League of Lasers

S. Alaka,
D.J.M. de Bruin,
N.A. Miedema,
J. Vermeer

Technische Universiteit Delft

TU Delft

League of Lasers

by

S. Alaka,
D.J.M. de Bruin,
N.A. Miedema,
J. Vermeer

Supervisors:	Dr.ir. A.R. Bidarra,	TU Delft, supervisor
	Dr. S.G. Lukosch,	TU Delft, customer
Bachelor Coordinator:	Dr.ir. H. Wang,	TU Delft
	Ir. O.W. Visser,	TU Delft

An electronic version of this document is available at `http://repository.tudelft.nl/`.

TU Delft

Summary

This report describes the development of a superhuman sports multiplayer game that makes use of augmented reality through the HoloLens. The game is a reenvisioning of the original League of Lasers, a pong/football-esque game for Android phones using camera tracking and special headgear for player positioning. The game was specifically redesigned to use neither a camera or a phone, but instead use the HoloLens' spatial localisation features. The goal of the project is to provide a superhuman sports multiplayer experience on the HoloLens. To be able to provide this, a system for managing spatial anchors and a custom networking architecture were developed, making use of a game server, game clients, a master game client and a web server for sharing anchors. Along with this, large-scale playtests were performed, where data was collected. The corresponding game design changes from the original game are described and implemented to ensure an optimal experience. As the game was developed in Unity with the Mixed Reality Toolkit, software quality was kept at adequate levels throughout the project through a custom CI setup, code written with testability and maintainability in mind through the humble object pattern, and rigorous PR reviewing guidelines. For the development process itself, Scrum was used with weekly meetings, a Waffle project board and daily status reports. Numerous problems with the combination of Unity, the Mixed Reality Toolkit and the HoloLens made the project a laborious endeavour, but the final result and design of the game exceeded our initial expectations.

Preface

This report describes the result of our Bachelor End Project. The bachelor end project is the last project to complete our bachelors degree in computer science. The goal of the project was to design a "superhuman sport" that would be submitted to the *Superhuman Sports* (SHS) Design Challenge in July 2018. The project's client is the TU Delft Sports Engineering Institute, with dr. S.G. Lukosch being the contact person for the institute. Our coach and supervisor was dr.ir. A.R. Bidarra.

During the project, we worked together as a group of four students starting on 23 April till 4 July 2018. For 11 weeks we delved into the development of games on the Microsoft HoloLens using Unity. At the time of writing the device is still only intended for developers. This meant that the HoloLens would support most of its official features, but documentation and working examples were scarce, moreover undocumented bugs were still present. This made the project a challenge at times, there were moments where we feared these limitations would prohibit us from realising the project, or that the game might not be fun. However, once we got our first alpha build running on the HoloLens in week 7 we were ecstatic, the game worked well and more importantly, it was fun.

We had our first official reveal on the 21st of June at the Virtual Playground Event in the TU Teaching Lab, where we got valuable feedback. We're proud of the resulting game and in the end found the project to be a rewarding experience. Unfortunately, the deadline of this report is before the SHS Design Challenge so we cannot reflect on that part of the project in this report, but we're hopeful for that podium spot.

Finally, we would to express our gratitude towards all people that helped to realise the end result. We thank dr. S.G. Lukosch for being an awesome client: during the project we had some setbacks and dr. S.G. Lukosch was always understanding of these issues and remained patient, and most importantly gave honest feedback about how to improve the game. We thank dr.ir. A.R. Bidarra for being the best supervisor we could possibly have for this project: not only would he provide us with valuable feedback, his guidance helped solve some of the critical issues in the game and his enthusiasm helped us stay motivated. Dr.ir. A.R. Bidarra would drop by from time to time just to say hi and check if we encountered any issues. We would also like to thank him for setting us up with a workplace in the INSYGHTLab, as the location made it easy for us to collaborate together. We thank dr.ir. H. Wang and ir. O.W. Visser for being part of the assessment committee, and finally we thank drs. M.J.J. Beerens for inviting us to the Virtual Playgrounds event, which was a fantastic experience.

S. Alaka,
D.J.M. de Bruin,
N.A. Miedema,
J. Vermeer
Delft, June 2018

Contents

1 Introduction

Over the past thirty years, computer games have risen to become a primary form of entertainment, being highly effective at captivating users into all kinds of virtual activities [21]. While there was interest in immersing the user more into games through means of *Augmented Reality* (AR) or *Virtual Reality* (VR), the technology developed for this did not work well enough to provide a compelling experience [31]. Thus, playing video games was and still is generally performed sedentary behind a computer screen.

In the recent years, interest in providing AR or VR video game experiences started to rise again [4]. Specifically for AR (but not necessarily just for games), Microsoft has developed the HoloLens, a wearable device capable of accurately laying holograms, representing virtual objects, over the real world through a visor [23]. This creates massive potential for providing new forms of gaming experiences, where the player is no longer bound to sit on a chair, but rather, is encouraged to move around. Players would then engage in physical activity to play the game competitively, having an experience that could resemble real life sports.

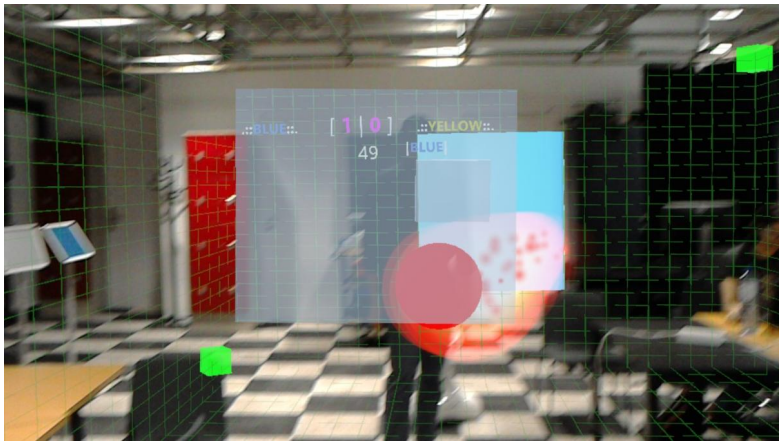


Figure 1.1: An in-game screenshot of the developed game

In this report we would like to describe how we made use of this opportunity to transform our previous attempt at creating an AR game, which we described in our own paper [46], into our very own superhuman sport experience on the HoloLens (see figure 1.1), where two teams of up to three players play a competitive and physically engaging game akin to football and pong. The game is being developed for the SHS Design Challenge [33], to showcase the current capabilities and high potential of AR sports.

1.1. Outline

First, an analysis and definition of the problem will be provided to clarify what the requirements of the product are and how the developed product will satisfy these. Then, a description of the software design process will be given, which will go over the team's cooperation procedures, meetings with supervisors, development tools, and challenges encountered during development. After that, the actual design of the game itself will be

described, explaining how the game is played and how it differs from the original concept. Then, the design of the developed networking architecture is elaborated upon. Following the network architecture's design, the methodologies for keeping the software quality adequate are discussed, touching upon testing, but also upon play testing, code analysis, *Continuous Integration* (CI) and lastly discussing the feedback from the code quality organisation: *Software Improvement Group* (SIG). After that, we go into the implementation details of all the components of the game, describing their how they work and how they are used. Then, an analysis of the game's ethical implications is given, using the *ethical Technology Assessment* (eTA) framework. Before concluding the report, discussion regarding the future of League of Lasers and superhuman sports in general will be provided, with recommendations for the client. Finally, the report will close with concluding remarks regarding the successfulness of the project, touching upon the fulfilled criteria and requirements.

2

Problem Definition and Analysis

This section will give a definition and analysis of the problem, followed by a description of the goal and requirements of the final product.

2.1. Problem Definition

In July of 2018, the International Superhuman Sports Society organises the SHS Design Challenge, an international symposium celebrating the next generation of inventors [33]. The aim of the SHS Design Challenge is to challenge designers to create sports-like experiences, which involves physical fitness and skill, with a focus on augmenting human ability with technology. The sports are designed to be played for recreational and health purposes.

The goal of this project is to redesign the game League of Lasers in such a way that it is more suitable to be regarded as a superhuman sport, and can be submitted to the SHS Design Challenge. League of Lasers was a pong/football like game that used a top down camera for player positioning (For further details of the mechanics of league of lasers see [46], section 4.1 or appendix E). League of Lasers' core mechanics revolve around reflecting a laser pulse to guide it towards a target. Each team has its own target, so the players of one team want to hit the opposing team's target while defending their own (see figure 2.1a and 2.1b). When a team's target is hit by the laser, the opposing team scores a point. The team that scored the most points at the end of the game wins. All in all, the general idea resembles football (soccer) and Pong.

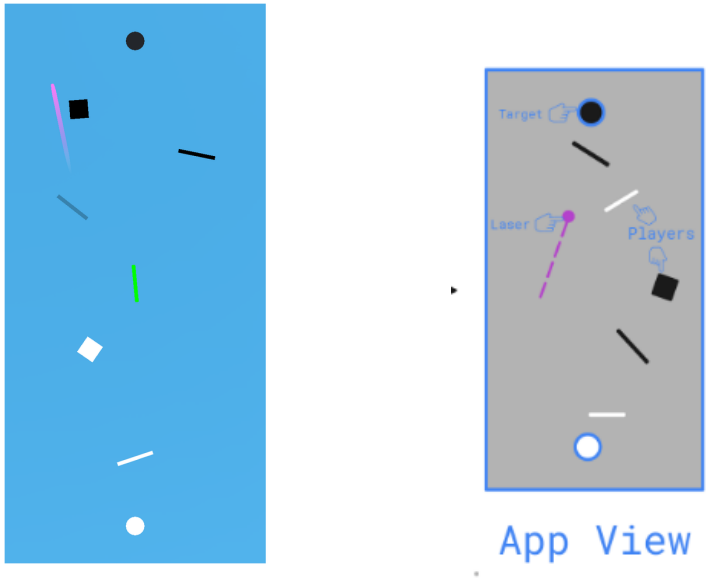
2.2. Requirements

An essential part of this project is the live demo at the SHS Design Challenge (see appendix B). The SHS committee will judge the following aspects of the game [33]:

- *Human augmentation:* How much does the superhuman sport augment human senses and capabilities?
- *Fitness and skills:* How much does the superhuman sport require or train physical fitness and skills?
- *Fun and engagement:* How engaging and fun is it for participants to play?
- *Innovation:* How innovative is the superhuman sport?
- *Audience:* How much fun is it for the audience to watch the superhuman sport?
- *Inclusiveness:* Can participants with different backgrounds practice the superhuman sport?

In order to submit League of Lasers to this competition the game will be redesigned as an AR game, which can be played using the Microsoft HoloLens. This brings a number of unique challenges that this project aims to solve, including (see appendix: B):

- Use of the HoloLens in a (large) multiplayer setting.
- Integration of tracking, orientation and precise interaction with limited bandwidth.



(a) An overview of the in app playfield.

(b) An abstraction of the app view.

Figure 2.1: An overview of the original League of Lasers

- Extend the game design to the extreme fast pace of the game (i.e. to the bare minimum imposed by hardware latency).
- Developmentment of new gameplay elements that prove to be fitting in this Superhuman context.

In addition to the live demo at the Superhuman Sports Design Challenge, another demo will be given at Virtual Playground, an VR/AR showcase event on 21 June 2018 [3]. This demo will serve as interim milestone to test the beta version.

In order to solve these challenges and make League of Lasers fit the SHS criteria, a design (see chapter 4) and MoSCoW analysis was created (see appendix C). In chapter 4 the core game mechanics are explained. These core mechanics are mostly the same as the old version of League of Lasers, since players enjoyed playing the old game [46]. The MoSCoW analysis was used to determine and prioritise the requirements of the project [7]. The first playable of the game shall include all the “must haves”. Then when the “should haves” are implemented, the base game is finished and some of the “could haves” will be implemented. These “could haves” will bring the game to the next level and take advantage of augmented reality technology the new game has access to. The finished project will have all “must haves” and “should haves” implemented. The “could haves” are implemented based on their priority. During meetings with the coach and client, these “could haves” are prioritised according to their needs and the SHS Design Challenge criteria.

3

Design Process

The design process that was used during this project will be elaborated upon in this section. The intention is to give an overview of the methodology used for cooperation, how the project was planned, which tools were used during the development of the game and the main issues encountered with HoloLens development.

3.1. Cooperation

During this project, the team worked as a game studio, where everyone has an assigned role. In this case, the roles were used more as responsibilities, for example the lead art director would be responsible for the art of the game, while the lead programmer would responsible that high quality code was written. Everyone could be assigned tasks of these different disciplines and since this is a software development project, everyone would write code, but the responsible person makes sure all tasks are finished on time and in high quality. The roles maintained and the team members assigned to these roles can be found in the plan of action (appendix D).

Since the HoloLenses were locked in the INSYGHTLab at the Delft University of Technology, the team worked mostly in the INSYGHTLab, five days a week, from nine to five. This also made it easier to share knowledge around working with Unity and the HoloLens.

3.1.1. Meetings

Each first day of the week, a meeting between the team members was held. Here the previous sprint was evaluated (see more about the Scrum workflow in section 3.2.2) and the new sprint is planned. The overall progress of the project and important issues and deadlines were discussed. After this meeting, issues were made and assigned on the Scrum board.

Each week, a meeting with the supervisor was planned. In these meetings, not only the weekly progress and the issues faced were discussed, but also what direction to take the game in and how well the mechanics that were implemented worked.

Weekly or biweekly meetings were planned with the client, depending on the progress made and how busy the client was. In these meetings it was discussed if they were satisfied with the progress made, which direction they wanted the game to take and thus which features should be prioritised and implemented.

3.1.2. Collaboration Tools

The team used several software tools to make collaboration and cooperation possible. For communication within the team, Discord was used. Discord gives the ability to create multiple text and voice channels to easily split up and discuss topics in a structured manner [8]. Discord also has excellent voice chat capabilities, including the ability to share one's screen to others [9]. For communication with our supervisor, WhatsApp was used. For all other communication purposes, we used email.

For collaboration for the documents, Google Docs was used in combination with Google Drive and ShareLatex. Google Docs was used to create the drafts of the documents, since it has extensive review tools, for example the ability to create suggestions for edits that can be easily implemented or rejected with the press of a button [15]. ShareLatex was then used to create the final versions of the documents, since it allows for multiple users to collaborate and work together in the same LaTeX document.

