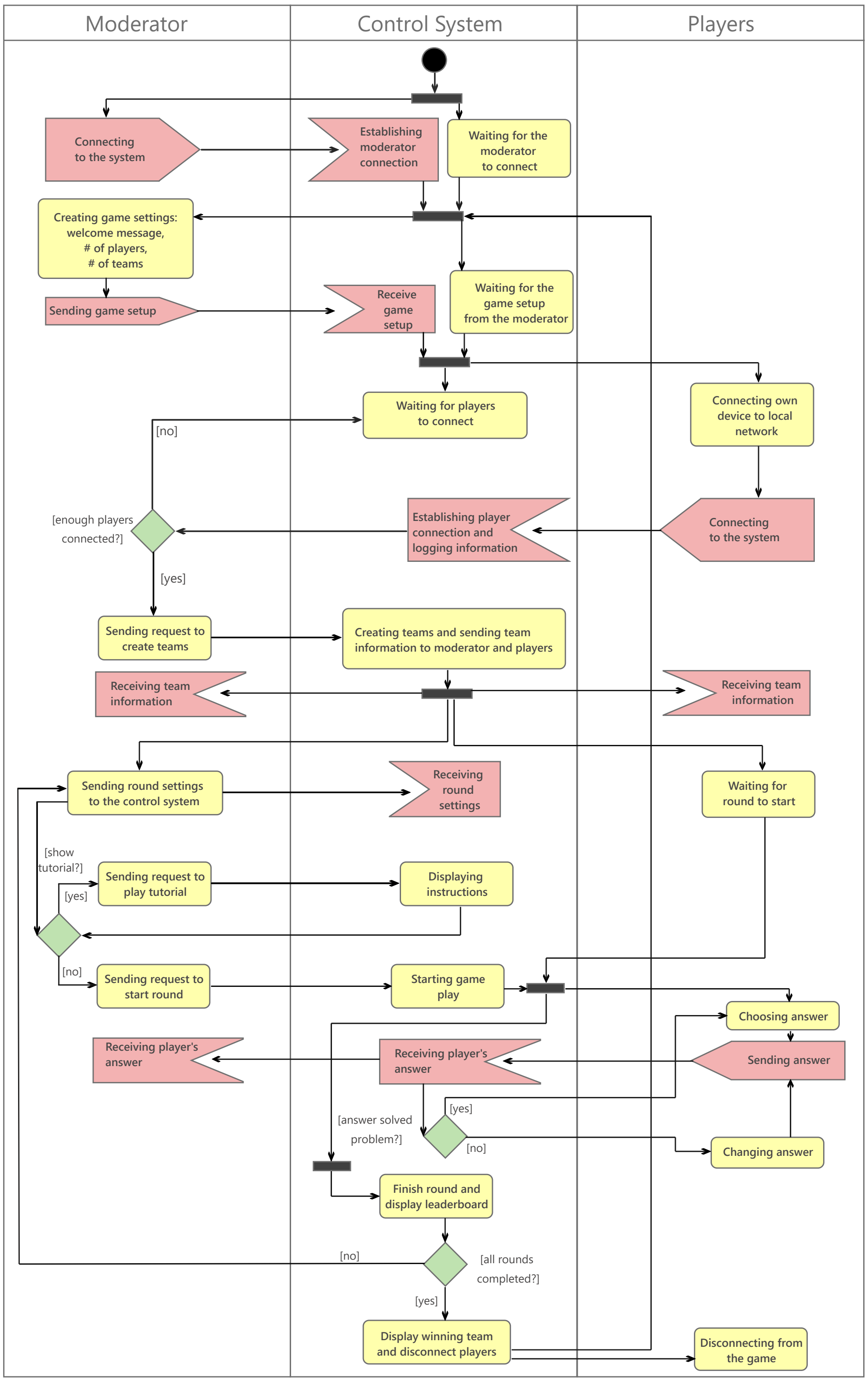

Moderator
Control System
Players
Connecting to the system
Establishing moderator connection
Waiting for the moderator to connect
Creating game settings: welcome message, # of players, # of teams
Sending game setup
Receive game setup
Waiting for the game setup from the moderator
Waiting for players to connect
Connecting own device to local network
[no]
[enough players connected?]
Establishing player connection and logging information
Connecting to the system
[yes]
Sending request to create teams
Creating teams and sending team information to moderator and players
Receiving team information
Receiving team information
Sending round settings to the control system
Receiving round settings
Waiting for round to start
[show tutorial?]
[yes]
Sending request to play tutorial
Displaying instructions
[no]
Sending request to start round
Starting game play
Choosing answer
Receiving player's answer
Receiving player's answer
Sending answer
[yes]
[answer solved problem?]
[no]
Changing answer
Finish round and display leaderboard
[no]
[all rounds completed?]
[yes]
Display winning team and disconnect players
Disconnecting from the game

Figure 2.5: Standard game flow

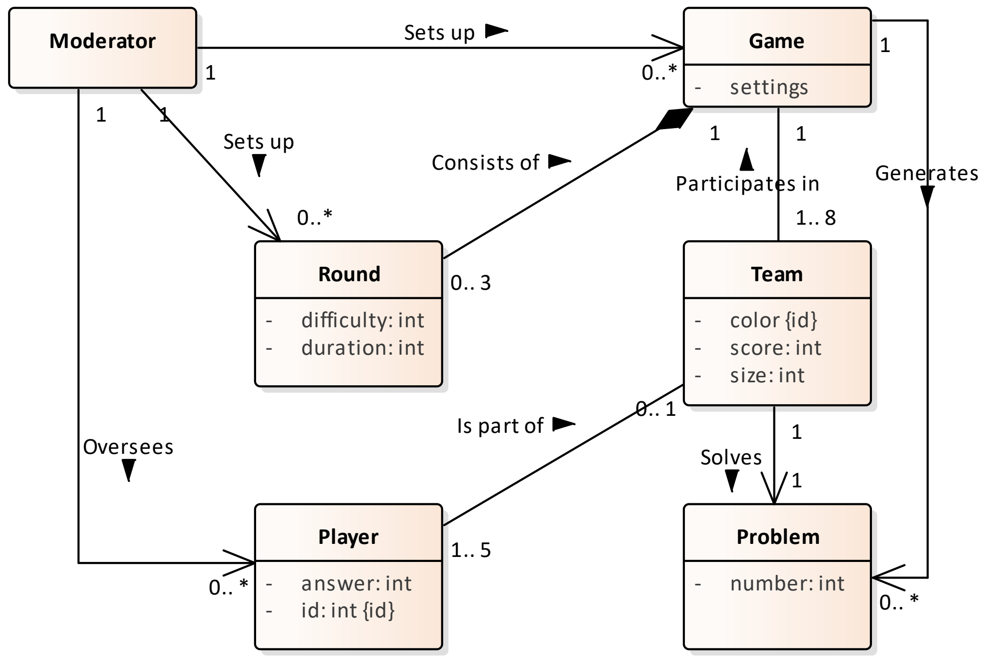


Figure 2.6: Simple domain diagram of the game *NUMBERS*

- **Game** Each game has its initial configuration which determines the number of players and teams. Each game has a set of teams that participate in it and solve the problems the game generates in each round.
- **Round** Each round has a duration and difficulty set by the moderator. Every game must finish three rounds before its completion.
- **Player** Every player connected to the game has its own unique identifier – by which is recognized on the moderator's controller. Once the player is assigned to a team and round is started, the player will participate in the game by choosing an answer.
- **Team** The team entity gathers answers from its players and keeps updated information about the number of players it contains and the score.
- **Problem** Once a round is started, each team is assigned a problem to solve. The problem is a number that the team must match to solve.

The entities and their relations are shown in figure 2.6.

2.11 State Machine Diagram

The described processes and game flow resulted in a state machine diagram in figure 2.7. This diagram is a base structure for the latter design of a prototype of the game *NUMBERS*, but can be re-used for other cooperative games as the principles of establishing connections, starting a game, gathering players' input and ending a game are universal. It is clear from the diagram that the application operates in two main loops. The first loop goes through states 1 – 6 and presents one complete game, after each such loop the game can start with a different set of teams and players, but the moderator stays connected. The second smallest loop goes through states 3 – 5 and represents a game round, each of which keeps the teams and players, but can have different difficulty and duration.

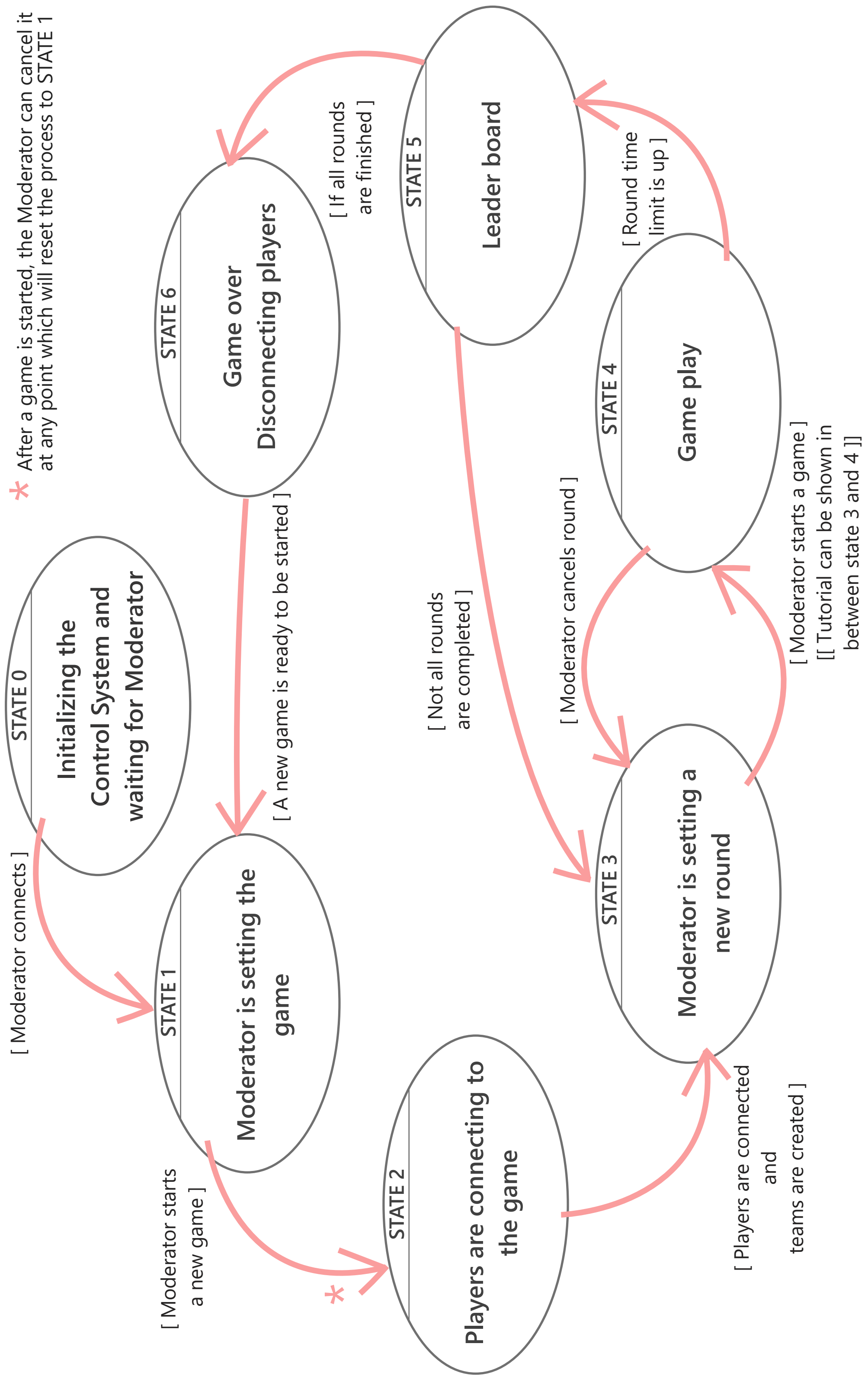

STATE 0
Initializing the Control System and waiting for Moderator
[Moderator connects]
STATE 1
Moderator is setting the game
[Moderator starts a new game]
*
STATE 2
Players are connecting to the game
[Players are connected and teams are created]
STATE 3
Moderator is setting a new round
[Moderator starts a game]
[[Tutorial can be shown in between state 3 and 4]]
STATE 4
Game play
[Moderator cancels round]
[Round time limit is up]
STATE 5
Leader board
[Not all rounds are completed]
[If all rounds are finished]
STATE 6
Game over
Disconnecting players
[A new game is ready to be started]
* After a game is started, the Moderator can cancel it at any point which will reset the process to STATE 1

Figure 2.7: State Machine

Chapter 3

Technologies for prototype development

This section deals with the analysis and selection of technologies and tools fitting the requirements of a local distributed cooperative game for chosen immersive environment.