GitHub was used as source control during this project. All code created is uploaded to our GitHub repositories. GitHub contains the ability to easily create and review pull requests [14]. GitHub also allows the creation of issues with milestones, which were used to plan and assign tasks.

3.2. Planning

This project has as final goal to have an AR reenvisioning of League of Lasers for the Superhuman Sports Design Challenge, which takes place July 2-5 2018. To realise this, internal deadlines have been created for several stages of the game:

1. *May 25: first-playable*, a first version of the game that includes the core functionality necessary to test if the core mechanics are enjoyable. This version will not yet run on the HoloLens, as that is not necessary for this purpose.
2. *June 8: alpha version*, the alpha should contain the core functionality of the game, as described by the must haves of the MoSCoW analysis (see appendix C). From this version on out, the game will be playable on the HoloLens.
3. *June 22: beta version*, the feature complete version of the game. This version should contain all features, that can be implemented during the time span of this project. After this, no new features will be added.
4. *July 4: final version*, the final version of the game for this project. This version adds polish, but does not add anymore features.

3.2.1. Prototypes

because of the new technology (the HoloLens) used in this project, a lot of research was required. Several throw-away prototypes have been developed to test different functionality of the HoloLens and determine how well certain game mechanics would function in the augmented reality setting. The following prototypes where developed:

- *Spatial mapping prototype:* The spatial mapping prototype was developed to determine how well the HoloLens could map a room to a mesh and if we could use this mesh to let a virtual ball bounce around in the room.
- *Mirror prototype:* This prototype was used to test whether a holographic mirror, controlled via head movement would be intuitive.
- *Anchor prototypes:* Multiple prototypes were created to test and research the anchors of the HoloLens. These anchors ensure holograms are “anchored” to a real-world position. These prototypes include: testing a single anchor, testing multiple anchors, testing positioning of objects relative to anchors and sharing anchors via files and *Transmission Control Protocol* (TCP).

For more information regarding the spatial mapping and mirror prototype, please refer to the research report (appendix E).

For a large part of this project, the team was split into two groups: the first group would begin laying the foundation of the game and working towards the first-playable, while the second group worked on these research prototypes for anchoring. By careful planning, the first group could begin laying the game’s foundation and when the second group was finished with the prototype, the new insight could be applied to the game. This pipeline was very important for the planning of the tasks, especially in the earlier stages of development. Without this pipeline, the project would not be finished in time, as it allowed us to start working on the game while we were researching the capabilities of the new platform, the HoloLens.

3.2.2. Scrum

The team used the Scrum framework, which uses work cycles called sprints. A sprint length of 1 week was maintained, starting on Monday and ending on Sunday. Each day a daily stand-up was held, where team members discussed their plans for the day and the issues they had encountered the previous day. If at least one team member was absent, this daily stand-up was also posted on a special channel in the team’s internal messaging platform, Discord. Waffle was used as Scrum board, because of its elaborate integration into GitHub [35]. Waffle uses GitHub’s issues as tasks and can move them automatically to different columns when referenced in pull requests or commit messages. More details about our Scrum usage can be found in the plan of action (appendix D).

3.3. Development Tools

This project was developed using the development tools specified by Microsoft for developing HoloLens applications [48]. To build HoloLens applications, Windows 10 is required. It is preferred to use Windows 10 Pro, Enterprise or Education since this includes Hyper-V, which is necessary for the HoloLens emulator. During this project, the team used Windows 10 Enterprise and Education, version 1709.

The recommended game engine is Unity 2017.4 [48]. The latest 2017 version of Unity is 2017.4.3f1. Unity 2017 is prefered above the newer 2018 version as 2018 had issues with the test runner (it was unable to create a solution to run our unit tests in) [2]. Unity's default *Integrated Development Environment* (IDE) is Visual Studio 2017 Community, which is also the recommended IDE by Microsoft.

Unity has some useful features for HoloLens development. For example it includes Holographic Emulation, which can simulate an HoloLens directly in the editor [36]. This virtual HoloLens is controlled using a controller (e.g. an Xbox 360 controller). It is ideal for quick prototyping and debugging since it does not require building the application. The downside of the Holographic Emulation is that it does not have full support for anchors. It can not import and export anchors for example. Microsoft has also made available a package for Unity [25], the Mixed Reality Toolkit, which contains lots of examples and prefabs for usage with the HoloLens in Unity. While this package promises and sometimes delivers nice features, it is also poorly documented and a lot of the code and examples are broken due to changes in the HoloLens' and Unity's *Application Programme Interface* (API).

Microsoft's HoloLens emulator can also be used to simulate the HoloLens [29, p. 7–8] [20]. As mentioned previously, it requires Hyper-V. It is more powerful than Unity's built-in Holographic Emulation, as it includes all the *User Interface* (UI) elements of the HoloLens and access to the Windows Store, Settings menu, Device Portal, etc. During this project, we didn't find this HoloLens emulator to be useful in our workflow. Unity's Holographic Emulation was enough for simple testing purposes. For testing that could not be done using Holographic Emulation (e.g. testing functionality that required anchors or playtesting the UX on the HoloLens), we would simply build and deploy the game to the HoloLenses available.

3.4. Development Challenges

Life is full of challenges and so these can be expected when developing software. However, during the development of League of Lasers the team face more issues and challenges than usual, causing substantial delays and design overhauls during the course of the project. In this section, the development challenges faced during the project are discussed. Most of these issues can be attributed to the HoloLens and its software development kit, others were due to external factors.

3.4.1. HoloLens Development

During development with the HoloLens many issues were encountered. The most prevalent and most time consuming issues will be discussed in this section. First, issues relating to networking will be discussed, followed by those relating to the software used.

Networking related issues

While the HoloLens can easily be connected to the internet, running a networked Unity application on the HoloLens is not that simple. The application's server and the HoloLenses are required to run on the same local network. Moreover, the HoloLens is not able to host unity game servers, hosting servers is used in almost all examples released by Microsoft. Roughly two days were spent, by two team members, to try and get hosting on the HoloLens working, but it was ultimately deemed impossible. Tests were done with WireShark [41] and Advanced Port Scanner [12] to check network activity, here the discovery was made that the HoloLens never opened the port where the game should be hosted. This meant that using a dedicated server is the only way to run the game and that most examples that we wanted to run in the Mixed Reality Toolkit [25] did not function.

During development and research it was found that certain examples and scripts within the Mixed Reality Toolkit did not function as intended. At a certain point the team asked [27] on the Github repository whether the networking with *Unity Networking* (UNET) examples were abandoned. We received a response on this within a day explaining that the examples still work. However, this cannot be the case, as the example in question used the network discovery component, which has been broken for roughly a year and a half (according to forum posts [22, 24], correspondence with other teams creating games for the SHS Design Challenge and our own experience). Five weeks later Microsoft responded to the issue as well telling us that the hosted Wi-Fi network could be the issue [27]. At this time though, we were already too far into the project and did not have

the budget to buy a new router. Moreover, network discovery was found to work when ran on laptops, only on the HoloLens did these scripts fail. These problems with the Mixed Reality Toolkit forced us to abandon and ignore large parts of it and instead create our own anchoring, anchor sharing and networking systems. These developed solutions made us the first group on the Delft University of Technology that successfully used anchoring on the HoloLens, according to our client.

Another issue with networking was encountered after roughly four weeks into the project: the HoloLenses updated themselves spontaneously (this is due to the Windows 10 updating policy, which cannot be configured on the HoloLens). After this update, they could no longer connect to the game server. However, four days later Microsoft released a new update which fixed the issue on the HoloLens. Unfortunately, Microsoft did not disclose the presence of this bug in public anywhere, as such the team spent the four days hopelessly trying to get the game to connect to the server (something that did work before the update) and attempting to debug the issue, wasting precious time.

A different issue was that the HoloLens has limited access to its network settings. In practice, this means that the HoloLens is not able to function properly in conjunction with some routers or wireless networks (including Eduroam and the local NAO network in the INSYGHTLab). In order to remedy this, the team brought one of their own backup routers which was used for the remainder of the project.

Software related issues

Networking was not the only source of issues, there were also issues related to developing software for the HoloLens. For example, builds could take up to 7 minutes (5 minutes on average), slowing down iteration speed, and debugging is difficult as stepping through running code deployed on the device is impossible. The latter was important as components such as the anchors (see section 7.2.1) are not supported on the desktop version of Unity, removing the ability to use Unity's extensive editor-based debugging options for the HoloLens clients. This has led to extensive logging in the application, by implementing a debug console to make it easier to find bugs.

During weeks six and seven the team encountered another issue: the HoloLens' anchors were suddenly not accurate anymore and were likely to move around during play sessions. After a week of testing, our coach brought us in contact with a Melbourne based HoloLens developer. After consulting with him, it was eventually found that the spatial map of the surroundings had become corrupted [42]. This is a known issue that is not well documented. Deleting the spatial maps on the device seemed to resolve the issue, however a week of development was lost in researching the issue.

3.4.2. External factors

In this section, the issues and challenges related to external factors are discussed. The most prominent of these factors was that during the project there were quite some moments where we did not have access to the HoloLenses. The HoloLenses were not available to us in the first one and a half week, meaning that we were not able to start development and prototyping early on in the project. Moreover, during the starting weeks there were quite some holidays during which the faculty was closed and thus we did not have access to the HoloLenses. Another recurring issue was that we were not the only group that needed the HoloLenses. One of the other groups also needed them for a symposium (international festival of technology) which lasted three days. We tried to work around these issues as much as possible, scheduling documentation work for the times when the HoloLenses were unavailable to us, however this was not always ideal and in many of these cases access to the HoloLenses would've been convenient for debugging the game.

4

Game Design

In this section the general game design of League of Lasers AR is explained. This section will start with an explanation of the core mechanics of the game, followed by an explanation what changes have been made to the game compared to its predecessor. Finally the game flow is discussed, which will give insight into how the player experience is like.

4.1. Core Game Design

League of Lasers is an AR HoloLens game that aims to be a superhuman sport. The game revolves around players (represented by mirrors) reflecting a laser pulse into a target. The game consists of two teams, each with their own target to defend. If a target is hit, the other team is awarded one point (see figure 4.1). The general idea is somewhat like football and pong. The game uses the HoloLens' positional and orientational awareness to control the game. Players have to move around in a pitch/field to also move within the virtual world.

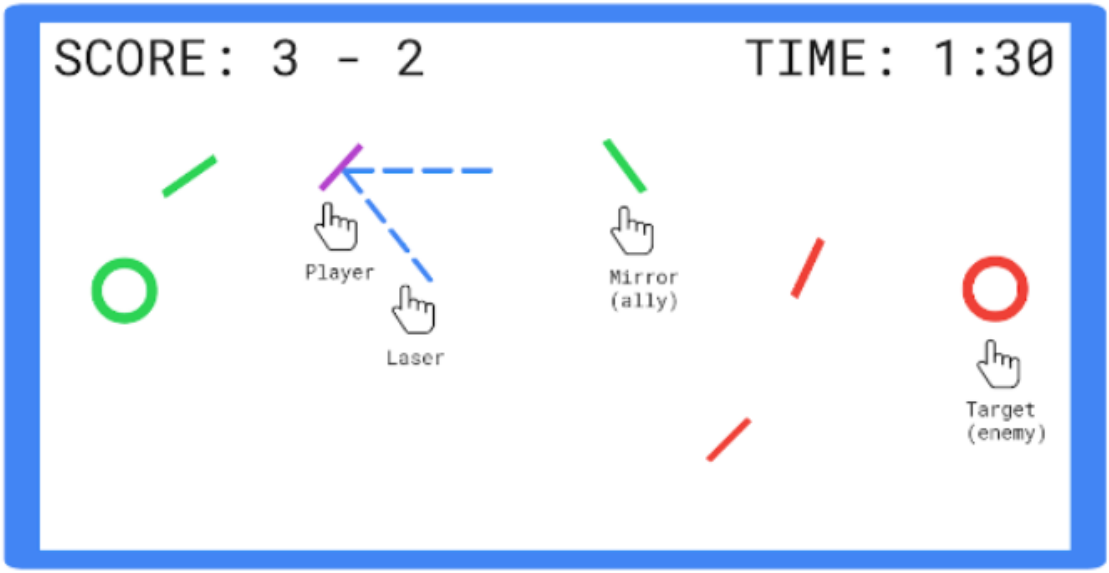


Figure 4.1: A mock-up of how the game is played.

Players each control a virtual mirror that the laser pulse will bounce off when the pulse's head collides with the mirror. The orientation of the mirror will bend the laser pulse in different directions through reflection. As such, players can guide the laser to the target of the opposing team. Currently the game is aimed at 2-6 players with teams consisting out of 1-3 players. The team that scored the most goals (a goal being the act of

shooting in the opposing team's target) after 5 minutes wins the match. If both teams happens to have the same amount of goals, the match ends in a tie.

4.2. New Additions to The Design

The new version of League of Lasers is different from its predecessor [46]. While the main game-play elements of the original have been maintained, the switch to AR has changed the game considerably. The game now uses a first person perspective as opposed to the old version's top down view. Player movement is tracked more easily through the HoloLens' integrated solutions, so no camera and sombreros are needed. Moreover, the change to a first person view has also changed the movement of the laser: in the old version this movement was on a two dimensional horizontal plane, while in the new version the laser moves around in full three dimensional space. One of the more important new features is that players no longer need to look down on their phones to get an overview of the game, and instead focus on looking around them, making them more aware of their surroundings and thus creating a safer and more pleasant experience This also means that the game can be played at a much faster pace.

4.3. Game Phases

In this section the different phases of the game are discussed. The game consists out of several phases, each contributing to either to the setup of the game, playing the game or ending the game. The different phases are explained below.

4.3.1. Start-up Phases

1. On start-up of the server the amount of players will need to be specified.
2. Players can join the game, each will be assigned a team by the server.
3. The player that joins first is marked as the master client.
4. The master client can now place the anchors (see section 7.2.1) in the game.
5. After placing the anchors, the master client exports these anchors through a button and uploads them to the web server.
6. The other player now download the anchors.
7. Once all players have joined and downloaded the needed anchors the game starts.

4.3.2. Play Phases

1. The timer counts down from 10 seconds, to make sure everybody is ready.
2. The game starts: the timer is set to count down from 5 minutes and a laser pulse is spawned on the network.
3. The laser pulse's position is set to the centre of the playfield and it is given a random horizontal direction.
4. If a team scores a point by guiding the laser into the opposing team's target, the team's score is incremented by one and the game resumes at the previous phase (Play phase 3).
5. Eventually the time runs out. If both teams have the same score, it will end in a tie, otherwise the leading team wins.

4.3.3. End Phases

1. There will be a waiting time of 10 seconds after the match, for players to see who has won. After this, the game starts over from the first play phase.

5

Global Network Architecture

The game's networking is based on UNET, the default networking solution embedded in Unity. The choice was made to use a dedicated server-client architecture, with the server being a traditional desktop application and the HoloLenses being the clients. The dedicated server was chosen as hosting on the HoloLens was found to be a frustrating process. The hardware is poorly optimised for hosting servers and connecting to these servers seems impossible most of the time (See section 3.4). Another major advantage to using the dedicated server is that its representation of the game world can be projected or shown on monitors, this way the audience can get a clear overview of the game played.

Due to some challenges related to synchronising spatial positioning between HoloLenses, the clients will get more authority than in a traditional server-client application. In fact, the server will rely on one client in particular, called the master client, to determine its internal representation of the game world, that will be shared with other clients. In addition to the aforementioned server, another server is run (see figure 5.1 for

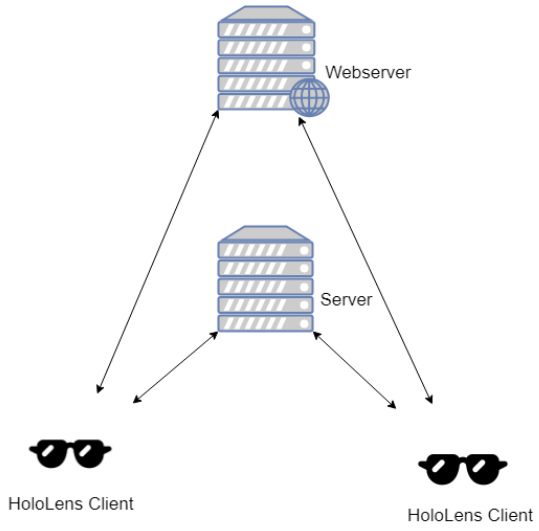


Figure 5.1: An overview of the how clients communicate with the servers.

an overview of the services). This is the web server which is used to upload and download files to and from clients (serialised anchors, see section 5.1 for more information). The web server can also be used as a means of network discovery.

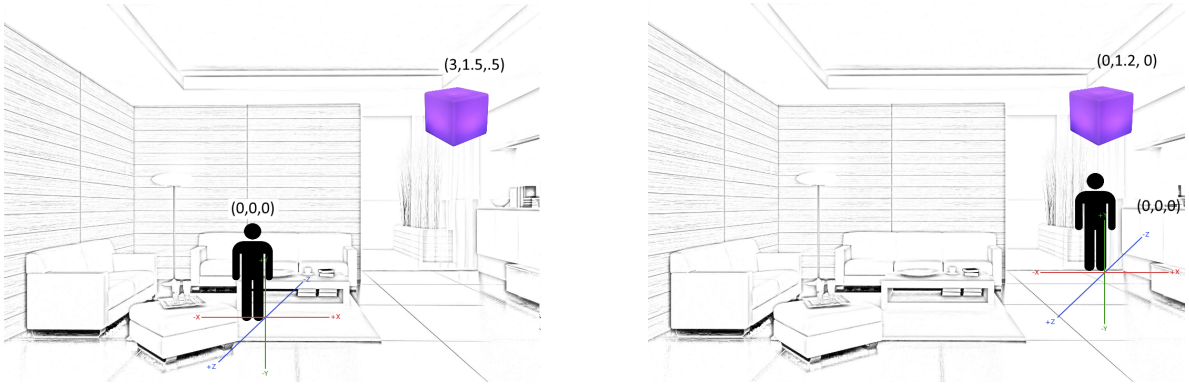
In the next sections, the aforementioned systems will be discussed and the design choices will be elaborated upon, starting with the server, then the master client, describing the process of anchor parenting, and finally the web server will be discussed.

5.1. Server

League of Lasers has some unique challenges that need to be solved. One of these is that the real world positions of virtual objects should be the same for all players. To ensure this, the server, which will control objects, such as targets, and handles the initialisation of networked objects, needs to have an internal representation of the game world which matches the real world. The server must communicate this to the client devices as well.

HoloLens applications, made in Unity, are projected in the space around the user. By default the objects will be placed relative to the position of the user and not relative to the position of the room the user is in. In essence, this results in the user's starting location as being the origin position in the Unity application. All in all, the user's starting position is the sole factor in determining the position of the projected holograms and replicating the exact same position (corrected for drift, see appendix E) is nearly impossible. In practice, this results in large offsets in the observed world between different devices and play-sessions, meaning that holographic objects are never displayed at the same real-world position.

The HoloLens is able to create spatial anchors, which "anchors" a hologram to a real-world position (see section 7.2.1). Objects can be "parented" to such an anchor, which makes the position, rotation and scale of the object relative to the anchor's position, rotation and scale, essentially using the object parented to as the origin for the parented object. The aforementioned anchors can be serialised and shared between HoloLenses so that multiple players can see holograms in the same real-world position. Sharing of these



(a) (b)

Figure 5.2: The position of objects in the game when the game is started at different real-world locations, using anchors.

anchors can be done via the dedicated server, which would seem to solve the aforementioned synchronisation issue, however this does not work in our case. While the anchors themselves consistently lock to their real-world position, their location within a Unity scene will differ per device and per session.

The use of anchors will only move objects to the correct real-world location and not offset the internal coordinate system in a Unity scene. This means that a cube on device 1 can be at position (a vector with the x,y and z coordinates) [3, 1.5, 0.5] (see figure 5.2a) and at position [0, 1.2, 0] (see figure 5.2b) on device 2, while being located at the same real-world position on both devices.

Another issue is that the desktop computer that runs the server does not know the concept of spatial anchors. This makes sense, since the desktop application does not have the spatial perception capability and thus cannot place a spatial anchor. In essence this means that the server has no clue where the anchor is located and thus has no knowledge of the scene as observed by the clients.

5.2. Master Client

As discussed before, game objects that need to be shown at the same location across devices need to be parented to an anchor, but the server cannot lock these anchors to a position, thus the server has no idea where the anchors are, relative from each other. In order to grant the server this knowledge the master client is used. The master client is the first client that connects to the server, this client will place the required anchors to build the "playfield". The playfield needs to be setup in such a way that the anchors are spread out somewhat evenly. The game starts with presenting the master client with three anchors, two of these need to be placed on the floor and one in the air (see figure 5.3). The aerial anchor must be placed right of

Figure 5.3: An overview of how the playfield is set up using anchors.

both ground anchors. The distance between the anchors on the floor represent the length of the field, the aerial anchor is used to determine the height of the playing field. The distance between the aerial anchor and the closest ground anchor is used to determine the width of the playing field. The rotation of the field is determined by the placement of the two ground anchors.

Preferably a distance of approximately 3-5 meters should be present between all anchors. Anchors should be placed at unique and easily identifiable positions: it is wise to avoid using dark, reflective or white surfaces to place the anchors on [43], as these will not be recognised so easily by the HoloLens, causing anchor drifting.

After the anchors have been placed, their positions in the Unity scene will be sent to the server. The server will then change the internal position of the anchors to the received positions, giving the server a representation of the master client's world. After this the anchors are serialised, as a byte array, by the master client and send to the web server, after which the anchors can be downloaded by other clients. More information about the interaction between the clients, the game server and the web server can be found in section 5.4.

5.3. Anchor Parenting

As discussed before (in section 5.1), objects that need to be shown at the same location across devices need to be parented to an anchor. As the server only knows what the master client sees, an object cannot be moved to another location via the network using Unity's Network Transform, as the coordinate system within each client is offset. The solution is to instead move objects relative to an anchor, which ensures that the position and orientation of objects is synchronised correctly between all clients. This means that all networked objects should be parented to an anchor at all times. The Microsoft documentation shows that anchors tend to only be accurate in a 3 meter radius [43]. To ensure accuracy, multiple anchors will thus be needed and networked objects will need to switch the anchor they are parented to if they move outside of the accurate range of an anchor. Networked objects will parent themselves to the nearest anchor (see figure 5.4) and message this change to the clients and server (whoever has authority over the game object), so that these can adjust the relative position based on the new anchor. Anchor parenting and relative positions synchronise objects, but players seeing objects parented to anchors further than 5 meters away will sometimes see objects drifting (see appendix E) together with their anchors. Unfortunately, at the time of writing, no solution to this issue has been found, but since accuracy is mostly needed in close range contexts (e.g. bouncing off the laser with the player's mirror) the choice to optimise for close distance accuracy was made.

5.4. Web Server

If objects are to be placed in their appropriate real-world location, it is essential that anchor files are shared efficiently and most importantly without any data corruption. To achieve this, an external web server was designed to share anchor files between HoloLenses. The web server also allows for discovering active games and registering new game sessions. In this section, a brief outline will be given of its implementation and inner-workings. Starting with an overview of the design and technology used, followed by an explanation on how to create a new game session. Finally the anchor sharing functionality is examined.

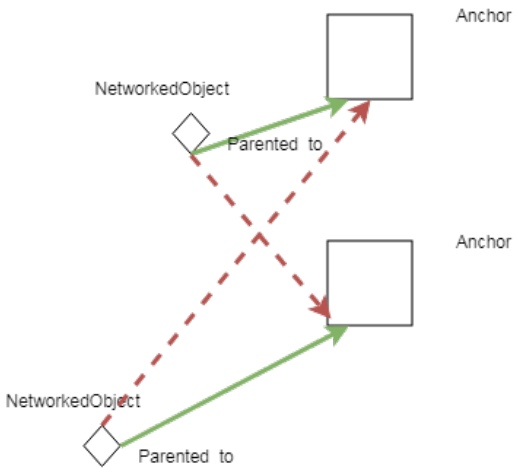


Figure 5.4: A diagram showing the functioning of the anchor parenting system. Green arrows mean that the anchor is parented to that anchor, red lines mean that the distance is too large and thus not parented

5.4.1. Design & Technology

The web server will hosted on a remote cloud server, which makes setting up a game easy as players are not required to setup their local web server before playing. However, the game should still be playable when no internet is available, hence it should be possible for any user to run the server on their local network. After the team experimented with several technologies, the choice was eventually made for *GoLang* (Go). Go has a solid standard library, including a built-in http server, and can be cross-compiled into an one-click executable for all major platforms [18].

5.4.2. Sharing Anchors

The web server makes use of several *Representational State Transfer* (REST) endpoint which allows for the creation of both anchors and games. The process of sharing anchors has is displayed in figure 5.5. Before the game starts, one client will become the 'master client', meaning that it will be responsible for placing and serialising the spatial anchors. After the serialisation has been completed, the master client uploads an byte array, which contains one or more serialised spatial anchors, to the web server (1). The web server stores the anchor data and provides the master client with an unique integer id to identify the anchor array (2). The master client will now sync this id over UNET to the Unity game server which runs on the local network (3). The Unity server then syncs that id to all other clients in the game (4). Using this id, the clients are now able to download the anchors from the web server (5, 6).

5.5. Network Discovery

This section briefly discusses the inner workings of network discovery, which was used to make it easy for the clients to connect to the game server. To join a game it is essential to obtain the *Internet Protocol* (IP) of the local game server. However, as discussed in section 3.4, the team was not able to make Unity's built-in network discovery work on the HoloLens. Therefore, the web server was to be used for network discovery. However, on the last day of writing this report, a HoloLens update was pushed that would fix some of these issues. Unfortunately, the update did not fix network discovery, however, after this update, the team was able to listen for UDP multicast messages on the HoloLens.

Previously, the team was not able to receive UDP multicast message, but now with the new update, UDP sockets are able to do so, but only if you write a message to them first (this worked on all HoloLenses available to us). This message contains a predefined string and will be ignored when it is received from itself or other HoloLenses. Using the fact that network discovery does work when running on the server (running regular Windows 10 using UDP multicast), the sender of these messages can be determined and thus the original IP of the server is found. It is then possible for the HoloLens to join the local game instance.

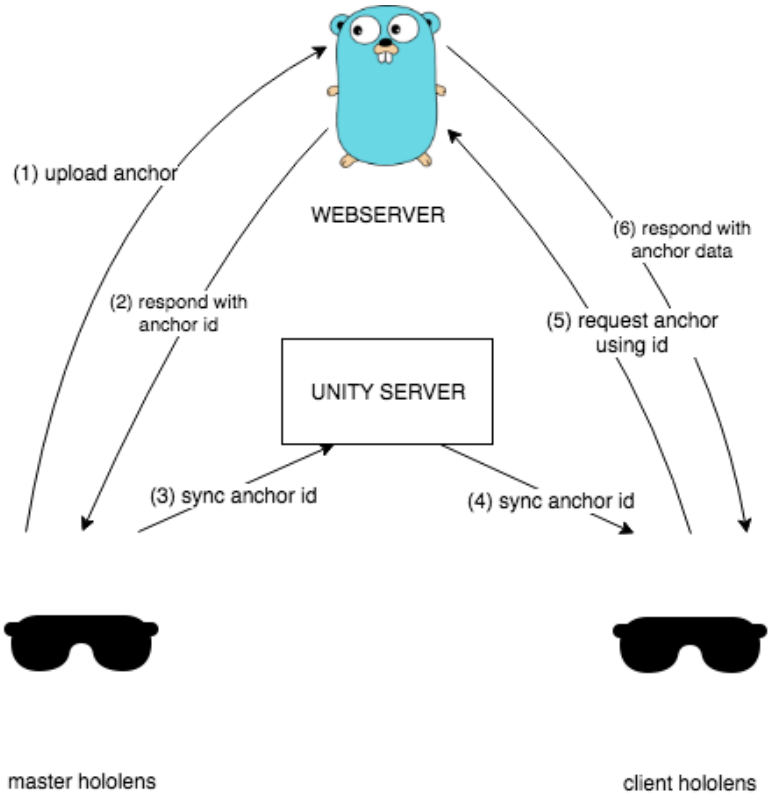


Figure 5.5: Process of sharing an anchor.

6 Software Quality

In this chapter an overview is given how we guaranteed that the software written for this project was of high quality. It will explain how our software was tested (including the custom CI solution) and which tools were used to analyse the code. Then the playtests and live demos and the feedback received there are described. Finally the SIG feedback is discussed.