3.1 Immersive environment

The environment in which the game will be played will have, without a doubt, the greatest impact on the visual form of the game. There are several types of projection rooms and surfaces but some are more immersive than others:

- **Two-dimensional flat screen** While flat projections, such as classroom projections, TVs or movie theater screens, are the least immersive types of environments, they are very accessible and can be found virtually anywhere from households and schools to train stations and shopping malls. The most common monitor and video projector aspect ratios are 4:3 (XGA and SXGA), 16:10 (WXGA and WUXGA) and 16:9 (standard HDTV, 1080p) [24]. Theaters nowadays usually offer 2K or 4K resolution [25], but cluster based systems like SAGE and SAGE2 offer presentation across multiple synchronized displays, reaching even higher resolution[11] [26].

- **CAVE and dome projection** This type of environment undoubtedly offers higher level of engagement as the audience is literally surrounded by the projection. CAVE is a multiprojection system where scenes are projected onto the lateral surface area of a cube-like room [27]. Similiarly immersive to a CAVE projection is a dome projection used in

[11]The system at SAGElab, CTU FIT (https://sagelab.cesnet.cz/en/) consists of 20 FullHD monitors resulting in 9600 x 4320 pixels or 8K+ resolution.

planetaria. While mapping a 2D canvas on a hemisphere is possible either by using a fisheye projection or by warping the 2D canvas prior to projecting, the content should be customized for such environment as the two-dimensional frame must deal with the issues of 360° angle and the need for high resolution of a three-dimensional space [28].

- **Other** There are many more technologies focused on immersive experience, and some, such as VR and AR headsets offer far superior experience than even a dome projection. Because of the fact that the game *NUMBERS* is set to be controlled via smartphones, the author dismissed these technologies for the design of own prototype.

Because the proposed game is set for maximum of forty players who need to be able to use their smartphones, the author decided to develop the prototype for a movie theater like setting, with one flat projection screen.

3.2 Network architecture

There are two[12] [29] main network topologies for online multiplayer games, the peer-to-peer solution and server-client, and both are being currently used for different games.

The peer-to-peer architecture has a few undeniable advantages; for fully distributed fast paced games such as car racing, the response time is faster as each player updates the game state view based on signals received from other players [29].

Although there are systems that benefit from fast response times using peer-to-peer broadcasting, such as the framework MicroPlay, the game *NUMBERS* is, for one, not as fast paced to utilise that feature. Moreover, while the MicroPlay is fully distributed [30], the game *NUMBERS* is a centralized system, revolving around one playfield displayed on one surface.

Considering our application already has a need for control system, the server-client solution is more straightforward solution.

3.3 Platforms

This section proposes few suitable tools for developing a distributed cooperative game and the choice for develpomn *NUMEBRS*.

3.3.1 Game Engines

A game engine provides a set of tools for developing a complete game – from implementing game logic, synchronization and memory management to ani-

[12]The third network architecture is a hybrid between the peer-to-peer and server-client. This type of architecture is used for mass online multiplayer games.

mating and rendering graphics. Most game engines contain these main components: a rendering engine, an audio engine, a physics engine, AI module and the game logic program [31]. The two currently most popular game engines are Unreal Engine 4[13] and Unity3D 5[14]. Both Unity3D 5 and Unreal Engine 4 have API for scripting, support for 2D and 3D game development for multiple platforms and both offer conditionally free versions. Since a prototype of a cooperative game could be developed in either of those engines, the choice of using Unity 5 is based on its simplicity and better documentation.

3.3.2 Server-Client communication

One of the main responsibilities of the server is keeping the clients synchronized. The trouble with a browser client comes with the connection over an HTTP request/response protocol. One way to push data from server to web page client is to use AJAX. AJAX uses the XmlHttpRequest object to periodically send HTTP request to server, this process is known as long-polling. Having multiple clients automatically sending requests to the server, just in case there is a change of state, is rather inefficient and could very well lead to latency and server overloading, depending on the implementation [32]. There are multiple [33] solutions to this problem and one of them is using WebSockets, which enable to send and receive data through a TCP socket.

- **Unity dedicated server** The Unity Game Engine offers its own dedicated server – the Unity Multiplayer[15]. To utilise WebSockets, another option for Unity based server is using a library such as websocket-sharp[16] to implement a websocket server.

- **External server** Another approach would be to use an external server and connect the game engine as a client. This solution would effectively separate the game play logic from handling the players' connections. It would also allow for the game to be run on a different device and have the game updated only in need of changing the content on the central screen. This approach leads to a more plugin-based design and significantly improves the application's potential for future expansions.

After considering both options, having an external server is possibly a more complex feat for the implementation of the prototype, yet it is the better solution for creating a modular expandable software. One of the most popular

[13]https://www.unrealengine.com/en-US/
[14]https://unity3d.com/
[15]https://unity3d.com/unity/features/multiplayer
[16]https://github.com/sta/websocket-sharp

and well documented options for WebSockets is combining Node.js[17] server – along with the Express.js[18] web framework – with the Socket.IO[19] library.

3.3.3 Browser-based controllers

One of the requirements, discussed in section 2.5, suggests the controllers to be accessible via web browser, which rules out the option of using an Unity deployed mobile app as a remote control and demands a web-based solution.

- **JavaScript** This multiplatform, dynamic and objected oriented language is one of the most popular tool for creating interactive web content. JavaScript runs in the browser client – a signifficant distinction over some other scripting languages such as PHP, which run on the server [34].
- **JavaScript Client and WebSockets** Having a Node.js server along with Socket.IO JavaScript lends itself to extending the same tools to developing the players's and moderator's controllers. The Socket.IO client API is analogous to the server API, and uses JavaScript callback functions to react to communication.
- **HTML5** The standard markup language for web pages is used to create structured web-page document. HTML elements, for example canvas, div or paragraph, are structured containers that allow to draw interactive graphics via JavaScript [35].

3.3.4 Connection

Communication protocol Since the clients connect to the server through a web browser, the initial connection takes place over the HTTP protocol [36]. After the initial HTTP handshake the WebSocket connection has been opened and the communication progresses further over its own protocol.

Updated Architecture

Because of the choice to use an external Node.js server instead of Unity Multiplayer dedicated server, the control system from the initial architecture should be divided into two independent agents with a two-way communication. This addition not only separates both the game play mechanics and visualisation from the communication layer but also enables the game engine to only keep data relevant to its current state, without having to synchronize all of the players' events. See the updated architecture in figure 3.1.

[17]https://nodejs.org/en/
[18]https://expressjs.com/
[19]https://socket.io/

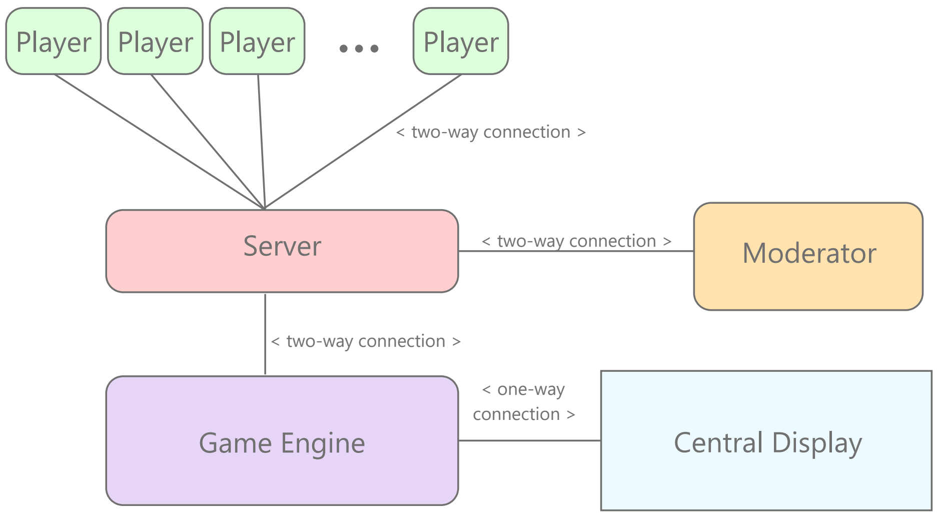


Figure 3.1: Updated architecture: Control system has been separated into server and game engine

Chapter 4

Prototype Design

After analysing several applicable game ideas and deciding on one, the next stage is the formation of a solution. The requirements analysis along with the deduced state machine and the initial design of the architecture, lay the foundation for designing the states of each module: the game engine, player controller, moderator controller and server. After having clear picture of each module's behavior, the communication that will signal a change of state between the clients and the server will be proposed. The remainder of this chapter is dedicated to the design of graphical user interface, data structures and manipulation and the application modularity.

4.1 Scenes and controller screen design

This chapter describes how each state translates into a visual response on screen. The author chose a 2D projection as a demonstrative utility for main projection surface – for that reason all the following storyboards will be in 16:9 ratio.

4.1.1 Main projection

The game engine needs a network manager agent that will be initiated at the beginning of running the program and handle a two-way connection to the server. The connection must remain uninterrupted over the course of the game and can be only closed by terminating the program[20]. The transition between scenes is shown in figure 4.1 along with wireframes of the graphical user interface of each scene.

0. **Initialization state** Game engine initializes its Network Manager and establishes connection to the server. After the connection is established, the game engine can switch scenes and wait for the game setting.

[20]This also implies that the server must be running prior to running the Unity3D program.

1. **Waiting for the game to start** Once the moderator sends the game settings to the server , welcome message, number of teams and expected number of players is passed on to Unity and a new scene is loaded.

2. **Welcome screen** This static screen displays the instructions for connecting a player device to the game via web browser. Once the first player has connected, new scene is loaded.

3. **Waiting for players** The instructions from last scene are still displayed. Unity gets a message from the server anytime a player has connected or disconnected. The number of connected players ready to play is displayed along with the expected number of players. This way the players can collectively see who has connected and how many have yet to be connected. When the server sends a message that the moderator assigned teams the next scene will load. While Unity3D application does not keep an instance of players' connections, once the teams are assigned, the application will keep and update the information about each team until a new game is started.

4. **Round Introduction** Static screen displaying which round is up next and what its duration and difficulty will be. Both of those can be changed by the moderator on request and passed over to the Unity3D application through the server.

5. **Tutorial** Optional scene which is triggered by a request from the moderator to the server. After the scene animating the game play and rules is finished. The game waits for the game to be started by the moderator.

6. **Game Play** The game starts by generating a problem for each team and ends after the round duration elapses. Server keeps updating the team's collective answers and also separate players's answers. Problems, one for each team, will appear on one side of the screen and proceed to roll to the other side. Once a problem rolls off the screen the team has failed to complete it. Team scores 10 points for each solved problem. If the difficulty is set to 1 – the easiest – there is no time limit in which the team must solve the problem, apart of the duration of the round, and the problem will appear again after rolling off the screen. For moderate difficulty, the team will lose 5 points for each time a problem rolls off screen, but problem with the same number as before will appear. The hardest difficulty will have teams lose 5 points for each failed problem and a completely new problem will be generated.

7. **Leader Board** This scene displays the teams in order by their overall score from all rounds combined. If not all rounds have been completed, the scene is switched back to scene 4 on moderator's request.

8. **Game Over** After all three rounds are finished, the game will instead of scene 4 switch to this scene, which will display the winning team.

Teams in *NUMBERS* As mentioned above once a team is allocated the Unity3D application keeps and updates its attributes. Each team must have assigned its size, which is the number of players in the team. Once a round is started, the team entity gathers its current players' answers along with the reached score. While each team will have a name, they must be easily recognizable by distinct colors. Team data is destroyed after the game is finished or cancelled.

Moderator settings in *NUMBERS* Each of the moderator's settings such as the welcome text and round difficulty is saved and updated in the app for the whole duration of one game. Once a game is cancelled or finished, the setting will be reset back to default values.

In the case that the moderator resets a round, the application will return to scene 4. If moderator cancels the game or after the game is finished, the application will return to state 1 waiting for the next game to begin.

4.1.2 Player controller

The player controller is accessible via web-browser on local network. After the player connects to the local network, he or she should be redirected to the controller web-page. Once the page is loaded, connection to the server is established and must be kept through the duration of the game. The player will be disconnected once they close the browser window. The transition between the separate controller screens is visualized in figure 4.2. Below are listed the separate states of the player controller application:

0. **Game is not accessible** If the game is currently in progress or has not been started yet, the controller displays a message to try again later and the connection to the server is interrupted. This screen includes a refresh button which the player can use to try to reconnect.

1. **Joining a game** After the game has been started and setup by the moderator, the screen will show a welcome message along with a single button that will connect the player to the game.

2. **Waiting for other players** Once a player has joined the game, they are assigned a unique player identification number. The number is displayed on the screen at all times until the player disconnects from the game. At this point, the player is connected to the game and is waiting for other players to join him in waiting for the teams to be assigned. He or she can choose to stop waiting, at which point they will not be included in the game once the teams are assigned, and will be redirected to state 0.
   Team selection: Alternatively, the player can signalise preference for one team over the others.