6.1. Testing

To ensure the software written works as intended, it important to test the software rigorously. In game development unit testing can be difficult and can not be used to verify if a gameplay element or mechanic is fun and engaging, therefor playtests were also conducted. This section will give an overview of unit testing in Unity and how we organised our playtests.

6.1.1. Humble Object Pattern

Testing (unit testing in particular) is non-trivial in Unity, because of MonoBehaviours [32]. These MonoBehaviour classes are a core component of Unity, as they control the game objects and run in the game world. It is the base class from which every Unity script derives, providing access to Unity's API [39]. Due to performance reasons, these MonoBehaviours are not implemented with interfaces or inheritance, but they get invoked from Unity's *C++* code [32]. MonoBehaviours can not be instantiated, so they can not be mocked using the mocking frameworks out there.

This means the logic in MonoBehaviour scrips cannot be unit tested. Therefore, a special design pattern has been recommended by Unity's development team, called the Humble Object Pattern. The Humble Object Pattern works by moving all logic from the MonoBehaviour script to a separate controller script. The MonoBehaviour script is then called from the controller, using interfaces, to only execute Unity API calls, but all the logic is contained in the controller. Since the calls the controller makes to the MonoBehaviour script are made using interfaces, the logic can be easily tested by instantiating the controller class and providing it mocks for the MonoBehaviour instances that implement the aforementioned interfaces. The Humble Object Pattern is visualised in figure 6.1.

The Humble Object Pattern thus provides testability for classes that normally wouldn't be testable. The major downside of this pattern is that it is not always trivial to split the logic from Unity API calls. This gives the code implementation of features requiring this pattern quite a bit of overhead.

6.1.2. Unit Testing

As mentioned in section 6.1.1, we use the Humble Object Pattern to unit test our game's logic. Using interfaces, calls to MonoBehaviours are verified using mocks. We use NSubstitute for mocking objects.

Unity can run unit tests in the so called Unity Test Runner, which uses a Unity integration of the NUnit library [40]. The Test Runner is a window in the editor that gives the option to run the selected *C#* test methods (indicated using NUnit Test attributes) and will visualise whether the tests succeeded or failed. Since the Test Runner is an integrated component in Unity, it can be ran in Unity Cloud Build (more about Unity Cloud Build will be explained in section 6.2.2).

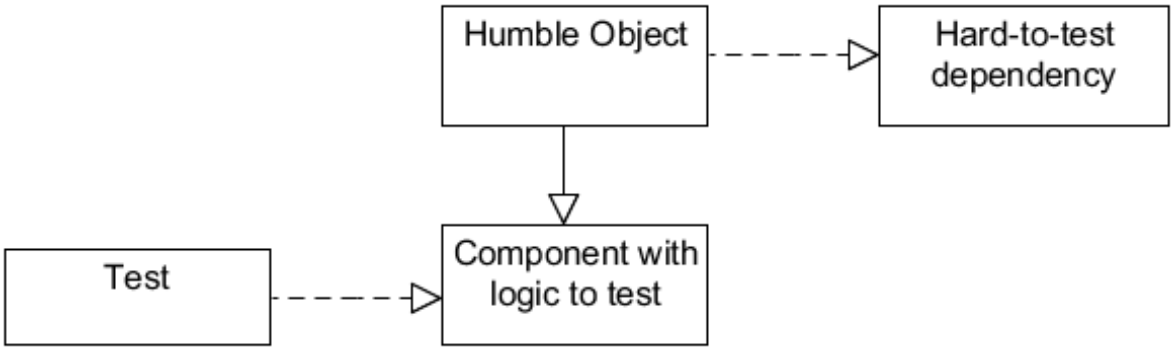


Figure 6.1: The Humble Object Pattern [32].

One problem we encountered using Unity's Test Runner was that in Unity 2018.1, the Test Runner was unable to find our test classes. This is an issue more people on the Unity Forums had [2]. Our solution was to downgrade to Unity 2017.4.3f1.

6.1.3. Parameterised Tests

Aside from regular NUnit tests, parameterised tests are used. These are test methods where the input data is split off from the test method body and given to it via attributes that are placed above the test method. This makes it very easy to test methods on meaningful arguments in their argument domains, a methodology also known as boundary value testing [28], where simply the values causing state or behaviour transitions on changes are checked, instead of all possible or arbitrary values. In addition to this, maintainability is enhanced due to the ease of adding additional test data, which just requires specifying a new NUnit TestCase attribute (see figure 6.2) with the relevant data, without altering the test body or adding an entirely new test method with nearly duplicate code. In our case, all tests using the TestCase attribute have a descriptive name string as the first test parameter. This name string is not used in the test, but it makes the test easy to identify in the test runner, as it briefly describes the purpose of the test case.

```
/// <param name="started">
/// Indicates whether the match should've started.
/// </param>
[TestCase("default_time_started_state_before_end", DefaultDuration, DefaultCountdownTime, DefaultPostmatchTime,
    DefaultDuration - 1, true)]
[TestCase("default_time_started_state_on_end", DefaultDuration, DefaultCountdownTime, DefaultPostmatchTime,
    DefaultDuration, false)]
[TestCase("default_time_started_state_after_end",DefaultDuration,DefaultCountdownTime,DefaultPostmatchTime,
    DefaultDuration + 1, false)]
public void TestTimerStartStateTransitionsCorrectlyToFalseOnMatchEnd(
    string name, int duration, int countdownTime, int postmatchTime, int time, bool started)
{
    var timeController = Substitute.For<IMatchTimeController>();
    var timerController = new TimerController(timeController, duration, countdownTime, postmatchTime) { Time = time, Started = true };
    timerController.CheckMatchHasEnded();

    Assert.AreEqual(started, timerController.Started);
}
```

Figure 6.2: NUnit TestCase attribute tests for the timer.

The downside of using TestCase attributes in NUnit is that the parameters can exclusively be static immutable objects, such as the base types and arrays of them. This is due to the way input values in .NET attributes work [5]. A solution to this, which was discovered later, was to make use of NUnit's TestCaseSource attribute (see figure 6.3), where a static object array (the test cases) containing object arrays (the parameters per test case) can be referenced. These can also have their names set directly, instead of requiring a string as first parameter. This feature was not used everywhere in testing due to its later discovery, but will be implemented during future maintenance.

6.1.4. Playtests

In this section, an outline will be given about how playtests were organised in order to retrieve valuable feedback from players.

```
/// <summary>
/// Gets the test cases for the NearPlayfield tests with the center of the playfield with an XYZ offset.
/// </summary>
[UsedImplicitly]
private static IEnumerable<TestCaseData> PositionsToCheckIfNearPlayfieldAtCenterWithXyzOffset
{
    get
    {
        Vector3 center = new Vector3(2.5f, 1.5f, 5);
        yield return new TestCaseData(center, true).SetName("CheckIfCenterPositionIsNearPlayfield");
        yield return new TestCaseData(center + Vector3.up, true).SetName("CheckIfUpPositionIsNearPlayfield");
        yield return new TestCaseData(center + Vector3.down, true).SetName("CheckIfDownPositionIsNearPlayfield");
        yield return new TestCaseData(center + Vector3.right, true).SetName("CheckIfRightPositionIsNearPlayfield");
        yield return new TestCaseData(center + Vector3.left, true).SetName("CheckIfLeftPositionIsNearPlayfield");
        yield return new TestCaseData(center + Vector3.forward, true).SetName("CheckIfForwardPositionIsNearPlayfield");
        yield return new TestCaseData(center + Vector3.back, true).SetName("CheckIfBackPositionIsNearPlayfield");
        yield return new TestCaseData(center + new Vector3(2.5f, 0, 0), true).SetName("CheckIfRightBorderOfPlayfieldIsStillNearPlayfield");
        yield return new TestCaseData(center + new Vector3(0, 0, 5), true).SetName("CheckIfFrontBorderOfPlayfieldIsStillNearPlayfield");
        yield return new TestCaseData(center + new Vector3(0, 1.5f, 0), true).SetName("CheckIfCeilingOfPlayfieldIsStillNearPlayfield");
        yield return new TestCaseData(center + new Vector3(2.5f, 1.5f, 5), true).SetName("CheckIfCornerOfPlayfieldIsStillNearPlayfield");
        yield return new TestCaseData(center + new Vector3(9.8f, 0, 0), true).SetName("CheckIfXLessThanThresholdIsNearPlayfield");
        yield return new TestCaseData(center + new Vector3(0, 9.8f, 0), true).SetName("CheckIfYLessThanThresholdIsNearPlayfield");
        yield return new TestCaseData(center + new Vector3(0, 0, 9.8f), true).SetName("CheckIfZLessThanThresholdIsNearPlayfield");
        yield return new TestCaseData(center + new Vector3(10, 0, 0), false).SetName("CheckIfXAtThresholdIsNotNearPlayfield");
        yield return new TestCaseData(center + new Vector3(0, 10, 0), false).SetName("CheckIfYAtThresholdIsNotNearPlayfield");
        yield return new TestCaseData(center + new Vector3(0, 0, 10), false).SetName("CheckIfZAtThresholdIsNotNearPlayfield");
    }
}
```

```
/// <summary>
/// Tests if the NearPlayfield method works correctly if the playfield's center has an XYZ offset.
/// </summary>
/// <param name="position">
/// The position to check if it is near the playfield.
/// </param>
/// <param name="expectedReturnValue">
/// The expected return of the NearPlayfield method with the given position.
/// </param>
[TestCaseSource(nameof(PositionsToCheckIfNearPlayfieldAtCenterWithXyzOffset))]
public void TestNearPlayfieldCenterWithXyzOffset(Vector3 position, bool expectedReturnValue)
{
    Vector3 bottomFrontPosition = new Vector3(0, 0, Dimensions.z);
    Vector3 bottomBackPosition = new Vector3(0, 0, 0);
    Vector3 topFrontPosition = new Vector3(Dimensions.x, Dimensions.y, Dimensions.z);

    this.playfieldController.InitPlayfield(bottomFrontPosition, bottomBackPosition, topFrontPosition, Dimensions.y / 2);
    Assert.AreEqual(expectedReturnValue, this.playfieldController.NearPlayfield(position));
}
```

Figure 6.3: NUnit TestCaseSource attribute/pattern used for testing the playfield.

Organisation

During the course of the project, roughly two kinds of playtests were conducted. Our team was located in the INSYGHTLab which has a decent amount of people passing through it. Since the HoloLens is relatively new technology it was not hard to regularly find people that were willing to try out the game and give us some feedback. During meetings with supervisors, the team also demoed (parts of) the game in this manner.

Demo at the Virtual Playground Event

Figure 6.4: Players enjoying a game at the Virtual Playground event.

League of Lasers' first big demo was organised at the Virtual Playground VR & AR Showcase event on the 21st of June [3]. The team was pleased to be invited to display League of Lasers to a large crowd with a specific interest in the field. During the event, the game was being played continuously (see figure 6.4), without any interruptions. The interest in the game went above the team's expectation. In fact, during the peak of the event, it was not uncommon for there to be a queue of people waiting to play the game. From the event, the team gained many valuable insights, most notably:

- The INSYGHTLab has limited space and therefore limits movement. League of Lasers is designed as a superhuman sport and when the available space accommodates it, we found that players quickly started to play it as such.
- League of Lasers is enjoyable for spectators. Even though the audience could only see the game via the spectator view (see section 7.1.9) displayed on a screen, they still enjoyed seeing people run and jump around.
- Part of the people that played the game also filled in a questionnaire (see appendix F and G for the questionnaire and its results) which gained us some valuable feedback. Including, but not limited to, the fact that for players it is not always clear what team they belong to. Also several players experienced that the laser passed through their mirror. We addressed and fixed these issues afterwards.

6.2. Continuous Integration and Unity

In this section, an overview will be given about how CI was used in conjunction with Unity and how this benefits the code review process. Secondly, some of the challenges faced implementing CI will be discussed.

6.2.1. Benefits of Continuous Integration

Continuous Integration (CI) is a powerful tool which can be used to build solutions and run tests in a clean and uniform environment, perform static code analysis, and more [6]. CI services such as Travis CI can be integrated into some platforms that facilitate pull-based development such as GitHub. In turn, CI becomes a particularly powerful tool in pull-based development since it allows reviewers additional security that merging *Pull Request*s (PRs) will not break build. The latter evaluation could be performed by the reviewer themselves on a local machine, however, the result between local machines might differ. Secondly, when multiple PR are opened simultaneously each merged PR will require the others to be re-evaluated once more; hence automating this process can save a lot of time.

6.2.2. Challenges Using CI and Unity

A brief outline will be given why it was almost impossible to get CI working for HoloLens in an elegant manner. Firstly, League of Lasers makes use of the Unity as build system. Since the Last few iterations of Unity, building solutions and running tests can only be done via the command line [45] (necessary for running in the CI environment) with an activated Unity editor, unless you are in the possession of a Unity Pro serial [34] which was far out of our budget. This meant that we could not activate the Unity editor remotely. Secondly, the HoloLens only runs *Universal Windows Platform* (UWP) apps ([26]), which can only be built on native Windows 10 [16] (Mono is not sufficient) hence a CI service supporting Windows was needed.

This prevented the usage of most CI services because command line building was required. We did find an alternative and relatively affordable option (9$/month) ([44]) called Unity Cloud Build. Which offers a managed platform to build a solution and run Unity's built-in editor/play-tests. Yet the design of this service is far from accommodating for pull-based development. To perform a build, a so-called build-configuration is required, which in turn required the git branch to be specified in advance. Hence, this requires the creation of a new build configuration for each PR and specification of the branch in question since branch names are not the same for each PR (and merged branches are generally removed). Additionally, Unity Cloud Build does not offer any native integration for GitHub, GitLab, etc. such that PRs can easily be blocked when the build fails. Luckily it does provide a solid API which we used extensively as described in the next section.

6.2.3. Our CI Solution

In our CI solution, Travis CI is essentially used as a wrapper CI service. Travis has near seamless integration with GitHub Pull Requests allowing us to easy block PRs that failed to pass the CI. Yet we cannot build or test our solution directly on Travis CI since it does not support building UWP apps nor do we possess a Unity Pro serial. Consequently, a script was created which uses the Unity Cloud Build API to generate a fresh build configuration for each pull request. This script runs on Travis CI and prompts the Unity Cloud Build API to start a new build and consequently poll its status until it has either succeeded or failed. Finally, the build logs are downloaded from Unity Cloud Build and echoed to Travis CI such that most information concerning the build can be reviewed straight from the Travis build logs.

6.3. Code Analysis Tools

The usage of (static) code analysis tools can be very helpful for maintaining high code quality and readability. During this project code analysis tools were extensively used. One of the tools we used was ReSharper, which performs on-the-fly code analysis [19]. ReSharper integrates well into Visual Studio and provides warnings for code smells. Instead of only warning about code quality issues, ReSharper will provide and automatically implement fixes at the user's discretion. ReSharper also contains various refactoring, code generation and code formatting tools.

ReSharper also has support for plugins. One of these plugins is StyleCop, which enforces style rules [10]. The rule set used was that of the context project of League of Lasers. Because StyleCop is a ReSharper plugin, the warnings StyleCop gives are integrated into ReSharper's code inspection. This means that ReSharper will not only notify us of code smells, but also of code style issues.

Since our CI runs via Unity Cloud Build, there is no possibility to run ReSharper (and thus also StyleCop) in our CI. This means the code warnings provided by ReSharper can not be check for, to fail the CI build.

The solution used was to make sure there were no code issues before creating a PR and that reviewers always pulled the branch of the PR and checked if it is indeed the case that the PR does not contain ReSharper warnings.

6.4. SIG Feedback

During the course of the project, there were two moments where code quality feedback was provided by SIG. In this section, we will discuss the feedback we were given and how we processed it. The feedback itself can be found in appendix A.

6.4.1. First Upload

For the first upload, we received a score of 4.4 out of 5, which according to SIG meant our code achieved above average maintainability. The feedback also mentioned that the only reason why we did not receive the highest score is due to lower scores for unit size en unit complexity. Hence, given the quality of code, SIG did not have any concrete recommendations apart from recommending us to keep up the quality standard.

The team was pleased with the result of the first SIG feedback. In short, there were no concrete suggestions for improvement and it was noted that the current state of the system was good. We suspect the pragmas used throughout the code base played a big part in the higher unit size and complexity

The team contacted Dennis Bijlsma whether he had any suggestions to improve upon unit size and unit complexity. He suggested that there should be no focus on marginal improvements, but instead, the current standard should be upheld since the code base was already in good shape. However, he did suggest running tools such a SonarQube or BetterCodeHub for the remainder of the project.

6.4.2. Better Code Hub

As suggested by Dennis Bijlsma, we ran BetterCodeHub on the code base to identify potential code issues and maintain the high code quality [17]. The team chose to use BetterCodeHub since SonarCube did not work well in conjunction with Unity (mainly due to all the magic annotations) [11]. Consequently, the team analysed the BetterCodeHub results and corrected most issues found. The word *most* is chosen here since not all issues found by BetterCodeHub are actually real problems. For instance, we obtained a lower score on coupling because the debugger class (which is the only way to do any kind of debugging on the HoloLens, see section 3.4.1) was coupled with almost all classes. The team did not see this as a real issue, however, it did prevent the score from becoming a 10 on BetterCodeHub, resulting in a 9/10 final score.

7

Implementations

During the development of the game many different components have been developed, this chapter gives a brief overview of the implementations of these components. The chapter starts with discussing the game object implementations, followed by the implementations that compose the anchoring system.

7.1. Game Object Implementations

In this subsection, the implementation details of the game objects will be discussed. With game objects, specifically the objects that somehow take part or control mechanics of the gameplay are meant. For all physical game objects, a brief summary of their implementation will be given, followed by physics details, then the details about their appearance are given, closing off with their behaviour on the network. For the non-physical game objects, a more generic description of their relevant implementation details will be given, since their implementations can vary wildly due to generally not making much use of Unity's built in components.

7.1.1. The Laser Pulse

The laser pulse is a dynamic physical game object that collides meaningfully (the collision has an effect on the game) with any other physical game object in the playfield. Its implementation is mostly based on Unity components, with some minor script-driven exceptions to make it behave according to specification, see appendix C. The laser pulse should bounce off any other physical object except for the targets, where it should be reset to the centre of the playfield with a random velocity. This velocity should be constant at all times, unless influenced by external forces such as power-ups. See figure 7.1 for an overview of the Laser Pulse's code implementation.

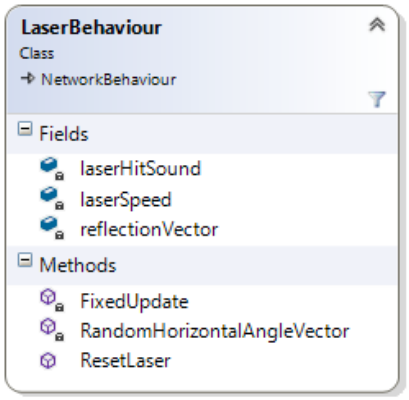
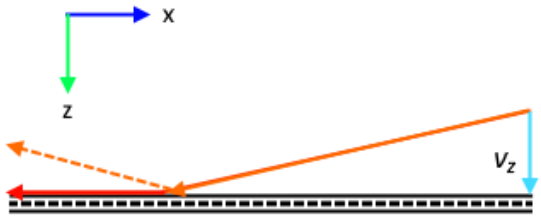


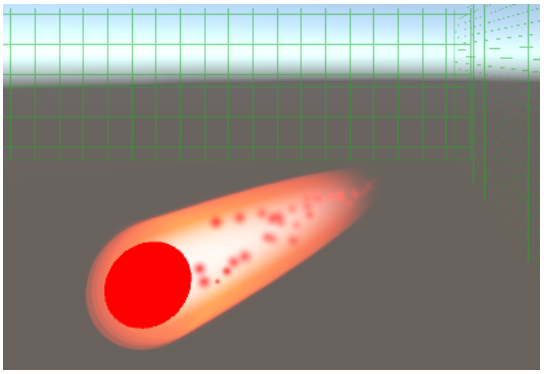
Figure 7.1: Overview of the laser pulse's code implementation in *Unified Modelling Language* (UML).

In Unity, only one of the two objects involved in a collision require a rigidbody component. In League of Lasers, the laser pulse carries this responsibility, since all meaningful collisions in the game involve the laser

pulse. Some subtle tweaks had to be made in order for the laser pulse to behave correctly. First of all, the "bounce threshold" setting in the Unity physics settings, which disables bouncing if the incoming velocity, of the colliding object, perpendicular to the bouncing surface is too low (see figure 7.2a), had to be put to zero, since this behaviour is undesirable for the laser pulse: it should always bounce. The second tweak involved all other physical objects that would collide with the laser pulse: these required a physics material that has minimum friction (0) and maximum bounce (1), along with ensuring that the combination settings always picked the maximum bounce value of two colliders in a collision and the minimum friction value, otherwise the reflection angle would not be correct for a "laser pulse". The only object the laser pulse should not bounce with is the target. On collision with a target, the laser pulse is reset to the centre of the playfield instead, with a new random horizontal direction.



(a) No bouncing happens when the perpendicular velocity component v_z is too small, resulting in the laser pulse incorrectly following the red vector's path.



(b) The laser pulse's visual

Figure 7.2

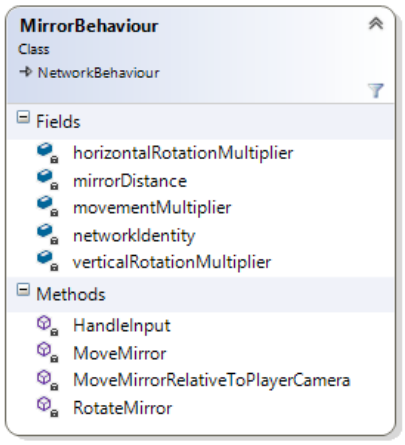
With these tweaks, the laser pulse can still show undesirable behaviour: its velocity could be altered and it could even come to a stand-still. For this, additional code was written in Unity's FixedUpdate method that would keep the laser pulse's velocity constant by normalising its velocity vector and multiplying it with a predefined value. After this, it could still occur that the laser pulse would stand still due to synchronisation issues between the physics thread and the game loop thread, causing normalisation of the velocity vector to give a zero vector. To combat this, the laser pulse's reflection vector after a collision was tracked and updated every time. If the velocity would ever become zero, this reflection vector multiplied with the predefined laser pulse speed value was used to make the laser pulse move again. After this fix, no anomalies in the laser pulse's behaviour were found anymore. Checks are also present to keep the laser pulse present in the playfield. Should the laser pulse ever misbehave and leave the playfield due to buggy physics, it will be reset to the centre of the playfield with a random horizontal direction.

The laser pulse is displayed as a sphere that uses a shade-less red material, rendered early in the render queue. It features a particle trail with particles using additive rendering and a sharp circular gradient, that appear over the sphere. Another particle trail with smaller shade-less particles converging at the centre gives it a nice finishing touch, while also maintaining visibility. See figure 7.2b for the laser pulse's visuals.

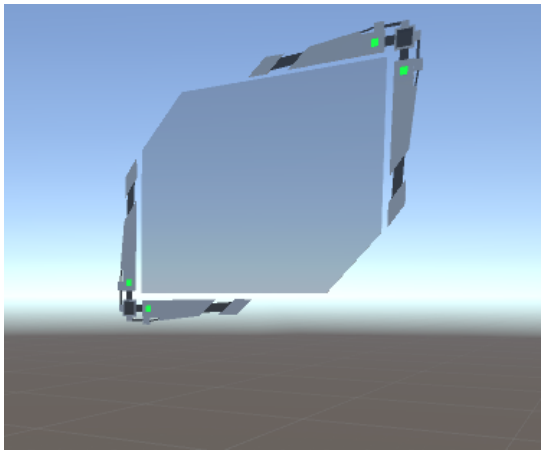
The laser pulse's movement is networked through our custom network transform, which makes use of anchors. Since the laser pulse is a dynamic object, it will dynamically attach itself to the closest anchor as it moves around. The laser pulse's movement is fully controlled by the server, hence its script needs to inherit from NetworkBehaviour to ensure the laser pulse's movement is only being managed on the server. The movement of the laser pulse is interpolated when it moves in a straight line. The combination of this and a high send rate for its position make the game feel smooth and responsive for the clients.

7.1.2. The Mirror

The mirror is a dynamic physical game object that is entirely controlled by the player and only collides meaningfully with the laser pulse, bouncing it off. It is almost exclusively implemented using Unity components. Some code is present in order for it to correctly position itself in front of the camera while rotating and moving along with it. Additionally, some input code intended for debugging is present, providing keyboard controls for movement and rotation of the mirror. See figure 7.3a for an overview of the mirror's code implementation.



(a) Overview of the mirror's code implementation in UML.



(b) The mirror's 3d model

Figure 7.3

The mirror is box-shaped, with the thin axis (the flat face) pointing to the player. The mirror has a 3d model that looks like a floating screen (see figure 7.3b). The borders, of the model, are opaque while the screen of the mirror is a semi-transparent material, so that HoloLens users can look through it to see the rest of the game. The colour of the highlights at the front is changed depending on the team of the player that owns the mirror.

It uses a box-collider that is slightly thicker than the mirror's own appearance, to make the collision detection with the laser pulse more robust. The collider has a physics material with friction 0 and bounciness 1, while taking maximum bounciness and minimum friction as combined values for collisions.

Since the mirror is a dynamic object, it is also networked through our custom network transform (see section 7.2.2) that makes use of anchors, in the same manner as the laser pulse. Like the laser pulse, the mirror also has a high send rate, enhancing the responsiveness of the game on the clients.

7.1.3. The Wall

The wall is a stationary physical game object that can collide meaningfully with the laser pulse. It is spawned six times by the Playfield game object with different dimensions to fully encase the entire field according to the given parameters. This may make the name somewhat misleading, since the wall also functions as floor and as ceiling. By itself it contains some code, but mainly consists of Unity components, though some *High-Level Shader Language* (HLSL) code was written as well for its grid shader. See figure 7.4 for an overview of the wall's code implementation.

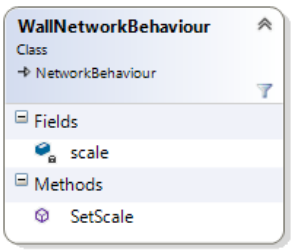


Figure 7.4: Overview of the wall's code implementation in UML.

The wall is box-shaped, its collider fully matching the visual representation. A volumetric object was used instead of a plane to make collision detection more robust. It uses the same physics material as the mirror to ensure correct bouncing behaviour of the laser pulse. The transparent grid shader was written specifically for the wall and is used by its material. It allows configuration of grid line thickness, colour and scaling through the shader's property settings, so that it can be easily configured to the needs of the game.

Since transform scaling is not automatically networked, some additional synchronisation code is present

so that it may be scaled correctly by the playfield game object for clients (see figure 7.4). Hence, the script for this must inherit from NetworkBehaviour.

7.1.4. The Target

The target is a stationary physical game object that can collide meaningfully with the laser pulse. The playfield game object spawn two targets, one for each team. The target has some additional behaviour code for handling collisions with the laser pulse, synchronising its team and setting its material colour according to the team. See figure 7.5 for an overview of the target's code implementation.

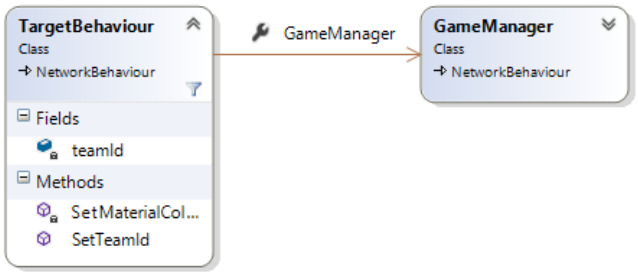


Figure 7.5: Overview of the target's code implementation in UML.

The target is cylindrical in shape, and uses a mesh collider for the collision detection. The two targets are placed at the centre of the adjacent short sides of the playfield. They will have the colour of their corresponding team.

When the laser pulse collides with the target, it resets itself to the centre of the playfield with a random horizontal direction, hence the target does not require a physics material, since no meaningful bouncing behaviour can take place. The collision check for this takes place in the target, since the target carries the team information, which is required for determining who has scored. On collision, the target increments the goals made against the team it belongs to in the GameManager, as can be seen in figure 7.5, and also resets the laser pulse by calling its reset method.

7.1.5. The Playfield

The playfield is a non-physical manager style game object, responsible for organising the playfield. The playfield concerns itself with all objects physical objects that, together, form the playfield, such as the walls and the targets, and will instantiate and spawn these over the network at their appropriate locations. See figure 7.6 for an overview of the playfield's code implementation.