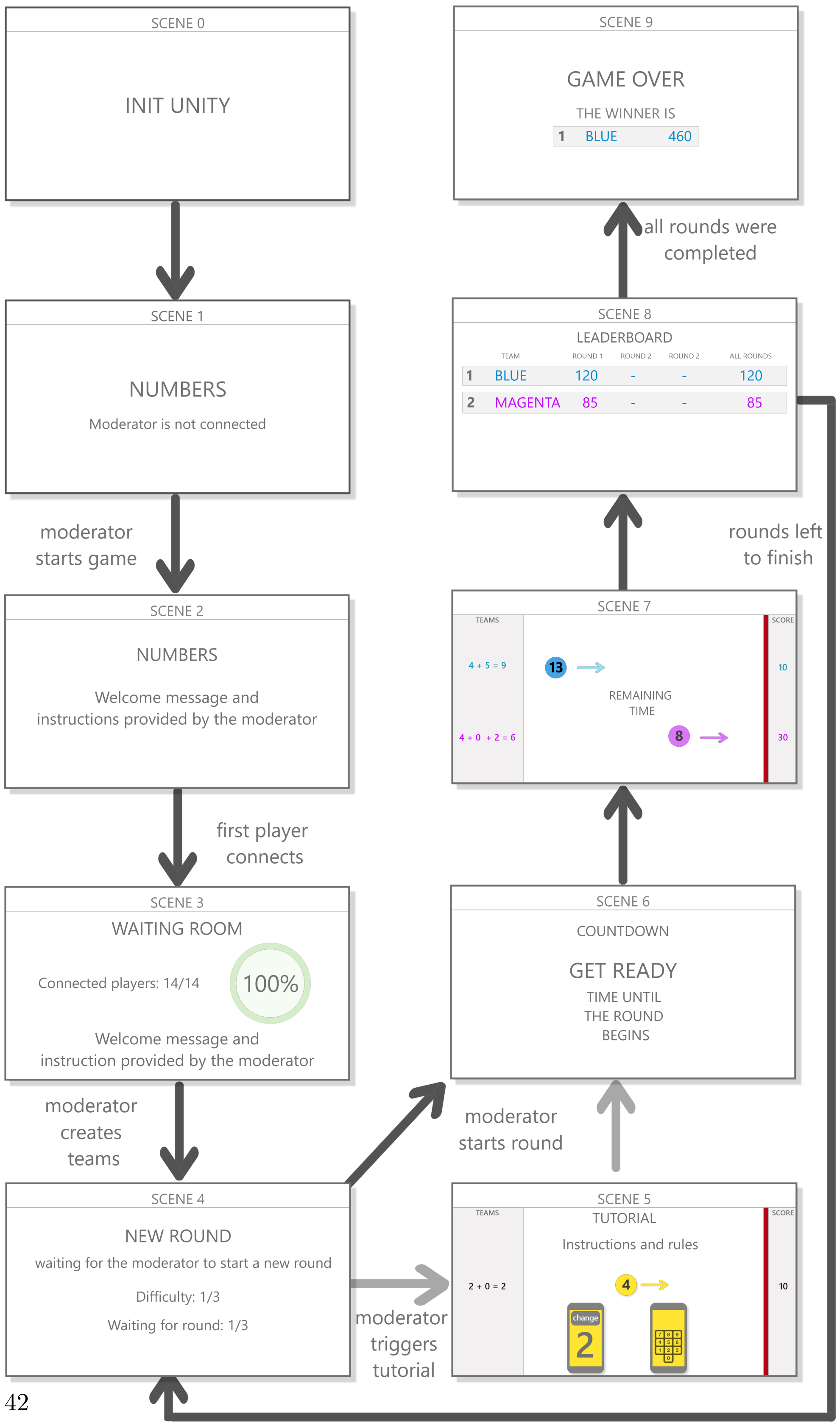


Figure 4.1: Central screen scene designs

3. **Team assignation** After the teams are allocated, the player is notified about the team they have been assigned to. From this point the team color will be at display until the game is over.

4. **Tutorial** When the tutorial runs on the central screen, players are notified to watch the instructions.

5. **Game Play - choice** After each round is started, player has access to multiple possible answers to choose from. By tapping a button with an answer, the answer is sent to the server and then passed on to the Unity3D game application and to the moderator, who can oversee the player's answer at any point in the game play. By choosing an answer the next screen is loaded. If a player later decides to go back and change his answer, the current answer will be highlighted among all other answers.

6. **Game Play - answer** Since player has chosen an answer, the current answer is now visible on screen. The player can choose to change his answer by clicking a button that will take him to the former screen.

7. **Round Over** After each round is finished, players are notified and encouraged to wait for a new round. From this point, the game will progress either to the tutorial screen – screen 4 – or to the game play screen – screen 5.

8. **Game Over** When all rounds are finished, the server will signalise the end of a game so players won't expect another round. The screen contains a notification about the end of a game.

9. **Disconnection** After the game is over, all players are disconnected from the server to allow for new game settings. The screen contains a notification about disconnection from the server.

4.1.3 Moderator controller

The moderator must connect to the server prior to any players as the game stays otherwise inaccessible.

1. **Game Settings** The initial page shows multiple input fields for the game configuration. After the fields are filled in the moderator can start the game – action that will pass the game settings to Unity3D application.

2. **Waiting for players** The controller will notify the moderator each time a player has connected to the game. The page will create a slot for each connected player and display the player's status[21] and unique identifier.

[21]connected or disconnected

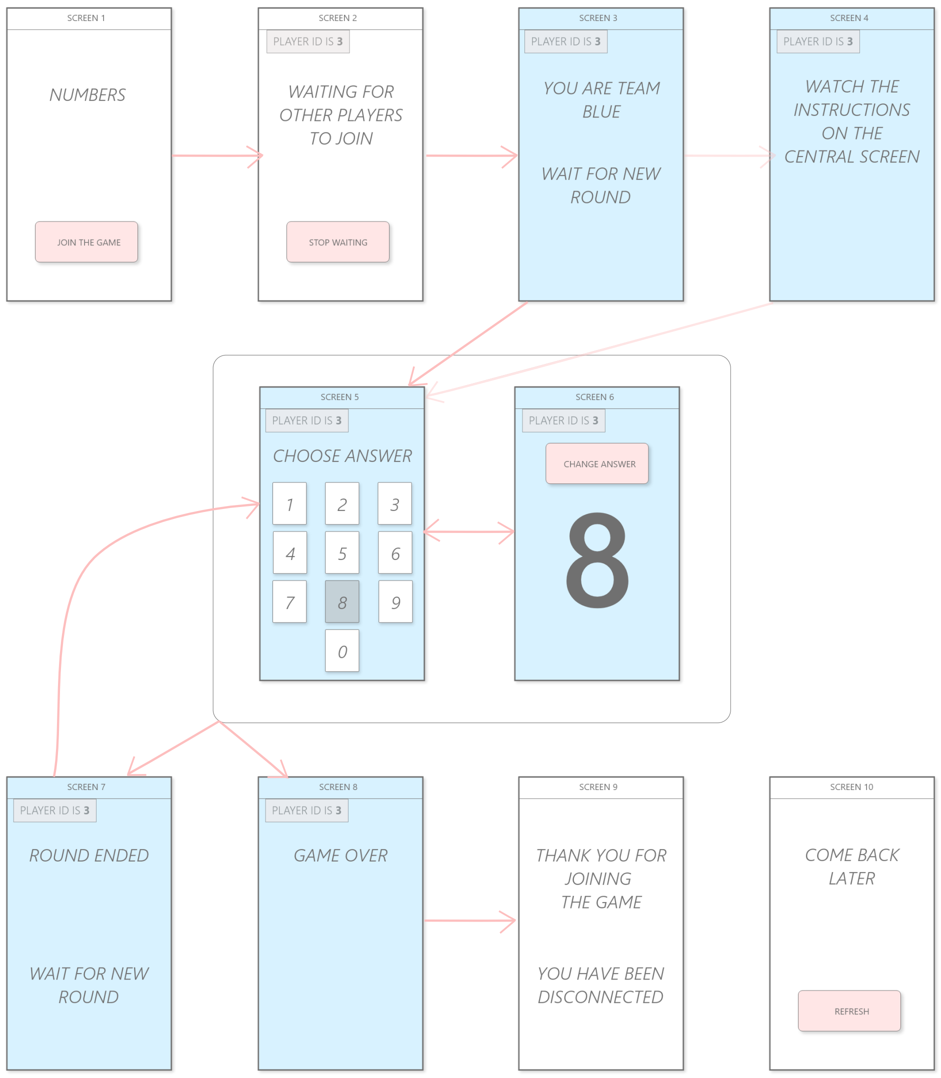


Figure 4.2: Player controller screen wireframes

The moderator can request team allocation by tapping a button, which will also load the next page.

3. **Round Settings** The page will maintain the players' slots and also show input fields for the round settings. Each player slot also displays which team they belong to. The page contains buttons for cancelling the game, showing the tutorial and starting the round.

4. **Game Play** The page further contains a button to cancel the round. Moderator now can oversee each player's answers.

5. **Round Over** After the round is over, the central screen shows the leaderboard until the moderator taps a button to start a new round or finish the game – in case all rounds were completed.

4.2 Connection and communication sequence

The connection between each part of the prototype is crucial to the game. It has been established in section 2.5, that the actors must be interconnected via local network, and later the decision to use external server in addition to the game engine application led to separating the design into four applications. As these applications should be as independent as possible, greater amount of thought must be given to the communication flow used to synchronize and transfer data across the server to and from the clients. It is apparent from the design of the game and the described states of each application, that the communication is very time-sensitive and many events in the game have a firmly set order of succession.

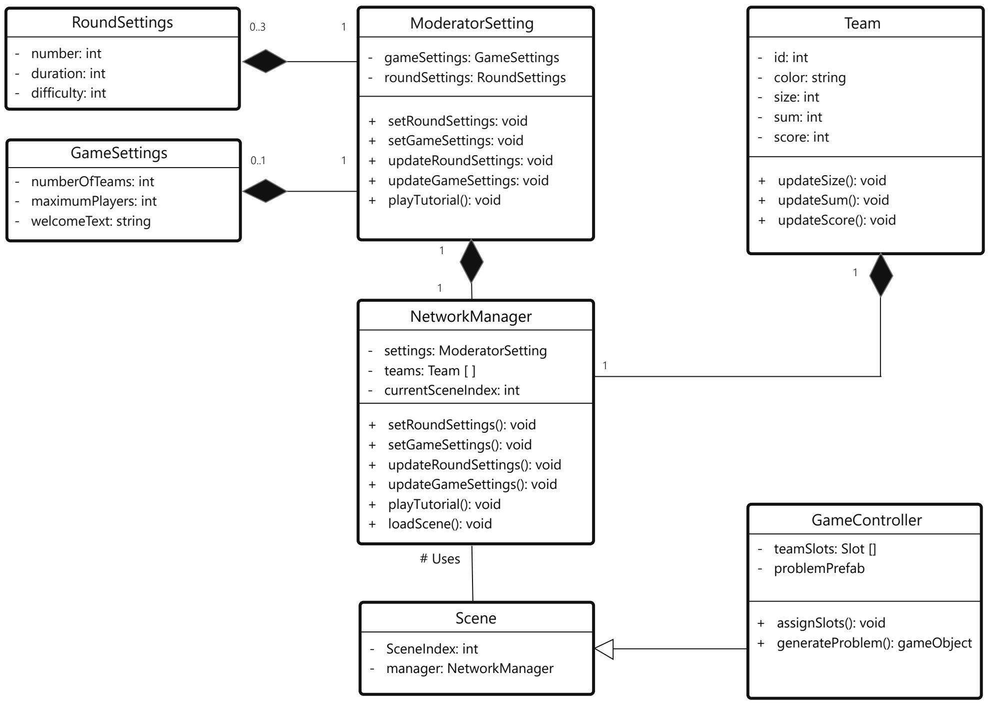


Figure 4.3: The main proposed data structures kept in the Unity3D application

Given that all communication is passed from and to the server, the main responsibility of the server is to correctly evaluate each request and send a fitting response either to the same or different client or clients. The communication flow is shown in figure 4.6.

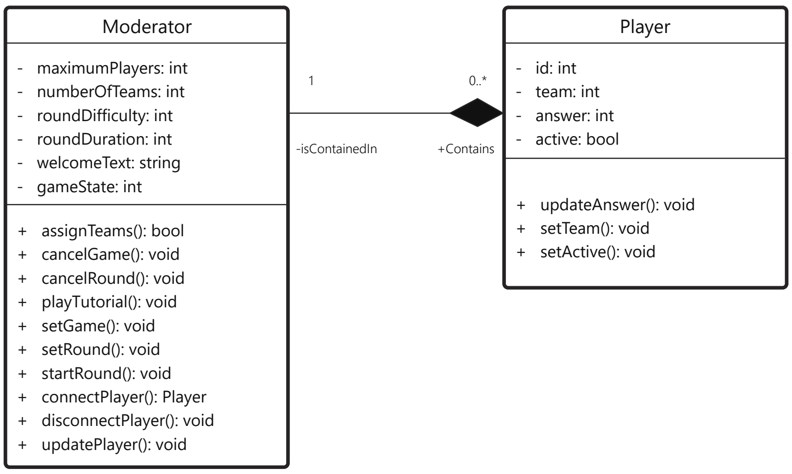


Figure 4.4: Moderator controller keeps data concerning the game settings and connected players

4.3 Class diagrams

Several independent data structures will form the the core of the game *NUMBERS*, some of which will contain data needed to be synchronized between several applications via the proposed communication protocol. The proposed class diagrams for the Unity3D application, moderator controller and server are shown in figures 4.3, 4.4 and 4.5.

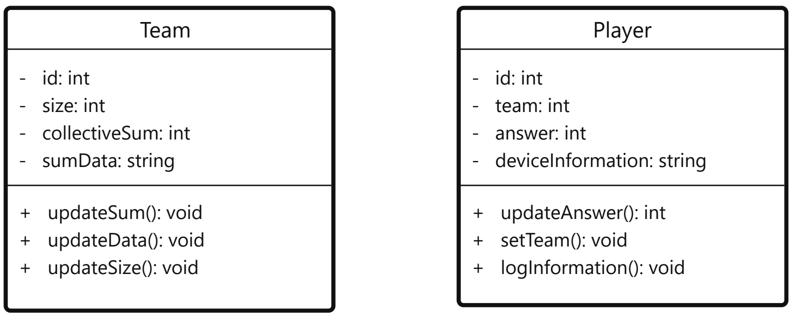


Figure 4.5: Server keeps stored all client connections in addition to storing data about the players and teams

4.4 Graphical user interface

While attractive graphics is not the most crucial part of the prototype as it is a proof of concept and not a fully developed game, certain graphic standards must be taken into consideration. Poorly designed graphical user interface could easily render the prototype impractical, therefore these following ideas will improve the usability of the game.

Playfield The scene is divided into several horizontal slots, one slot for each team. Each team has all its information displayed in this horizontal line. The text and the problem visualization are both in the team's color. Each slot displays information about the team's score and the current collective answer of the team. For better clarity the separate answers of each player may be also displayed, to indicate how each player voted. The timer and round information is displayed on the background of the scene and in the middle of the screen. For more information about the design, please see wireframes in figure 2.5. Each of the team's problems must be visible at all times, and once a team succeeds in solving the problem, the event shall be visually and audibly indicated.

Player controller The user interface on the client device must be reasonably responsible to accommodate different types of mobile devices, but is primarily meant to be accessed through smartphone. Another key feature is visual similarity to the central screen in font use and color palette, as the controller should clearly indicate that it is an extension of the course of events happening on the central screen.

Moderator controller The controller must be able to effectively display all players's information on one screen, without requiring the moderator to scroll or list another page.

4.5 Logging and data gathering

From the moment the player is connected, the server can gather and save his interactions or any other type of information the player chooses to share. Now, this game does not require for the players to submit their personal information such as age, gender or favorite color, yet even logging and subsequently analyzing the player's behavior during the game may offer valuable information. Among others player's behaviour logging and analysis may give information about the game play – for example who is the most active player, what is the strategy the team chose, how much time it took per average to solve a problem – but also about the prototype itself – How many players were idle during the first round? Were they more active in the second? How many player's decided to leave the game before it was over? Were they in the currently losing team? While many of those questions could be answered during a lengthy testing by asking the players, all of those could be also answered by an extensive behaviour logging and proper analysis.

The simplest way to monitor player's behaviour is simply logging the traffic incoming from the client to the server.

4.6 Modularity

Each application connected to the server must be independent, working only with the data from user's input or accessed via the server. That way future changes and improvement can be applied to just one part, without affecting the whole system. This principle should be applied also to the two main application layers in Unity3D project – the communication layer and the visualisation layer.

While the above described application states were created for the game *NUMBERS*, the architecture and communication protocol are very well applicable to other games. The only game-specific data transfer between the game engine and the server are the players' answers. The essence of the problem to be solved is present only in the Game Controller layer of the Unity application and nowhere else.

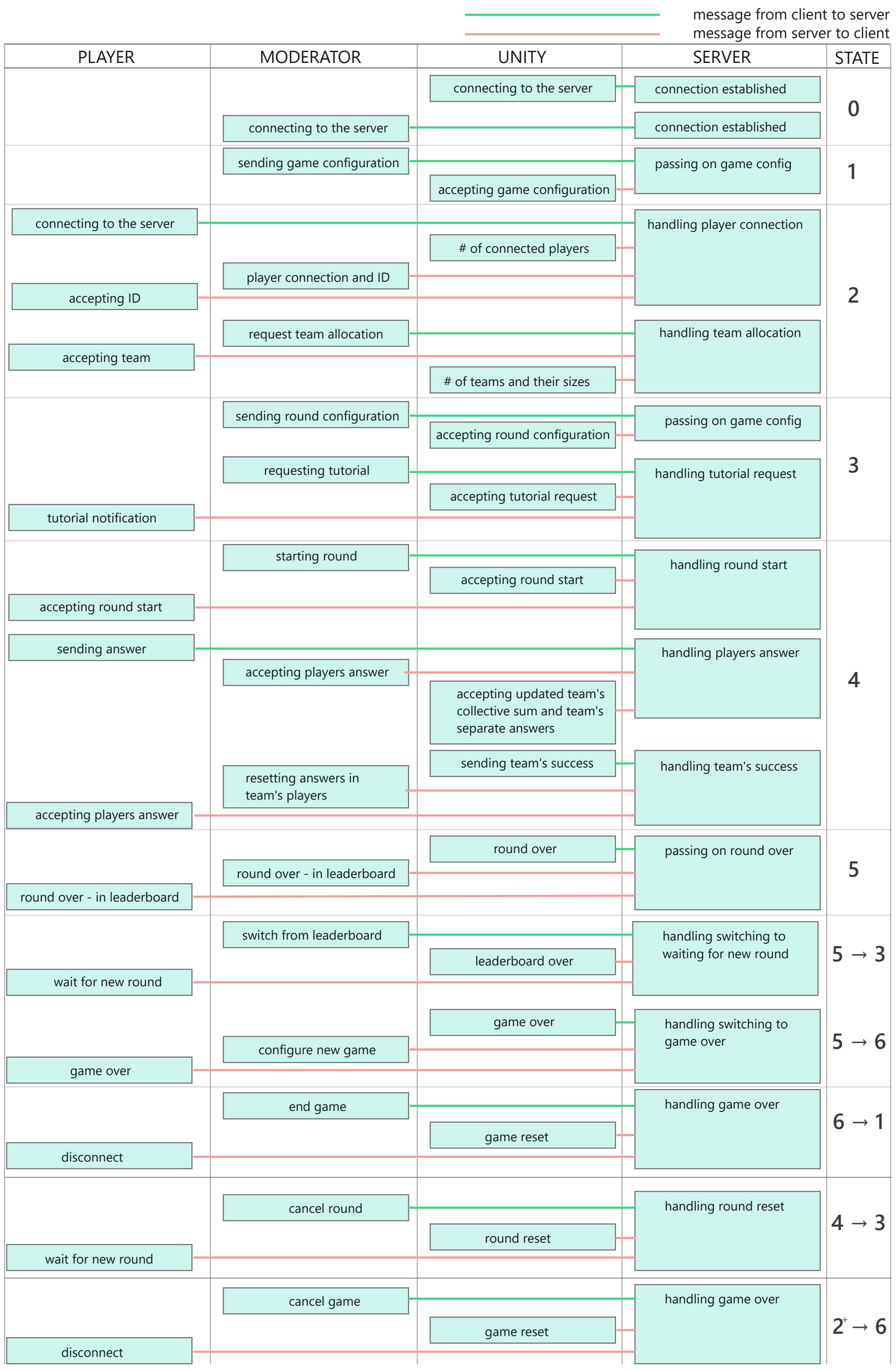


Figure 4.6: Communication flow aligned with states from figure 2.7

Chapter 5

Realisation

This chapter is focused on the complete realisation of a prototype of the local cooperative game *NUMBERS*. The first section is dedicated to the overall structure of the project, while the second describes the most important parts of implementing each application. The chapter further describes the main data structures and issues that came across during the process of implementation. The final section addresses the process of installation and setup in order to successfully launch the prototype.

5.1 Project structure

The prototype consists of 4 separate applications – the server, two web based applications, referred to as *PlayerController* and *ModController*, and one Unity3D game application, hereafter referred to as the *UnityApplication*.

Server application The server is operated from a single JavaScript file called `server.js` in the root of the web folder. The gathered data are stored in the log folder.

PlayerController The application consists of a JavaScript file `sketch.js` with a commented set of functions handling the connection to the server and interactivity. Predefined variables along with the IP address and port are kept in the file `resources.js`. The files are linked together along with two CSS files, ensuring responsible design, via the `index.html` file.

ModController Similarly to the *PlayerController* the logic and interactivity is largely kept in the `mod.js` file in the Mod folder. The predefine variables are again kept in `resources.js` file. Other files contain declarations of data structure functions and HTML elements.

UnityApplication The Unity3D project consists of several files needed for the project to be opened in the Unity3D editor, but all the user-made files are stored in the Assets folder. The ten separate scenes are kept in the folder _Scenes within the Assets folder. All the C# script the author made are stored within the _Scripts folder, regardless of the scene or game object they are linked to. The font family Orbitron[22] that the author used for all the text within the game scenes is stored in the folder Fonts. The Libs folder contains the imported `.dll` files enabling the use of the *socket.io-unity* library used to establish a WebSocket connection between the *UnityApplication* and the server. Pre-made game objects are stored in the Prefabs folder and explosion effects are in a folder named Smoke. All created and subsequently imported textures are in the Textures folder while all of the materials that are real-time loaded during the game play must be stored in the Resources folder.

5.2 Implementation

This section describes the most significant parts of implementing each part of the distributed game.

5.2.1 Server and controllers

The server application handles the incoming and outcoming messages from *UnityApplication*, *ModController* and multiple *PlayerControllers.* To establish the connections, the author setup the server to listen on three separate ports, each for one type of client. As both *UnityApplication* and *ModController* connections are kept in single variable, the server only allows for one connection from each application. Communication with client applications was realized through setting a different event listener to each port and subsequently storing each connection through a Socket.IO `socket` object for later use. The server keeps the current state in the `gameState` variable used to synchronize the state on the client devices.

Data log on server While the design proposes extensive player behavior logging, the prototype implements the general idea of keeping a set of information about each connected player. Every time a player connects to the game, the server logs information about the device accessed via the user-agent header presented in the HTTP protocol [37]. The logged device information consist of the device's IP address, web browser and operational system. The server logs each game to a different file and utilizes the *useragent-parser-js*[23] package to parse the data from an HTTP request.

PlayerController The controller was implemented to mimic the design proposed in the wireframes in 4.2. The connection to the server is realized through Socket.IO Client API and kept through the `socket` variable.

ModController The controller was implemented to accommodate each state as described in section 4.1.3 of the Design chapter. The application keeps data about the players and their actions. The connection is realized through Socket.IO Client API.

[22]https://fonts.google.com/specimen/Orbitron
[23]https://www.npmjs.com/package/useragent-parser-js

5.2.2 Unity project

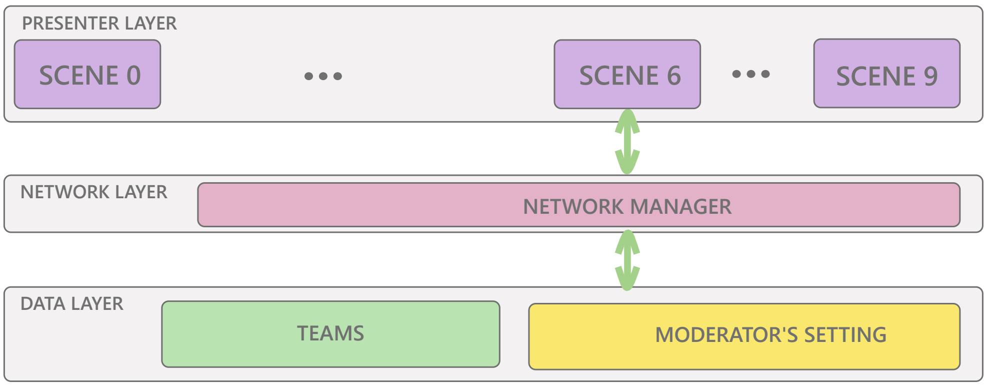


Figure 5.1: Three layers of the *UnityApplication*.

The work on the game engine side of the prototype lies in the realisation of a stable two-way socket connection to the server and the visualisation of the 10 scenes seen in figure 4.1. The *UnityApplication* could be divided into 3 structural layers: network layer, data layer and presenter layer. Each scene has own script that displays or animates its contents. After one scene is loaded, all the objects, settings and data stored in the previous scene are lost. For that reason, the data that needs to be kept across multiple scenes ought to be stored in a lower layer as seen in figure 5.1.