The locations and scaling of the walls are fully determined by the playfield size, offset and wall thickness prefab parameters. The targets are positioned at both ends of the playfield, such that they still stick out on the inside of the playfield. The walls are put around an imaginary box sized as the playfield's dimensions at the position of the given offset. They are scaled in a way that makes them as thick as the given wall thickness along the axis they belong to, and cover the entire face of the imaginary box. This creates an enclosed area for the laser pulse to be contained in.

In addition to this, the script for the playfield inherits from NetworkBehaviour, since the instantiation of objects only needs to take place on the server. They automatically get synchronised to the clients and linked correctly to the server's instance via the network spawn.

7.1.6. The Game Manager

The game manager is a non-physical manager style game object, responsible for managing everything related to getting the game and the matches going. For this to happen, it is necessary to spawn in a playfield (which will spawn in the walls and the targets), a laser pulse, and a timer. In addition to this, it also keeps track of the goals made in the targets of the two teams through a dictionary and it has controls for match flow (prepare a match, start a match, end a match), which are used as callbacks in the timer. These methods will control when the laser pulse should be spawned or destroyed and when scores should be reset and a winner should be chosen. See figure 7.7 for an overview of the game manager's code implementation.

Like the playfield, the game manager's script also inherits from NetworkBehaviour, to ensure that the objects get instantiated on the server alone and then get synchronised to the clients.

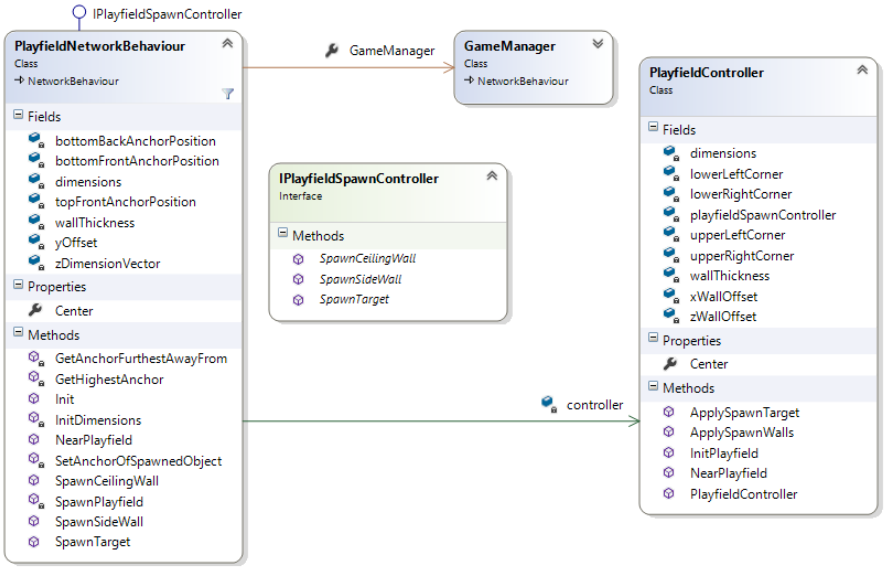


Figure 7.6: Overview of the playfield's code implementation in UML.

7.1.7. The Player

The Player is a non-physical game object, serving as the game object that relates itself to connecting clients on the server. The Player is a NetworkBehaviour, meaning that it will be spawned on both the client and the server. A special property of the player is that it has local authority over itself by default, allowing clients to move the object without further permission requests towards the server. In our implementation, the Player carries some additional information, such as the team it belongs to. It also triggers the spawning of the Mirror associated with the client when the Player is spawned on the network. See figure 7.8 for an overview of the Player's code implementation.

An important role of the player in our implementation is the assignment of the master client (see section 5.4.2), which can set up the playfield and export it to the other clients, while also providing the server with a properly synchronised overview of the game. The first client that joins will be labelled as master client through the PlayerBehaviour. Any client that joins afterwards will be a normal client, and has to wait for the master client to finish setting up the playfield.

7.1.8. The Timer

The timer is a non-physical game object responsible for handling events related to match time. Internally, the timer has three states: pre-match, match, and post-match. These states are represented by the value of a time integer and a "started" boolean, indicating whether a match has started. The pre-match time, match duration, and post-match time can be pre-configured in the prefab carrying the behaviour script. See figure 7.9 for an overview of the timer's code implementation.

At the appropriate state switches, callbacks obtained from the game manager are called to control the game's flow. During the pre-match time, no laser pulse is present, it only spawns a laser pulse when the game transitions from pre-match to a state where the match has started. Then, when the match is over, a callback is used to despawn the laser pulse and a winner is determined on the transition from the started match state to the post-match state. After this latter state, another callback ensures that the game is reset and prepared for the next match.

The time variable is incremented by adding up Unity's delta time, which is the amount of time that has passed between two frames, and then incrementing its value if this amount of passed time exceeds 1, resetting the passed time back to 0 again. This only happens on the server, which is why the behaviour script inherits from NetworkBehaviour. Additionally, the time integer is a SyncVar, which ensures that it is synchronised between server and all clients. For testability concerns, most of the timer's logic resides in the accompanying controller class. The behaviour class only contains Unity-interfacing calls, or variables that need to be network-synchronised, such as the time integer.

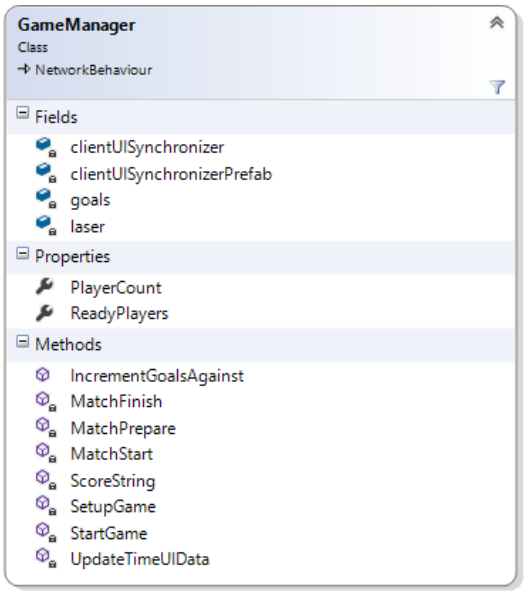


Figure 7.7: Overview of the game manager's code implementation in UML.

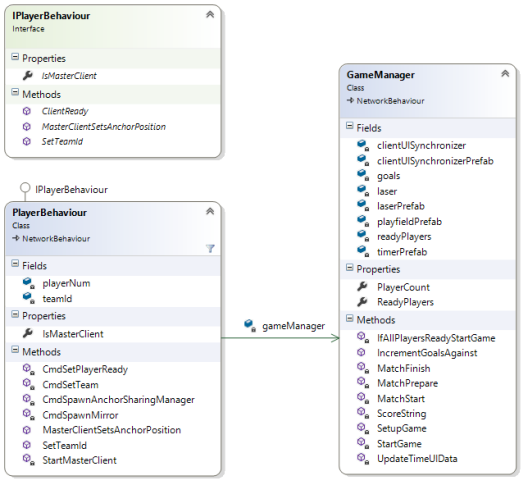


Figure 7.8: Overview of the player's code implementation in UML.

7.1.9. Spectator Mode

Spectator mode was developed for the audience to see an overview of the game. The spectator mode runs on the server (see figure 7.10), for the reason that adding a separate observer client was redundant. Moreover, running the spectator mode on the server also allows for easy debugging. The component uses the CameraSpectatorBehaviour script, which implements the humble object pattern (see section: 6.1.1). The script places the camera above the playfield and provides an overview of the game. See figure 7.11 for an overview of the spectator view's code implementation.

7.2. Anchor Related Implementations

In this section the implementations related to anchoring are discussed. Some of the discussed components are present on others components, such as the AnchorRelativeNetworkTransform, but their importance makes it necessary to explain them in more detail. Other components aid the game in transferring anchor information or managing their use within the game. This section will start with a brief explanation of the anchor game object and will then explain how the anchors are used to derive positions using the AnchorRelativeNetworkTransform, followed by sections about the anchor sharing and management system and the

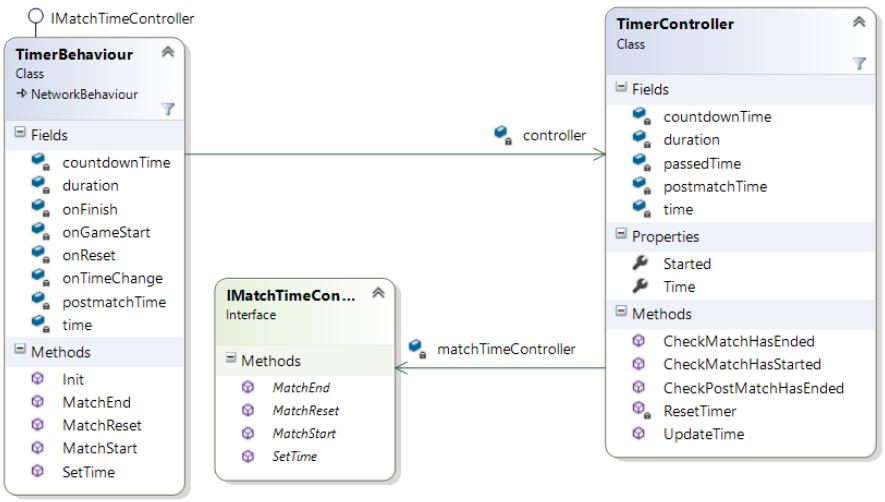


Figure 7.9: Overview of the timer's code implementation in UML.

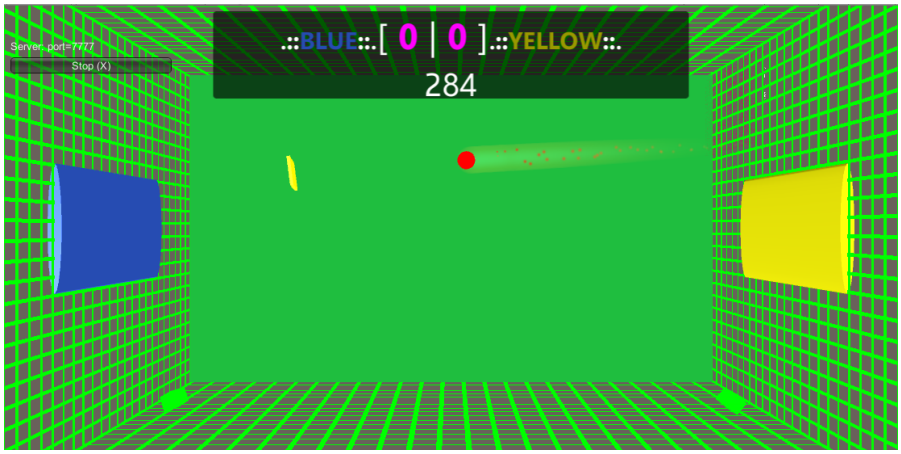


Figure 7.10: A screenshot of the server-side spectator mode.

components that comprise it: the anchor manager, the anchor sharing manager, the file manager for local storage and the web service used to communicate with the web server. See figure 7.12 for an overview of the entire implementation of the localisation and anchor management framework.

7.2.1. The Anchors

Anchors, called spatial anchors by Microsoft [43] and world anchors by Unity [38], represent an important point in the real world. These anchors' position can be locked in physical space. Each anchor has a coordinate system that ensures that anchored holograms stay precisely in place. Game objects can be parented to an anchor to position them relative to that anchor. Holograms parented to anchors are accurate up to 3 meters, otherwise the holograms will have notable positional errors proportional to their distance to the anchor. Anchors can be shared between HoloLenses and can thus be used to ensure all players see the holograms on exactly the same real-world position.

In League of Lasers the master client will see the anchors as cubes. Using a modified version of the Mixed Reality Toolkit's tap to place script, the master client can place the anchors at the correct real-world positions. Then the anchors are serialised and shared between all clients. Game objects parented to these anchor game objects will use this anchor to position themselves relative to it.

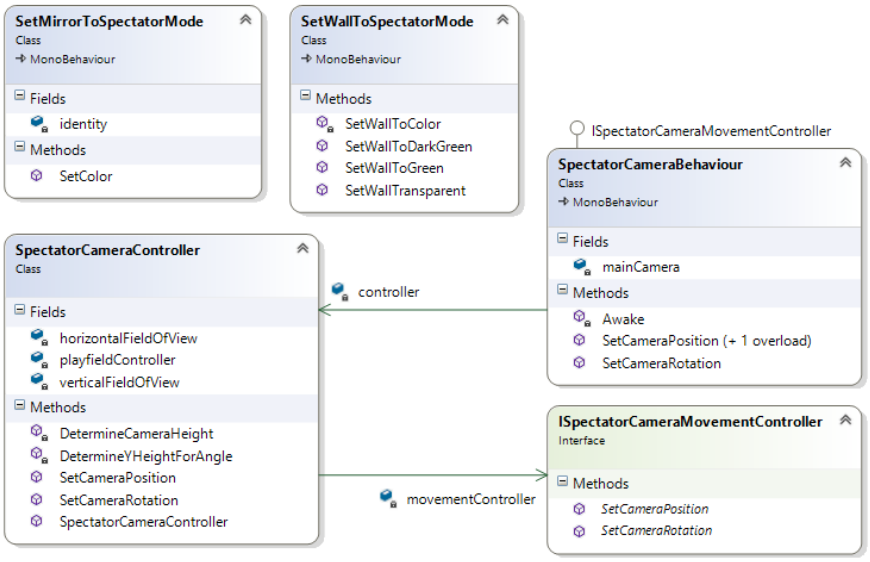


Figure 7.11: Overview of the spectator view's code implementation in UML.

7.2.2. The Anchored Relative Network Transform

The AnchorRelativeNetworkTransform component is the backbone of the application's localisation framework. Without this component, holograms cannot be displayed at the same real-world location for the different HoloLens clients (see chapter 5). Therefore the script is attached to all game objects that require their position to be shared (e.g. the laser pulse, walls, mirrors and targets). The script has a simple function: it has to ensure that the transform (position and rotation) of a game object is relative to its parent anchor across all client. See figure 7.13 for an overview of the network transform's code implementation.

The component uses the humble object pattern to make the class testable. The class implements the IAnchorRelativeMovementController and IAnchorRelativePositionSender interfaces. The IAnchorRelativeMovementController interface is used to move objects to a position relative from the parented anchor, the methods in this interface are used on objects that do not belong to the current client or server. The IAnchorRelativePositionSender interface contains methods that send the relative position to other networked instances of the object. These methods are used by clients or servers that own the object.