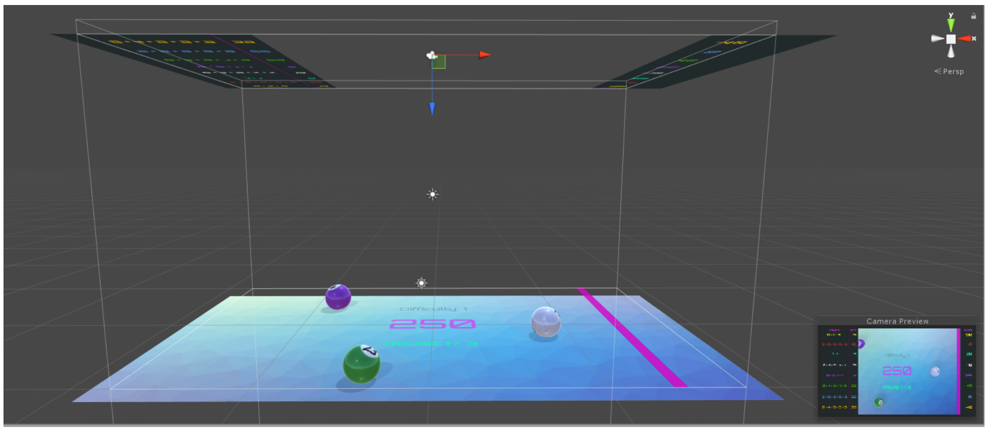


Figure 5.2: Creating scenes in the Unity3D editor.

Two-way socket connection The connection between the server and the `NetworkManager` class[24] was implemented using the *socket.io-unity* [38]

[24]The class can be found within the _Scripts folder in the NetworkManager.cs file.

library. The connection is kept during the whole run of the unity project.

Creating scenes in Unity3D Each of the scenes was first created in the Unity Editor. The background canvas is a plane with assigned texture. There is only one ortographic view of the scene, provided by a camera positioned in 90° angle over the background canvas. Each scene is illuminated by three directional lights. All texts and panels were realised by the Unity3D UI Canvas, Text and Image objects.

Problem visualisation There are many ways to show a number on screen in the Unity3D game engine, some more straightforward than other. Since the displayed problem must indicate both number and team color, combining these two into one object became the most practical solution. In deciding what exactly the object should be, pool balls stood up in association as they combine colors, numbers and roll on a given surface. To implement the idea, that the application will generate new problems during the game play, it was necessary to create a prefab[25] asset of the chosen object. As sphere shape is

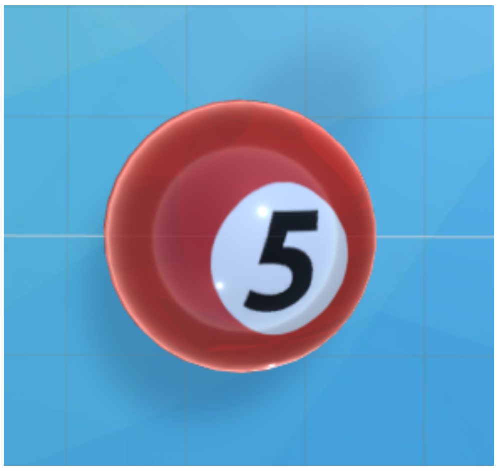


Figure 5.3: Ball prefab: visualisation of a problem on screen.

one of the primitives that can be created directly within the editor, the author did not have to model and import own object. Each visible model in Unity has its Mesh Renderer Component with the Materials attribute [39]. The most effective way to change the numbers and colors of each sphere, was to assign two materials to the Mesh Renderer before the object's instantiation. The first material would be the color of the team and the second, partly translucent, would show the number. Materials can be easily created in the Unity3D Editor by importing a texture and subsequently assigning the texture to a newly created material. Therefore the author prepared set of materials in the Resources folder of the project. Files saved to the Resources folder can be

[25]https://docs.unity3d.com/Manual/Prefabs.html

Figure 5.4: Smoke explosion animation: visual feedback to solving a problem.

subsequently loaded during the game play via script. The author then added several scripts to the sphere GameObject:

- `Mover` This script enables the object to realistically move across the scene, depending on the `speed` variable. The `speed` is dependent on the team size T_{size}, as larger teams need a bit more time for communicating strategies:
$$speed = \frac{(12 - T_{size}) * 0.95}{2.5}$$

- `Random Rotator` Enables the object to rotate by setting its angular velocity vector, the speed of rotating depends on the `tumble` variable.

- `Destroy By Time` Each instantiated object gets destroyed over some time after it rolls off the screen.

- `Destroy By Team` The object recognizes if a team has been successful in matching its number and calls subsequent events such as explosion animation, team score update, ...

Build settings The project was built as a standalone application for Windows platform with x86_64 architecture with the resolution set to 3840x2160 pixels as those are the specifications of the hardware available at Techmania Science Center in Pilsen[26], where the prototype will be subjected to usability and stress testing.

[26]http://techmania.cz/en/

5.3 Data manipulation

The vast majority of data is stored on the side of the server and the *UnityApplication*. The least amount of data is stored in the player's application, as the player must only be aware of the state of the game and their answer.

5.3.1 Data structures in *UnityApplication*

The main two data structures in the Unity3D objects are the class `Team` and the class `ModSettings` as described in chapter 4, figure 4.3. The data are only accessible via the `NetworkManager` class that acts as a controller layer between the scene scripts. Because all of those three classes must remain unaffected by loading different scenes, the script they are contained in must be assigned to a Unity3D game object. The object must be then preserved through all scenes by applying the function `DontDestroyOnLoad(game object)` in the object's initialization.

5.3.2 Data structures on server

The server application keeps three main data structures to successfully store and then redistribute the information. The functions `Player`, `Moderator` and `Team` are identical to the structures proposed in the design chapter, figure 4.5. The team sum on the server is updated by receiving the current answer $P_{current}$ from the player along with their previous answer $P_{previous}$. This way the `Team` object on server does not have to keep information about its players and their answers. The team sum T_{sum} will be:

$$T_{sum} = T_{sum} - P_{previous} + P_{current}$$

The individual player's answers are kept in two ways, in the `Player.number` variable and then also in `Team.sumData` array. The array is updated every time a player of the team changes their answer and passed over to Unity application. That way the `Team` and `Player` units are effectively separated even on the server, with the `Team` information only sent to Unity and the `Player` data communicated to the player devices and moderator.

5.3.3 Data structures – *ModController* and *PlayerController*

The *ModController* stores the connected player's data in the same way as on server and displays them on screen in a grid of HTML div elements, including the players' IDs, teams and current answers, as can be seen in figure 4.4. The *PlayerController* keeps only information about the current game state, the current and last chosen answer and the assigned team.

5.4 Issues and solutions

This section focuses on the challenges that needed to be overcome during the process of implementation.

5.4.1 *UnityApplication* build

The author's initial intention was to build the *UnityApplication* to WebGL platform, and then use a *sage2_unity* library[27] for SAGE2, that would make it possible to display the *UnityApplication* in Electron browser. Unfortunately, the library used for communication with the server fails to emit messages to the server in WebGL build. The library's github page now has multiple issues dedicated to this problem. One user offered the solution of switching the .NET version[28], which in this case did not help to solve the issue.

5.4.2 Matching colors and fonts

The first group of testers that helped to assess the prototype noticed, that the color of the balls visualizing the problems in some cases did not match the general team color shown on client's device and in some cases was too similar to other team's color. This issue was fixed by taking the same background texture used on the client's devices to indicate teams and turning it into a material for the ball prefab. See the difference in figure 5.5. Other issue the testers noticed was a difficult readability of white text on the team background for one of the teams. This issue was handled by changing the font color to darker shade for players with teams of lighter color.

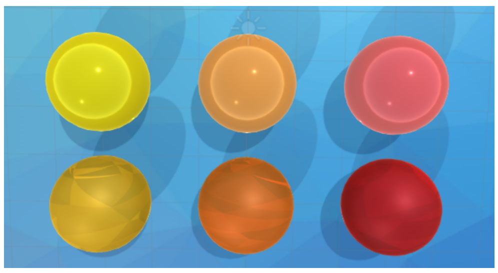


Figure 5.5: Difference between team colors: in upper row are materials before change.

[27]https://bitbucket.org/sage2/sage2_unity
[28]https://github.com/floatinghotpot/socket.io-unity/issues/20

5.5 Installation and configuration

The game *NUMBERS* requires to have installed *Node.js* on the machine that runs the server application. The machine that runs server must be the same one the moderator will use to access the controller.[29]

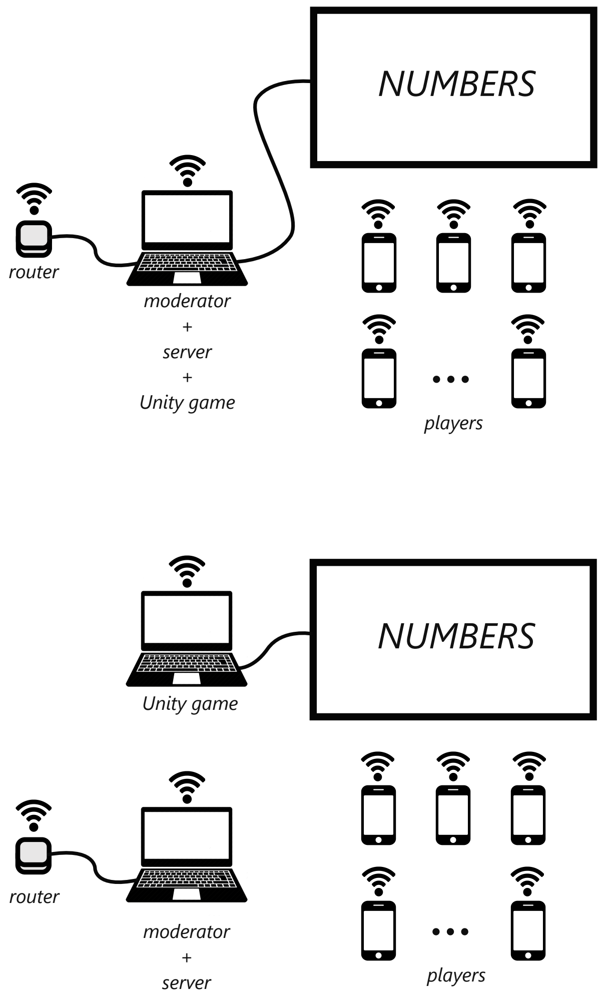


Figure 5.6: Two examples of game setup.

The standalone Unity3D application can run on the same or different machine. Before launching the applications or controllers, all devices must be connected to a local Wi-Fi network, with the IP address on the machine that runs the server set to 192.168.0.101. The author used the TP-Link TL-WR902AC portable router with network set up on 2.4 GHz wireless frequencies for all

[29]This can be easily changed, but the feature of configuring the input points is not yet implemented in the application.

testing and development. The application setup is described in figure 5.6. More detailed installation and setup guide is provided in the attachments.

Results The interface consist of two independent applications and controllers for the moderator and players. Realized prototype of the game *NUMBERS* implements all of the functional and non-functional requirements from section 2.5 and is prepared for usability testing and evaluation. Screenshots from conducted testings, showcasing the realized prototype and GUI, are enclosed in appendix C on page 79.

Chapter 6

Prototype Evaluation

Testing is a vital part of any application development as it provides useful information about the product's quality. In this chapter the author describes the testing process of the realised prototype of the game *NUMBERS* and findings concluded during this process.

6.1 Testing during implementation

The prototype has been tested in iterations all through the process of implementation in a smaller setting.[30] As the development was processed on personal PC, the author could not test the prototype on more than 5 devices[31] at the same time, thus it has been only possible to ensure basic functional testing.

The testings during the implementation helped to uncover and fix several problems. One of the problems had been an issue with generating the problem game objects. If a team solved one problem so quickly, that the object's mesh was still in the area a new problem would appear in, the new problem would be triggered by the other object and it's trajectory would change accordingly, leading to objects flying out of scene. This issue has been solved by disabling the game object's mesh trigger for until the object passed half of the screen distance.

6.2 Usability testing

According to [40], a product is usable when *"the user can do what he or she wants to do the way he or she expects to be able to do it, without hindrance, hesitation, or questions."* Because only a person, who has not developed

[30]One PC unit running the server, *UnityApplication* and *ModController* and 2–4 connected smartphones.

[31]One extra computer and 3 smartphones.

the final product, can approach an application without bias, it is necessary to conduct usability testing with testers that have not been involved in the process of realisation.

6.2.1 Test scenario and enclosed questionnaires

This section describes the additional tools used to gather feedback from testers.

Introductory questionnaire The testers were first asked to fill an introductory questionnaire focused mainly on their ability to use a web browser on their mobile device and the likeness of having them spontaneously connect to a game that would require an installation of additional application on their device. The questions are listed below:

1. Do you own a smartphone or a tablet? {Yes/No}
2. Can you connect mobile device to secured Wi-Fi? {Yes/No}
3. Which web browser do you use on your device?
4. Are you allowed to install mobile applications on your device? Do you know how? {Yes/No}
5. Would you install an application on your device just to play one game at a party event? {Yes/No}
6. Are you able to fill in URL address to your web browser? {Yes/No}
7. Would you join a cooperative game, if it would require playing with strangers? {Yes/No}

Test scenario To assess the prototype's usability, it was necessary to see the testers interact with the game and carefully observe whether they had any difficulty with using the prototype. The testers were asked to attempt to complete these following tasks:

1. Connect your device to the game.
2. Disconnect from the game (not the network).
3. Connect again to the game and wait to be assigned into a team.
4. Watch tutorial on the central screen.
5. Identify your team's problem on the central screen.
6. Cooperate with your teammates to solve a problem.
7. Identify your team on the leaderboard.
8. Finish the game.

Consecutive questionnaire Many players were not able to give a valuable feedback verbally, reflecting more on their gaming experience rather than the usability of the prototype. For this reason a short survey was created, that would help to further assess the prototype. The questionnaire consists of these following questions:[32]

[32]Translated from Czech.

1. Do you own a smartphone or a tablet? {Yes/No}[33]
2. Did you manage to connect your device to the game? {Yes/No}
3. Were you disconnected against your will over the course of the game? {Yes/No}
4. How much did you mind the fact you could not choose your team? {Scale from 1 to 5}
5. Did you understand the rules of the game and the events over the course of the game? {Scale from 1 to 5; 1 – everything was clear, 5 – did not understand even after several rounds}
6. Try to assess the game's complexity. {Scale from 1 to 5; 1 – extremely simple, 5 – too difficult}
7. How many players combined were in your team? {From 2 – 5 players}

Testers were encouraged to add own notes and further feedback under each question.

6.2.2 Testers

Because the section 2.2 describes the target audience as a diverse group, the selected testers should reflect that fact as well. The usability testings were conducted on several occasions with different sets of testers. The first few testers were students at CTU FIT, very technically savvy group of people in their 20's. The second group was rather diverse group of people during Open house at Technical college of information studies where some testers were teenage attendees and some part of the school management.

6.2.3 Conducted testings

Testing at CTU: Game configuration: Initially one team with four connected players, difficulty set easy and round time to 90 seconds. Over time the number of players progressed and further games were played by multiple teams, usually with 2 to 3 players and the difficulty set to hard. Connected players: At peak 24 player controllers were connected to the game, with 8 teams participating in the game. Testers have not been given any additional information about the system. Completion of usability test scenario can be seen in table 6.1.

Testing at Open house: Game configuration: 2 teams with 2 players per team. Difficulty set to easy, round duration was 1 minute. Connected players: At peak 4 player controllers were connected to the game. Testers have not been given any additional information about the system. Completion of usability test scenario can be seen in table 6.2;

[33]This question has been asked again because some testers used borrowed devices, which can affect their ability to use the game smoothly.

	Tester 1	Tester 2	Tester 3
1	✓	✓	✓
2	✓	✓	✓
3	✓	✓	✓
4	✓	✓	completed with difficulty, see section 6.2.4
5	✓	✓	✓
6	✓	✓	✓
7	✓	✓	✓
8	✓	✓	✓

Table 6.1: Test scenario completion: testers at CTU

	Tester 1	Tester 2	Tester 3	Tester 4	Tester 5	Tester 6
1	✓	✓	not completed, see section 6.2.4	✓	✓	✓
2	✓	✓	✓	✓	✓	✓
3	✓	✓	✓	✓	✓	✓
4	✓	✓	✓	✓	✓	✓
5	✓	✓	✓	✓	✓	✓
6	✓	✓	✓	✓	✓	✓
7	✓	✓	✓	✓	✓	✓
8	✓	✓	✓	✓	✓	✓

Table 6.2: Test scenario completion: testers at Open house

6.2.4 Testing analysis

Testing at CTU: After the test script was completed, the testers were further asked to compete for a substantial prize to further motivate the group to react faster. The prototype performed well even during this fast paced testing, where noticeable latency would cause frustration to the testers. The general assessment has been rather positive, with only a few complaints about the UI on mobile device and some unreliable response to their actions[34] on the main screen. Despite the positive feedback, it must be taken into consideration that all of the testers in this testing setting have been technically savvy and therefore had no additional problems with the setup, such as typing an IP address into a browser window. Although it has not resulted in difficulty during the game play, one tester stated, that she had difficulty following the tutorial, as she cannot read English very fast and suggested using more animation instead of text. In response to the first testing a major UI update was added to the

[34]While the game and score has been progressing safely, from time to time the players' numbers and sum printed on the main screen have not been up to date. This bug has been recorded happening anywhere from 0 to 16 times per round.

controller on mobile device along with a QR code for more intuitive joining process. The second testing couple days later, with the same type of technically savvy testers, assessed that the colors of a few teams are difficult to tell apart and thus the author changed the game object textures to closely match the team banners in color. This fix can be seen in figure 5.5. Another issue was raised as two or more teams ended up with the same amount of points but only one winner was announced.

Testing at Open house: Most testers were able to finish all steps of the testing scenario. One person needed further instruction to be able to connect to the game and later filled this following feedback to the follow-up questionnaire: *"[Connecting to the game] caused me difficulty; right after filling in [the address], I was redirected to google.com and nothing happened."* Which points out that the player connection to the game must be further simplified. This time players reacted positively to the game UI, commenting on clear indication of each player's answer and the team sum on the central screen.

6.2.5 Survey feedback

The introductory survey has been filled by 22 players and the consecutive one by 14 testers. All respondents stated that they know how to connect mobile device to a Wi-Fi and how to use a web browser on their devices. While most respondents, 21 out of 22, declared that they know how to install an application on their device, only 8 stated that they would install an application for an event game. Some testers added that the included tutorial did not help them and was frustrating as it contained too much text. Most testers evaluated the game's complexity as fairly low (2 out of 5) and played predominantly in a team of 2 or 3 players.

While the data from the questionnaires is interesting, it is needed to conduct several additional testings to gather more responses.

6.3 Concluded issues

While the testing questionnaires provided some information, the most valuable feedback has proved to be watching the players during the game and listening to their complaints.

Additional testing analysis resulted in a list of issues that appeared during testing. The author ordered these problems by its impact on the game play and will further work on eliminating them before another testing iteration.

6.3.1 Only one winning team – high priority

When two or more teams achieve the same winning score only the last surpassing team is announced as the winner.

6.3.2 Difficult connection – medium priority

Player must connect to the local Wi-Fi and then allocate web browser and input IP address. Despite having access to a hyperlink QR code, this has proven to be an issue for some players and they must have been assisted before they were able to connect to the game. Possible solution seems to be the implementation of a captive portal, that would redirect the players to the URL address right after they connected the device to the local network.

6.3.3 Error in keeping team sum – medium priority

Sometimes during game play the team sum is not reset on the server. This bug appeared twice during the described testings.

6.3.4 Error in displaying individual answers on central screen – low priority

Occasionally, after player suddenly disconnects during the game, *UnityApplication* takes a bit of time to change the number of players in team. This error appeared once during the described testings.

6.4 Future testings

After the above issues are fixed the prototype will be required to undergo another round of basic functionality testings before being tested again for user usability.

Although the prototype has been designed to handle 40 concurrent players, the most devices that had been connected at one time has been 28[35]. Because of that, the prototype will be required to undergo an additional stress testing to ensure that it does perform as designed.

[35]This has been an action outside of the usability testing, conducted spontaneously and without the test case scenario. Since the author of the prototype had heavily intervened during the process of playing, the event could not be included as an official testing.

Conclusion

This thesis is dedicated to the design and implementation of a distributed interface for a local cooperative game. The realised game prototype is transportable and controlled via mobile devices, implementing the BYOD gaming model.

First, the thesis describes several up to date solutions of distributed interfaces for local cooperative games, their advantages and limiting factors along with few concepts applied to cooperative gaming in general. Although the subject of all possible co-op games, that could be controlled via mobile devices and played in immersive environments, has not been exhausted in the text, the conducted research provided a steady base for a detailed list of functional and non-functional requirements and further analysis of own design for distributed interface for a local cooperative game. The latter chapters are detailing the process of choosing the right technologies and further developing the design of own solution. The author then proceeded to implement the design and realised a functional prototype of the cooperative game *NUMBERS* to demonstrate the functionality of own solution. The design resulted in a working prototype with minimal setup and the ability to manage from 2 to 40 players, which has been tested in real operational environment and performed well.

Room for future improvement

Testing of the prototype helped to uncover few notable errors, most of which has been fixed, as well as pointed out several areas for improvement.

It has not been explicitly mentioned during the testing, but some players seemed to be uneasy with having no choice in choosing their teammates. While the analysis does mention the possibility of players choosing their own teams, this feature has not been incorporated into the prototype, which is something that will change in the future.

Additionally, the process of connecting players to the game wil be further simplified by implementing a captive portal web page. As some testers visibly struggled with the language of the prototype, a fully fledged application should implement multiple language mutations.

Assignment completion

1. **Studying available literature concerning matters of cooperative games.** This point is presented in the State of the art, chapter 1. The author describes the main principles of cooperative games and analyses several up to date solutions.

2. **Design proof of concept for such distributed game; describe used mechanics and principles, ideally in a form of short Game Design Document.** From the conducted research, the author deduced several requirements for own local cooperative game. The author further specified the mechanics and principles of the developed game *NUMBERS* along with specifying the behaviour of each participant. The analysis led to a state machine diagram, describing the course of the game *NUMBERS*. The process of designing the proof of concept is described in chapters 2, 3 and 4.

3. **Implement prototype of a chosen simplified game for multiple mobile devices and one central screen.** Prototype of the game *NUMBERS* has been realised. The game can accommodate from 2 to 40 players and has been primarily developed for an UHD projection screen. The process of realisation is decribed in chapter 5.

4. **Test and assess the prototype in real operational environment.** Although the prototype is yet to be stress tested, it has been tested both for functionality and user usability on multiple occasions. The process of evaluation is described in chapter 6.

5. **Write down and analyze your observations.** The analysis of several questionnaires filled by testers resulted in mostly positive feedback. The prototype was viewed positively in regards to the use of mobile phones and local Wi-Fi, which assessed it as a viable solution to local cooperative gaming. While some testers struggled with few tasks, the prototype has been mostly evaluated as easy to use and provided a positive gaming experience. The testing also resulted in a list of improvements for upcoming update of the prototype. Because of the mostly positive feedback, the prototype will be further developed into a fully fledged application.

Bibliography

[1] HANCHAR, T. The Top 10 Event Networking Games. [online], 2018-01-09, [cit. 2018-05-02]. Available from: `https://www.gevme.com/blog/top-10-event-networking-games/`

[2] Z/YEN. Event Games. [online], 2016, [cit. 2018-05-02]. Available from: `http://www.zyen.com/what-we-do/training-games.html`

[3] BENFORD, S.; Gabriella Giannachi, T. R., Boriana Koleva. From Interaction to Trajectories: Designing Coherent Journeys Through User Experiences. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '09, New York, NY, USA: ACM, 2009, ISBN 978-1-60558-246-7, pp. 709–718, doi:10.1145/1518701.1518812. Available from: `http://doi.acm.org/10.1145/1518701.1518812`

[4] KINO SVĚTOZOR. Kinoautomat. [online], [cit. 2018-05-02]. Available from: `http://www.kinoautomat.cz/index.htm`

[5] FUSTER, J. 'Choose Your Own Adventure' Interactive Movie in Works at Fox. *The Warp*. [online], 2018-04-26, [cit. 2018-05-02]. Available from: `https://www.thewrap.com/choose-adventure-interactive-movie-works-fox/`

[6] CTRLMOVIE^{TM}. CtrlMovie. [online], [cit. 2018-05-02]. Available from: `http://www.ctrlmovie.com/`

[7] SLEEPING BEAST GAMES. SpaceTeam. [software]. Available from: `http://spaceteam.ca/`

[8] KLOPFER, E.; Judy Perry, e. a. Mystery at the museum: a collaborative game for museum education. In *Proceedings of th 2005 conference on Computer support for collaborative learning: learning 2005: the next 10 years!*, International Society of the Learning Sciences, 2005, pp. 316–320. Available from:

https://www.researchgate.net/publication/221033626_Mystery_at_the_museum_a_collaborative_game_for_museum_education

[9] LI, L.; Zhou, J. Virtual Reality Technology Based Developmental Designs of Multiplayer-interaction-supporting Exhibits of Science Museums: Taking the Exhibit of "Virtual Experience on an Aircraft Carrier" in China Science and Technology Museum As an Example. In *Proceedings of the 15th ACM SIGGRAPH Conference on Virtual-Reality Continuum and Its Applications in Industry - Volume 1*, VRCAI '16, New York, NY, USA: ACM, 2016, ISBN 978-1-4503-4692-4, pp. 409–412, doi:10.1145/3013971.3014018. Available from: http://doi.acm.org/10.1145/3013971.3014018

[10] BELLADA, L. Rybolov. [software], VIDA! center in Brno, The Czech Republic. 2014. [cit. 2018-04-17].