The AnchorRelativeNetworkTransform also stores and synchronizes the current parented anchor and the position and rotation relative to said anchor. The class allows the transform to be changed by whoever has authority over the object just like the regular unity network transform. It also supports some additional features besides the synchronisation of the relative transform for example, it supports the use of interpolation to smooth the movement and rotations of objects. This is done to combat the otherwise jittered movement that is due to network messages having packet delay and the send rate of packets being lower as the targeted frame rate [13].

The anchor relative network transform can be used in combination with the anchor parenter script. This script seeks the anchor closest to the game object the script is attached to and sets it as the new parent anchor in its AnchorRelativeNetworkTransform. The anchor parenter script only runs on the server or client that has authority over the object.

7.2.3. The Anchor Sharing and Management Framework

The anchor sharing and management system is a framework which manages the placement, importing, exporting and sharing of anchors. The system uses Unity's API, but adds a layer of abstraction for simplicity and reliability. The need for the framework arose when it was clear that the standard implementations were poorly documented or simply did not work on our network (see section 3.4). Moreover an implementation that simply handled exporting and importing anchors did not seem to be present.

The anchor sharing and management system has been written in a modular way, such that it can easily be pulled out of the project and made available for other projects in the form of a *Dynamic-link library* (DLL) or unity package. The current API allows exporting and importing anchors from and to either a file or a web server via http requests. However, these options can be easily expanded by simply extending the base

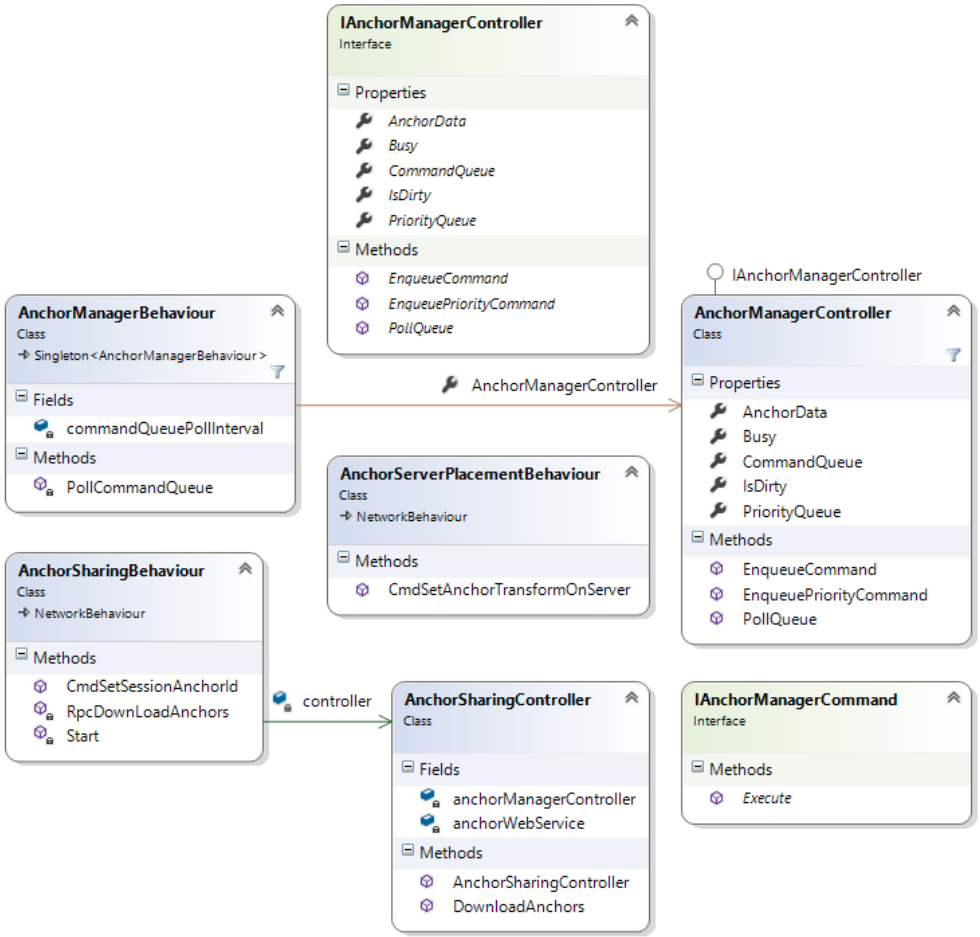


Figure 7.12: Overview of the localisation and anchor management framework's code implementation in UML.

import/export command classes. This makes the framework an attractive option for future HoloLens projects at the Delft University of Technology that want to use anchors, since, at the time of writing, we were the only project team there that has been able to successfully use the anchor capabilities of the HoloLens.

The anchor sharing and management framework uses the anchor manager as the core of the system, users can utilise its API and give it commands to execute. The anchor manager relies on a sharing manager, a file manager or a web service to execute its commands. These components will be discussed in the following sections.

7.2.4. The Anchor Manager

The anchor manager "manages" the anchors in the game. It gives a set of commands that can be used to interact with Unity's WorldAnchorTransferBatch [37], providing the functionality to add, remove, export and import anchors. The anchor manager consists of a controller and a behaviour. The controller follows the command pattern and has a queue consisting of anchor manager commands that is continuously polled by the behaviour. When the queue is polled, the controller will check if a command is currently executing, if not then it will execute the next command. A command calls the OnFinishedExecution callback when it has finished executing, signalling the anchor manager controller that the command has finished.

The behaviour is a script that has to be on a game object in the scene and is used to continuously poll the queue. Because this behaviour is a MonoBehaviour, this polling can be done in Unity's main thread. This means that the anchor manager commands are executed in Unity's main thread, which is necessary to be able to make Unity API calls. Since the anchor manager commands all work on Unity's WorldAnchorTransferBatch, they will all make Unity API calls. There are commands for the following actions:

- Adding an anchor

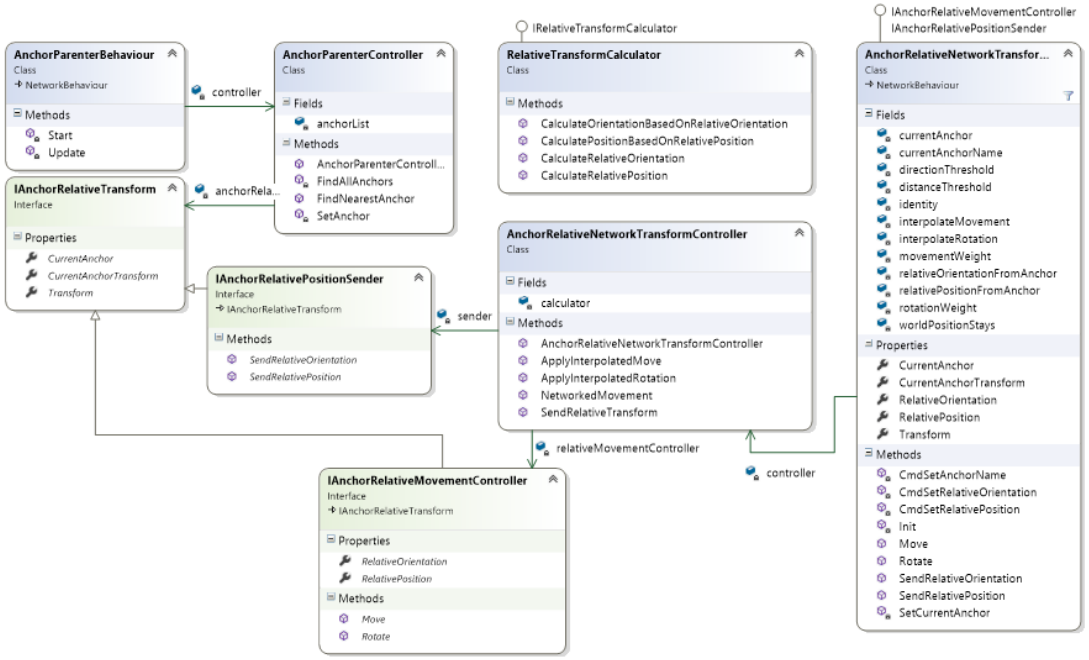


Figure 7.13: Overview of the network transform's code implementation in UML.

- Removing a specific anchor
- Removing all anchors
- Exporting anchors to a file
- Exporting anchors to a web server
- Importing anchors from a file
- Importing anchors from a web server
- Locking anchored game objects to their correct real-world location. Locking an anchored game object means moving their holograms to the real-world location of the anchor.

7.2.5. The Anchor Sharing Manager

The anchor sharing manager is a game object which has multiple scripts to ensure the sharing of anchors between the master client, the server and the regular clients is handled correctly.

The anchor sharing manager component, manages the sharing of anchors between the master client and the other clients. This script is split between a controller and a behaviour, following the humble object pattern. The behaviour will send a command from the master client to the game server containing the identifier of the anchor array on the web server (see chapter 5 for more information). When the game server receives this command, it will send an *Remote Procedure Call* (RPC) to all clients to download the anchors from the web server. The controller contains the actual logic to download and import the anchors from the web server, to make this logic testable.

The anchor server placement behaviour has a command that will be called by the master client, which will set the position and rotation of the anchors on the server to that of the master client. This is used in, for example, the creation of the playfield using anchors, where the position of the anchors decide where and how the playfield will be placed.

7.2.6. The File Manager

The file manager was created to handle the reading and writing of serialised anchors to local storage. It can read and write byte arrays to a file. The file manager uses the FilePathHelper, which helps to create file paths and contains a predefined file path to the application's persistent data path (see figure 7.14).

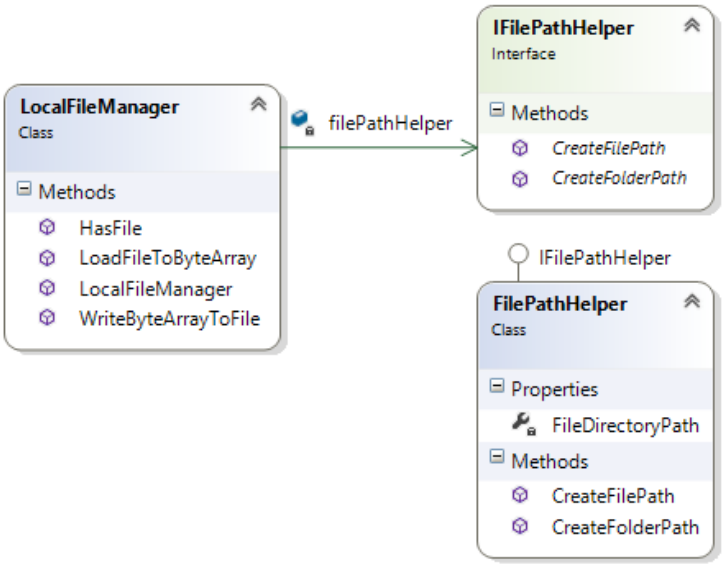


Figure 7.14: Overview of the file manager's code implementation in UML.

7.2.7. The Web Service

A web service has also been developed to allow clients to communicate with the web server (see section: 5.4). This web service is implemented as a singleton to ensure that only one instance runs at a time. This was done because multiple instances can interrupt each other's calls to the web server (due to Unity being single threaded).

The service implements the IAnchorWebService interface, which is used to upload and download anchor files to the web server, in particular it is used by the anchor manager in the export and import anchor commands (see section 7.2.4).

8

Analysis of Ethical Implications

To analyse the ethical implication of our project we choose to use a framework for *ethical technology assessment* (eTA) introduced by Palm and Hansson [30]. This method of eTA can be conducted by comparing the ethical implications of the new technology in nine crucial ethical aspects, namely:

1. Dissemination and use of information
2. Control, influence and power
3. Impact on social contact patterns
4. Privacy
5. Sustainability
6. Human Reproduction
7. Gender, minorities and justice
8. International relations
9. Impact on human values

In the following sections, we will briefly introduce these concepts and discuss the ethical implications of League of Lasers (if any) for each of ethical aspects.

8.1. Dissemination and Use of Information

Dissemination and use of information describes how information spread by this technology is being used. For example, the Internet is currently being used effectively for spreading pornography and facilitating crime. League of Lasers does not pose any ethical risk in this category. The players will all be in the same room and we do not aim to support *Wide Area Network* (WAN) capabilities, hence no information can be shared apart from the information that could already be shared implicitly through verbal communication by being in the same room.

8.2. Control, Influence and Power

Control, influence and power relates to whether the introduction of a technology will lead to a new distribution in control, influence or power, as happened with many new technologies in the past. Again no real ethical danger is present. The game will be released only for the HoloLens; limiting the audience to mixed reality enthusiasts, institutions or companies. Since the game is created for the sole purpose of entertainment and does not yield a competitive advantage for early HoloLens adopters in term of control, influence or power.

8.3. Impact on Social Contact Patterns

Impact on social contact patterns measures the impact a new technology might have on society. For instance, the smart-phone has fundamentally changed how people interact with each other. Whether this is good or bad generally depends on which generation you ask.

In case of League of Lasers, no substantial danger is present. Even if the game becomes a massive success since it is designed as a superhuman sport the amount of time people can play will be limited by their physical endurance. Secondly, people will still require other people to play with within the same room. Hence, it might actually stimulate social interaction.

8.4. Privacy

The balance between privacy is important for any new technology. League of Lasers does not send any personal data over the internet, however it is possible (though unlikely) that players sitting extraordinarily still during the scanning process can become part of the spatial mesh and can thus be used as anchors. Since anchors will be shared between HoloLenses it is possible that a player's spatial features will also be shared. However, it is very unlikely that these can be used for anything malicious, being merely a limited set of spatial points (with not enough accuracy to really distinguish or identify a person) used for anchoring that will become useless when the person moves.

8.5. Sustainability

All new technology may affect the economic, social, and ecological development of future generations. Making sure that a technology is sustainable and does not deplete the natural resources of generations is essential. In terms of sustainability, the product should be safe. First, our product can be reproduced by simply making a digital copy. Second, when running the game, the power consumption of the HoloLens will be so small that it is safe to assume no significant impact will be made upon the environment.

8.6. Human Reproduction

Certain new technology might have an effect on human reproduction either decreasing or increasing it (e.g. *In Vitro Fertilisation* (IVF)). No scenario could be envisioned in which League of Lasers can affect human reproduction.

8.7. Gender, Minorities and Justice

Technologies that discriminates against minorities or genders by using e.g. biased algorithms can be dangerous. No situation could be envisioned in which League of Lasers can be considered to discriminated minorities or genders.