[11] BROWN, M. *Spaceteam Apple TV Review: Can A Fun Mobile Game Work On The Big Screen?*. International Business Times. [online], 2015-11-05, [cit. 2018-05-06]. Available from: http://www.ibtimes.com/spaceteam-apple-tv-review-can-fun-mobile-game-work-big-screen-2171177

[12] WEISSKER, T.; Andreas Berst, F. E., Johannes Hartmann. The Massive Mobile Multiuser Framework: Enabling Ad-hoc Realtime Interaction on Public Displays with Mobile Devices. In *Proceedings of the 5th ACM International Symposium on Pervasive Displays*, PerDis '16, New York, NY, USA: ACM, 2016, ISBN 978-1-4503-4366-4, pp. 168–174, doi:10.1145/2914920.2915004. Available from: http://doi.acm.org/10.1145/2914920.2915004

[13] MEINHARDT, T. Fußball für alle: MMM Ball. [online], [cit. 2018-05-02]. Available from: https://www.uni-weimar.de/en/university/profile/events/archiv/events-2015/summaery2015/summaery2015-in-progress/fussball-fuer-alle-mmm-ball/

[14] BELLADA, L. [personal communication], 2017-10-26, [cit. 2018-04-17].

[15] SAVIČ, D. *Návrh a implementace řídicího systému pro exponát Kulatý stůl ve VIDA! science centru.* Master's thesis, Mendel University in Brno, 2017. Available from: https://theses.cz/id/qicf2q/zaverecna_prace.pdf

[16] AV MEDIA. ZÁBAVNÍ VĚDECKÝ PARK VIDA! [online], [cit. 2018-05-02]. Available from: http://www.avmedia.cz/reference/detail/59_2484-zabavni-vedecky-park-vida

[17] DUKOVANY POWER PLANT INFORMATION CENTRE. [personal communication], 2018-04-06, [cit. 2018-04-06], comunication with nonymous guide at the information centre.

[18] BARCLAY, P. Trustworthiness and competitive altruism can also solve the âĂIJtragedy of the commonsâĂİ. *Evolution and Human Behavior*, volume 25, no. 4, 2004: pp. 209–220.

[19] KOU, Y.; Gui, X. Playing with Strangers: Understanding Temporary Teams in League of Legends. In *Proceedings of the First ACM SIGCHI Annual Symposium on Computer-human Interaction in Play*, CHI PLAY '14, New York, NY, USA: ACM, 2014, ISBN 978-1-4503-3014-5, pp. 161–169, doi:10.1145/2658537.2658538. Available from: `http://doi.acm.org/10.1145/2658537.2658538`

[20] ROCHA, J. B.; S. Mascarenhas, R. P. Game mechanics for cooperative games. *ZON Digital Games 2008*, 2008: pp. 72–80.

[21] KAMEDA, T. e. a. Social dilemmas, subgroups, and motivation loss in task-oriented groups: In search of an “optimal” team size in division of work. *Social Psychology Quarterly*, 1992: pp. 47–56.

[22] ZAGAL, J. P.; Jochen Rick, I. H. Collaborative games: Lessons learned from board games. *Simulation & Gaming*, volume 37, no. 1, 2006: pp. 24–40.

[23] REUTER, C.; Viktor Wendel, R. S., Stefan Göbel. Game Design Patterns for Collaborative Player Interactions. In *DiGRA*, 2014.

[24] RED DIGITAL CINEMA CAMERA COMPANY. Video aspect ratios. [online], [cit. 2018-05-08]. Available from: `http://www.red.com/learn/red-101/video-aspect-ratios`

[25] PENNINGTON, A. The resolution war: is cinema falling behind home entertainment on innovation? [online], 2017-11-07, [cit. 2018-05-08]. Available from: `https://www.screendaily.com/features/the-resolution-war-is-cinema-falling-behind-home-entertainment-on-innovation/5124023.article`

[26] *SAGE2^{TM}*. [online], [cit. 2018-05-02]. Available from: `http://sage2.sagecommons.org/`

[27] CRUZ-NEIRA, C.; D .Sandin, T. D. Surround-screen Projection-based Virtual Reality: The Design and Implementation of the CAVE. In *Proceedings of the 20th Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '93, New York, NY, USA: ACM, 1993, ISBN 0-89791-601-8, pp. 135–142, doi:10.1145/166117.166134. Available from: `http://doi.acm.org/10.1145/166117.166134`

[28] YU, K.; Dan Neafus, R. W. Filmmaking for Fulldome: Best Practices and Guidelines for Immersive Cinema (Part I). volume 45, 12 2016: pp. 26–32,34,36,38.

[29] GERLA M. and D. Maggiorini, C.E. Palazzi, A. Bujari. A survey on interactive games over mobile networks. *Wireless Communications and Mobile Computing*, volume 13, no. 3: pp. 212–229, doi:10.1002/wcm.2197, `https://onlinelibrary.wiley.com/doi/pdf/10.1002/wcm.2197`. Available from: `https://onlinelibrary.wiley.com/doi/abs/10.1002/wcm.2197`

[30] LE, A. e. a. MicroPlay: A Networking Framework for Local Multiplayer Games. In *Proceedings of the First ACM International Workshop on Mobile Gaming*, MobileGames '12, New York, NY, USA: ACM, 2012, ISBN 978-1-4503-1487-9, pp. 13–18, doi:10.1145/2342480.2342485. Available from: `http://doi.acm.org/10.1145/2342480.2342485`

[31] NILSON, B.; Söderberg, M. Game Engine Architecture. *Mälardalen University*, 2007. Available from: `http://www.idt.mdh.se/kurser/cd5130/jgms/2007lp4/report9.pdf`

[32] KELLEHER, F. Understanding Socket.IO. [online], 2014-08-10, [cit. 2018-04-28]. Available from: `https://nodesource.com/blog/understanding-socketio/`

[33] QVEFLANDER, N. *Pushing real time data using HTML5 Web Sockets*. Master's thesis, UmeåUniversity, Faculty of Science and Technology, Department of Computing Science, 2010. Available from: `http://www.diva-portal.org/smash/get/diva2:354621/FULLTEXT01.pdf`

[34] WODEHOUSE, C. Front-End Web Development: Client-Side Scripting and User Experience. [online], 2016, cit. [2018-05-11]. Available from: `https://www.upwork.com/hiring/development/how-scripting-languages-work/`

[35] W3SCHOOLS. HTML5 Tutorial. [online], [cit. 2018-04-29]. Available from: `https://www.w3schools.com/html/`

[36] SHEPHERD, E. e. a. Writing Websocket servers. article, 2017-08-16, [cit. 2018-04-28]. Available from: `https://developer.mozilla.org/en-US/docs/Web/API/WebSockets_API/Writing_WebSocket_servers`

[37] SCHOLZ, F. e. a. User-Agent. [online], 2018-01-24, [cit. 2018-04-28]. Available from: `https://developer.mozilla.org/en-US/docs/Web/HTTP/Headers/User-Agent`

[38] STUDIO, U. socket.io-unity. [software], [cit. 2018-05-02]. Available from: `https://github.com/floatinghotpot/socket.io-unity`

[39] UNITY TECHNOLOGIES. Unity User Manual (2018.1). [online], [cit. 2018-05-05]. Available from: `https://docs.unity3d.com/Manual`

[40] RUBIN, J.; Chisnell, D. *Handbook of usability testing: howto plan, design, and conduct effective tests.* John Wiley & Sons, 2008, ISBN 978-0-470-18548-3.

Appendix **A**

Acronyms

AJAX	Asynchronous JavaScript And XML
BYOD	Bring Your Own Device
CAVE	Cave Automatic Virtual Environment
CSS	Cascading Style Sheets
FHD	Full High Definition resolution
GUI	Graphical User Interface
HTML	HyperText Markup Language
HTTP	HyperText Transfer Protocol
SAGE	Scalable Adaptive Graphics Environment
SAGE2	Scalable Amplified Group Environment
TCP	Transmission Control Protocol
UHD	Ultra High Definition
UI	User Interface
WebGL	Web Graphics Library

Appendix B

Contents of enclosed DVD

```
├── readme.txt ................................ DVD contents description
├── installation.pdf .......... installation guide for the game NUMBERS
├── src
│   ├── numbers_web ............... the directory containing web applications
│   └── numbers ..................... the directory with Unity3D application
└── text
    ├── thesis .............. the directory of LaTeX source codes of the thesis
    └── latex .................................... the thesis text directory
```

Appendix C

Examples of prototype GUI

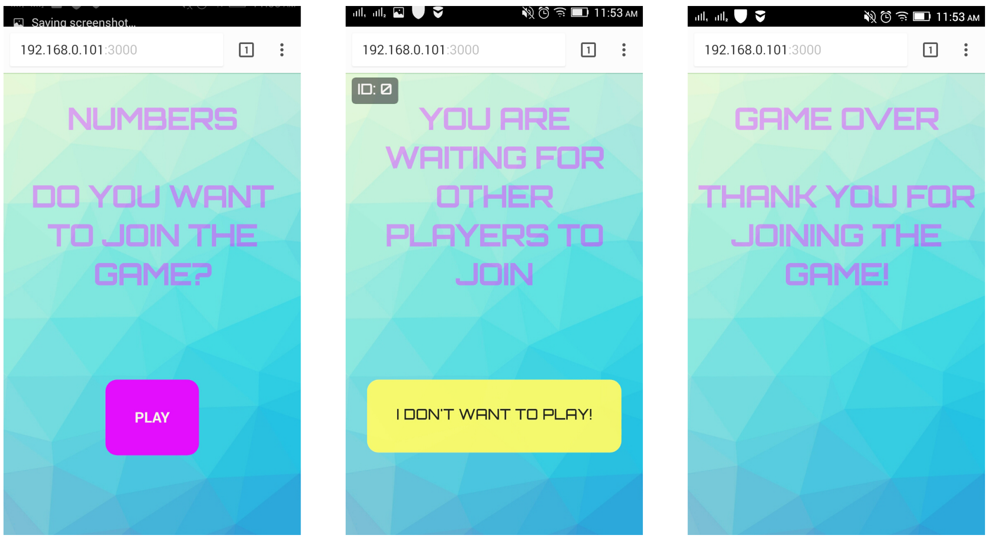


Figure C.1: *PlayerController* – welcome screen, waiting screen, game over.

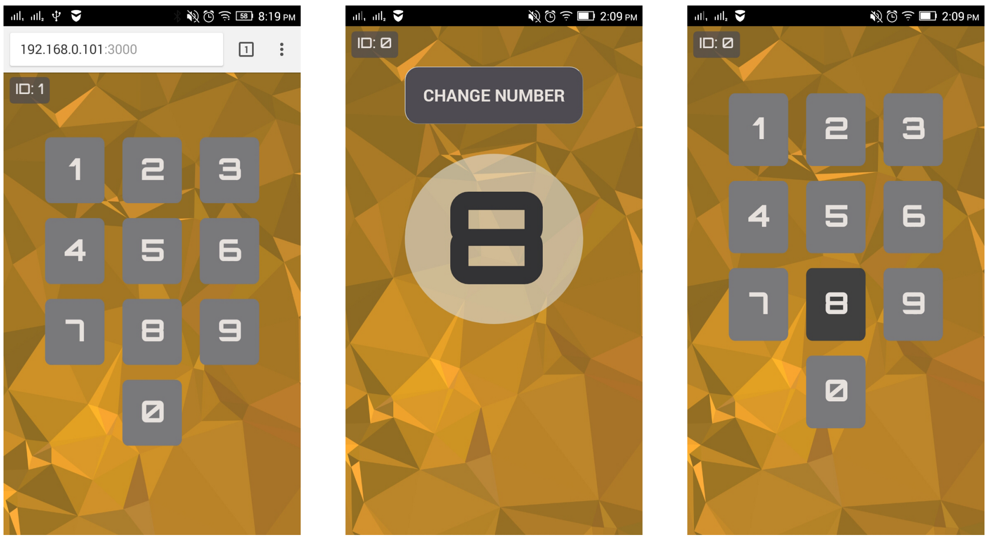
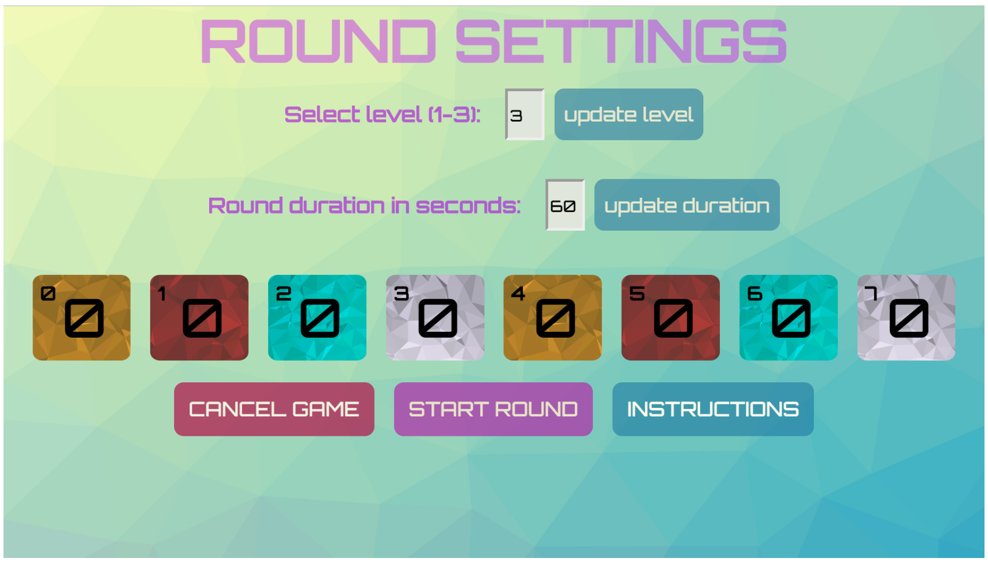


Figure C.2: *PlayerController* – screens during game play: choose answer, answer screen, change answer.



Select level (1-3): 3

Round duration in seconds: 60

Figure C.3: *ModController*: Round settings, 8 players are connected and divided into 4 teams.

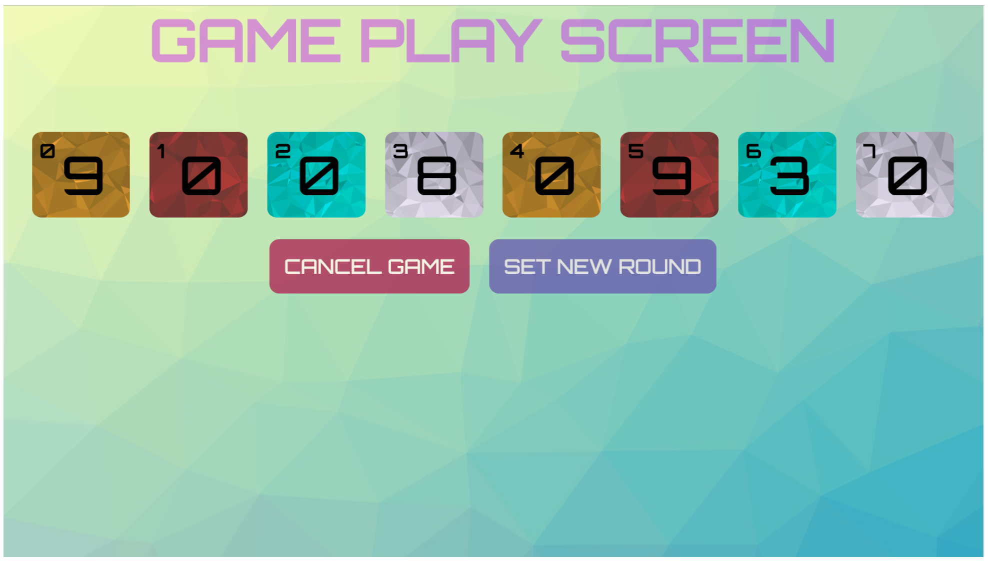
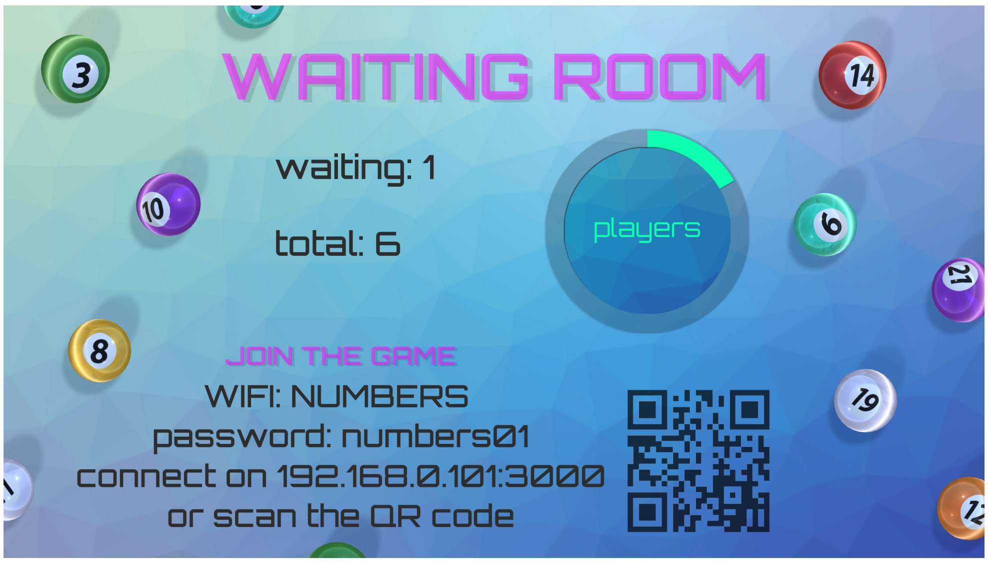


Figure C.4: *ModController*: Game play, 8 players are connected and divided into 4 teams.



WIFI: NUMBERS
password: numbers01
connect on 192.168.0.101:3000
or scan the QR code

Figure C.5: *UnityApplication*: Waiting for players to connect.

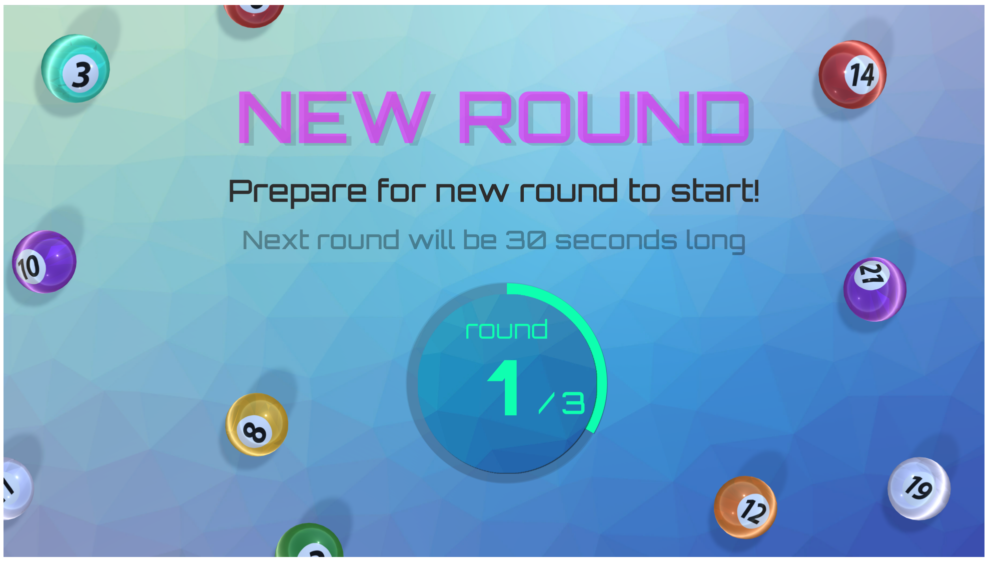
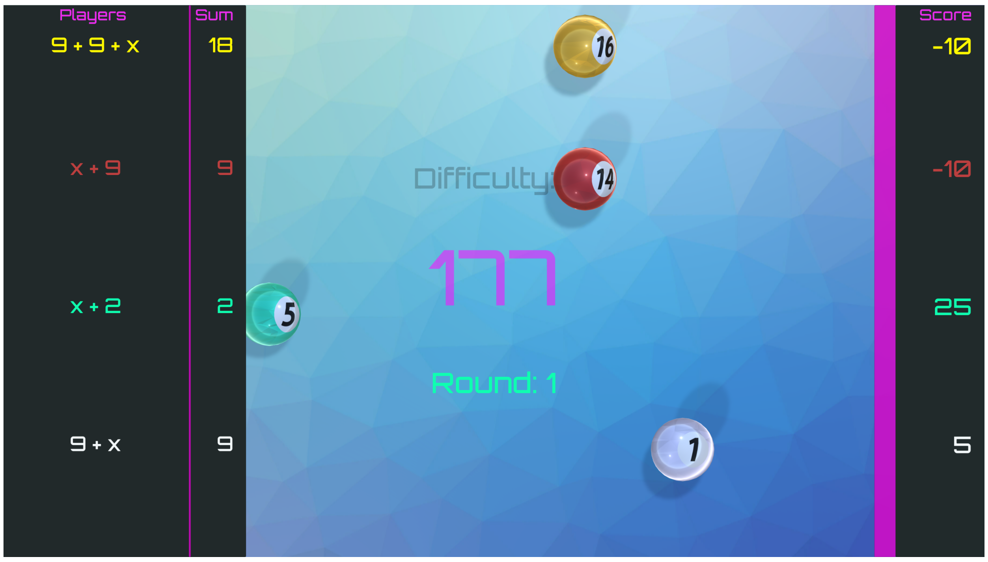
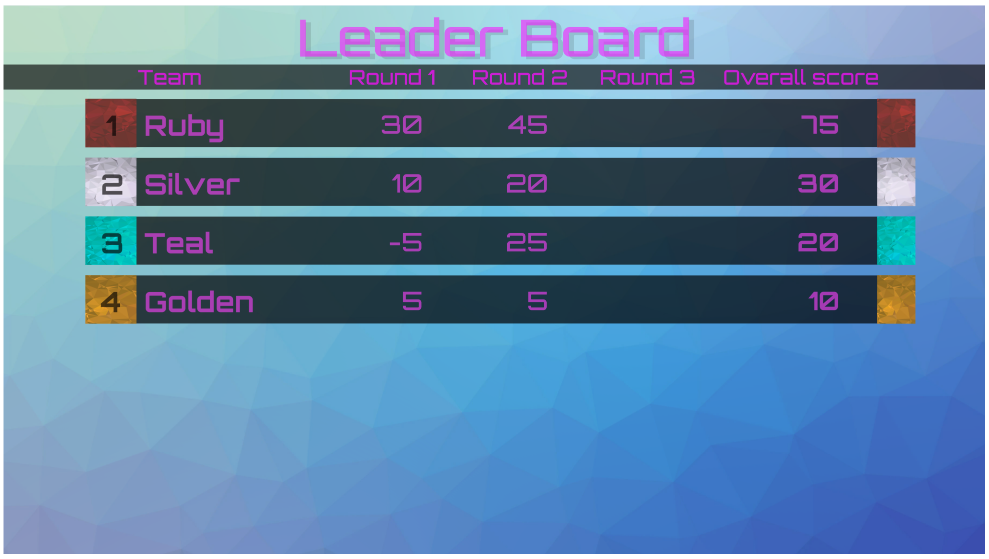

3
14
NEW ROUND
Prepare for new round to start!
Next round will be 30 seconds long
10
21
round
1 / 3
8
12
19

Figure C.6: *UnityApplication*: Waiting for new round.



Players	Sum	Score
9 + 9 + x	18	-10
x + 9	9	-10
x + 2	2	25
9 + x	9	5

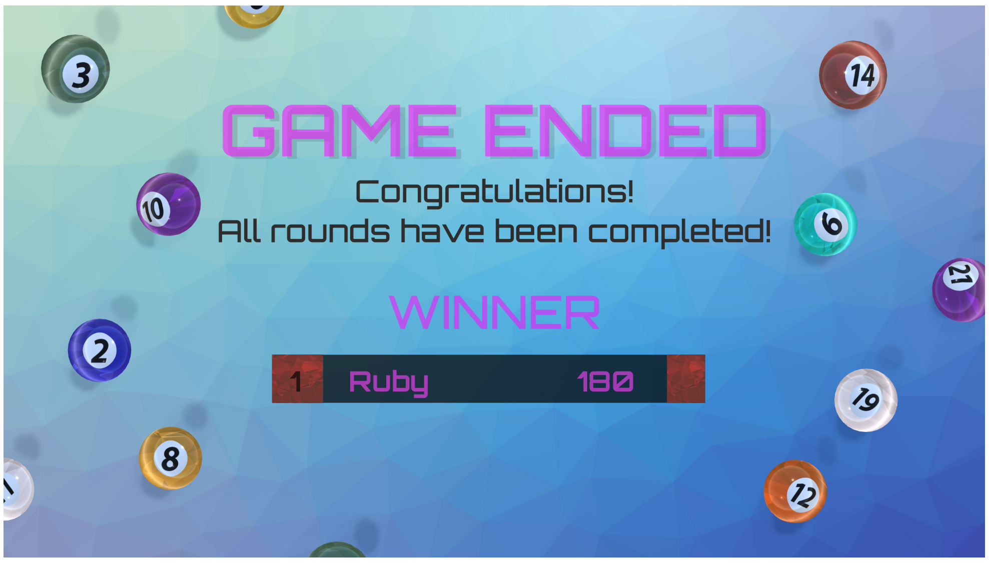
Figure C.7: *UnityApplication*: Game play, 9 players are divided into 4 teams.



Leader Board

Team	Round 1	Round 2	Round 3	Overall score
1 Ruby	30	45		75
2 Silver	10	20		30
3 Teal	-5	25		20
4 Golden	5	5		10

Figure C.8: *UnityApplication*: Leader board after 2 rounds. Four teams are competing.



GAME ENDED

Congratulations!
All rounds have been completed!

WINNER

1 Ruby 180

Figure C.9: *UnityApplication*: Game over after three rounds.