8.8. International Relations

New technology can change the relations between nations. For example, certain countries might demand access to a biotechnology at prices affordable to their citizen. On the other hand, nations might demand that certain countries do not develop particular technologies at all (e.g. weapons technology). We do not see any way in which the game League of Lasers can impact international relations.

8.9. Impact on Human Values

There are technologies that can have a distinct impact on the way we see the world. To give an example: In recent years widespread adoption of the *Non-Invasive Prenatal Testing* (NIPT) [1] has become a famous example in moral debate. The latter test has made it possible to accurately determine whether the unborn child has a genetic disposition towards, for example, Down Syndrome (trisomy 21, or 47,+21). And has raised fears for increased abortion rates and with that impacts the human values.

League of Lasers is not expected to affect human values in any negative way. Games only have two ways to effectively impact human values namely their content and interaction with other players. The game does not contain any explicit content such as violence or sex. Explicit content in modern games is sometimes considered a risk for the human values, but is not condoned in any way in League of Lasers. In terms of player interaction, since all players are required to be in the same room there is effectively no real difference

between the game and normal human interaction, hence no impact on human values different from regular sports is expected.

8.10. Conclusion eTA

To analyse the ethical implication of League of Lasers the eTA framework was used. After analysing the nine critical aspects in the previous sections it was concluded that the game does not pose ethical dilemmas or complications in any of these aspects. Hence League of Lasers passes the eTA and can safely be enjoyed by people of all genders, races and ages.

9

Discussion and recommendations

This project was based on hardware and technology which is still in its development phase, to create an experience not yet seen before: a real-time multiplayer game using the Microsoft HoloLens with accurate tracking of player's position and orientation. This meant there were all sorts of issues and limitations along the way, but it also meant that it showcases what can be done in the future with this technology.

It is of utmost importance that all players see the game objects on the same real-life location as each other, as this further enhances the augmentation of the real-world and enables team play and coordination. With this project, a framework was developed to solve this problem: it makes it easy to manage and share anchors. With a little more work, this framework can be extracted from League of Lasers to be used in future HoloLens projects. In League of Lasers, this framework was used to position and move game objects relative to anchors.

While the HoloLens still has some issues and limitations, it still is a fantastic device. It shows the great potential AR has and gives a glimpse of what will become possible in the future. At first, we were sceptical about the "realism" of the holograms produced by the HoloLens, but it blew us away by how "real" the augmentations felt. A new HoloLens is rumoured to be revealed later this year, which tackles the main limitations: the limited field of view and the high cost of the device [47].

Future work and research in AR sports, like League of Lasers, should focus on audience participation. Unlike a regular sport, these virtual augmented sports would make it possible for the audience to change certain factors in the game. The audience could, for example, activate a power-up, which will alter the match in progress. In the case of League of Lasers, we could let the audience choose to increase the speed of the laser pulse or let them add additional laser pulses. Unfortunately, due to time constraints, the team did not have time to further research, design and implement these gameplay elements.

Since the virtual game world is only visible to the players, the audience will only see the players, fully immersed in the game, run and jump around. While this is a rather hilarious sight, the audience will have little information about what is going on in the game. Hence, a spectator view was developed, providing the audience with a top-down representation of the game, but far more can be done in this regard. The game could, for example, use a dynamic camera system which switches between top-down, sideline or even first person viewpoints to give the audience the best view of the action. Unlike a regular sport, the game is fully visible in the virtual world, which makes it possible to have full control over the camera's location. Another possibility is to add replay recordings after a team scored a point, which could show a goal from the best angle, perhaps even using a slow-motion effect.

10 Conclusion

The goal of this project was to create a “superhuman sport” as submission for the SHS Design Challenge. In more detail, the game should be a continuation of the game, League of Lasers, that was developed during the context project in 2017. The game was required to run on the Microsoft HoloLens, integrate tracking of position and orientation in a multiplayer setting, extend the fast pace of the original game and develop new gameplay elements. Moreover during the SHS Design Challenge the game will be judged on the the following aspects: human augmentation, fitness and skills, fun and engagement, innovation, audience, and inclusiveness. These judging criteria formed the basis for our product requirements.

To meet these requirements, a multiplayer game that resembles the old League of Lasers was created. Prior to this, a MoSCoW analysis (see appendix C) was done and multiple gameplay spikes were prototyped. The MoSCoW's must haves have been implemented, as well as most of the should haves (some were deemed not needed at the end). The could haves have not made it into the final product due to time restrictions.

10.1. Reflection on Product Requirements

The game developed is a faithful continuation of League of Lasers since the rule set is the same; the player controls a mirror and wants the laser pulse to hit targets. To do this the player must use the mirror to reflect the laser pulse. The game expands the gameplay by using a first person perspective instead of the top down view used in the original game. Moreover, an additional dimension has been added to the gameplay, the game is no longer a two dimensional experience: the laser pulse will now fly and bounce in three dimensions.

The game uses the HoloLens' capabilities to track the orientation and position of the players. A framework was developed to easily manage and share anchors, which was used to determine the position of game objects between clients. The speed of the game has also been increased, the game plays faster as the original, but not extremely fast. This choice was not made due to hardware considerations, but due to gameplay reasons. If the laser pulse moves too fast players simply cannot react in time. The game will track players sprinting through the game field reliably, therefore we consider this criteria met.

10.2. Reflection on SHS Evaluation Criteria

SHS design challenge judges the game based on the following criteria: human augmentation, fitness and skills, fun and engagement, innovation, audience, and finally inclusiveness.

Human augmentation is a criteria that has multiple interpretations. One could reason that the user's vision and hearing is augmented as the game projects holograms in space and uses spatial sound. However, augmenting the player's (physical) capabilities does not seem possible with merely an AR/VR headset. With League of Lasers we focused on the first interpretation, where we augmented players' vision and hearing to create a mix between gaming and physical sports.

The developed game requires skill and fitness: a player that can run faster or jump higher can also move faster and more nimbly through the playfield than others, giving him a distinct advantage. Moreover learning to position the mirror at the right angle and learning to anticipate the trajectory of the laser pulse is a skill (see appendix E).

Most of the players in play tests found the game to be fun and quickly got immersed into the game. Therefore, we conclude that the game is fun and engaging.

League of Lasers is one of the first mixed reality AR sports games for the HoloLens. Other multiplayer experiences are scarce but do exist on the platform however they do not necessarily focus on sports. It can therefore be regarded as innovative and a step forward for superhuman sports.

Audience participation was one of the more difficult requirements of the game due to time considerations. A spectator view was developed to give the audience an overview of what is happening in the game, but there were more “could have” features planned for it which were not implemented due to time constraints. One example of such a feature is that the audience could vote for events to happen in the game, such as a sped up laser, or the appearance of power-ups that influence the players that pick them up. These features are unfortunately not present in the current release of the game.

Finally, inclusiveness has been met as both children and adults can enjoy the game. Moreover the game is perfectly playable by people from different backgrounds, the only obstacles being handicaps such as poor sight or mobility, but this holds for sports in general. However, for most people the game will be perfectly playable, therefore we consider this requirement met.

10.3. Final Remarks

All in all, the project can be considered a success. League of Lasers solves all technical challenges and address all of the requirements and challenge criteria. Not all product requirements and challenge criteria were implemented to the same degree, but overall the product is satisfactory and works as intended.

A

Feedback SIG

A.1. Feedback First Upload

De code van het systeem scoort 4.4 sterren op ons onderhoudbaarheidsmodel, wat betekent dat de code bovengemiddeld onderhoudbaar is. De hoogste score is niet behaald door lagere scores voor Unit Size en Unit Complexity.

Op dit moment is de score dusdanig hoog dat we geen concrete aanbevelingen voor verdere verbetering hebben, hulde! Wel is het zaak om ervoor te zorgen dat jullie dit niveau tijdens het vervolg van het project vast weten te houden, en al helemaal op het moment dat de deadline in zicht komt.

De aanwezigheid van testcode is in ieder geval veelbelovend. De hoeveelheid testcode ziet er ook goed uit, hopelijk lukt het om naast toevoegen van nieuwe productiecode ook nieuwe tests te blijven schrijven.

Over het algemeen scoort de code dus bovengemiddeld, hopelijk lukt het om dit niveau te behouden tijdens de rest van de ontwikkelfase.

A.2. Feedback Second Upload

In de tweede upload zien we dat het project een stuk groter is geworden. De score voor onderhoudbaarheid is in vergelijking met de eerste upload gestegen. Die stijging is echter net genoeg om de score tot 5 sterren te laten stijgen, complimenten daarvoor. We zien dat jullie de bestaande code op een aantal punten hebben verbeterd, terwijl de nieuwe code ook van betere kwaliteit is dan bij de eerste upload.

Naast de toename in de hoeveelheid productiecode is het goed om te zien dat jullie ook nieuwe testcode hebben toegevoegd. De hoeveelheid tests ziet er dan ook nog steeds goed uit. Wel valt op dat jullie bij de nieuwe code iets minder streng zijn geweest wat betreft de verhouding tussen productiecode en testcode, probeer er in de toekomst op te letten dat je gedurende het hele project dezelfde kwaliteitsstandaarden weet aan te houden.

Uit deze observaties kunnen we concluderen dat de aanbevelingen uit de feedback op de eerste upload zijn meegenomen tijdens het ontwikkeltraject.

B

Original project description from BEPSys

Provided by TU Delft Sports Engineering Institute for the BEP Computer Science & Engineering 2017/2018 Q4 course.

B.1. Project Description

Super Human Sports (http://superhuman-sports.org/) are acquiring increasing attention from both massive public and media. Superhuman Sports have been defined as activities that rely on technology for human augmentation to enhance a human ability, involve physical fitness and skills, and are played for fun, competition or health reasons.

The type of technology deployed, the amount of movement and the human skills required can strongly vary. So far, there have been only a few attempts at developing experimental prototypes, often based on Augmented Reality (AR) technologies.

In this project, you will design and develop a game concept that somehow resembles the traditional 'paaltjes voetbal' game (mix of Football and Pong...) but put into a modern, fast-paced, AR Super Human team sport. We will build upon previous experience with a team game (called League of Lasers) developed within the Context Project 2017, that was not AR-based, but rather used top-down, camera-based tracking and a mobile for both input (using its compass) and output (see attached image).

The project will approach and solve a number of unique challenges, including:

1. Use of HoloLens in a (large) multi-player setting
2. Integration of tracking, orientation and precise interaction with limited bandwidth and field-of-view
3. Extend to the extreme the fast pace of the game (i.e. to the bare minimum imposed by hardware latency)
4. Development of new gameplay elements that prove to be fitting in this Super Human context

Each gameplay element of the game will have to be frequently and thoroughly tested in order to assure it is consistent with the overall experience and smoothly serving the game goals stated above.

In addition, an essential milestone of this project consists of presenting a live demo at the Super Human Sports Design Challenge (http://superhuman-sports.org/delft/) in the first week of July. As an interim milestone, another demo will be given at a VR/AR Event in the TU Teaching Lab, on 21 June 2018 (possibly, you will wish to use this for beta testing).

C

MOSCOW Analysis

In this section the must haves, should haves, could haves and won't haves features of the game will be discussed.

C.1. Must Haves

All of these features are equally important for the final product, the game cannot be played correctly without any of these. Therefore they have an equal priority.

C.1.1. Mirror

- Each player shall have a mirror.
- The mirror shall bounce the laser on impact, using the angle of incidence and the angle of reflection.
- The mirror shall stay in front of the player's view, so that he/she can move the mirror around by looking around the room.
- The mirror shall move along with the player.
- The mirror shall be transparent in order to not obstruct the player's vision.

C.1.2. Laser Pulse

- The game shall have a single laser.
- laser shall continually move with a constant speed.
- The laser shall reflect off the walls of the playfield.
- The laser shall reflect off mirrors.
- When the laser hits a target, it shall be moved to the center of the playing field and launched in a random direction.
- The laser shall not be able to leave the playfield.

C.1.3. Playfield

- The playfield shall be surrounded with walls, that will bounce the laser off on collision.

C.1.4. Score

- The game shall track the score of both teams.

C.1.5. Time

- The game shall keep track of the time played.
- The game shall have a time limit of 5 minutes.
- At the end of the time limit, the game must end and the team with the highest score wins.

C.1.6. Targets

- Both teams shall have a target.
- The targets shall be at opposing sides of the playfield.
- Targets will grant a point to the opposing team when hit by the laser.

C.1.7. Team

- The game shall have two teams
- The game shall support teams of at least one player each

C.1.8. Multiplayer

- The game shall support local network multiplayer
- The game shall sync the position of all game objects (mirrors, walls, targets etc.) in the game at the same real world position for each player.
- The game shall support networking for at least 2 players.

C.2. Should Haves

All of these features are quality of live improvements that will change the game from being a first playable to a truly functional product. The priority is mentioned after each feature, the priority scale goes from A to F where A means most important and F being least important. Crossed out features are features that were removed after feedback from either the supervisor or client.

C.2.1. Mirror

- Moving the mirror shall be responsive, the player should not notice any lag when moving his head. (A)
- ~~The mirror shall have a short cooldown when the laser hits it. During the cooldown it may not collide with the laser anymore, this is to exclude exploits in which very short bounces are used to walk the laser to a target. (F)~~

C.2.2. Score

- The game shall have a visual representation of the current score, on the spectator view. (B)

C.2.3. Time

- The game shall have a visual representation of how much time is left, on the spectator view. (B)

C.2.4. Audience

- The game server shall have a top-down visual representation of the game that can be projected or shown on a monitor. (D)

C.2.5. Multiplayer

- The game shall support 2 to 6 players. (E)

C.2.6. UI/UX

- The game shall have an indicator that will point the user to the laser's position. (C)
- Game flow (multiplayer):
  1. ~~The first player that joins will get asked how many players will play the game.~~ Replaced with: The server will ask how many players are playing, once it is started. (E)
  2. ~~When the game starts, it will automatically connect to the server.~~ Replaced with: Once a client starts a server selection screen is shown where the client can select a server. (D)
  3. Then each player will get asked which team they want to join. (E)
  4. When each player has selected their team, the match will start in 10 seconds. (E)
  5. When the match ends, the final score will be displayed. (C)
  6. ~~After 30 seconds, one player will get asked how many players there are. (F)~~
  7. ~~After this each player will get asked which team they want to join and the match will start.~~
  8. This will repeat itself.

C.3. Could Haves

These features will be developed if there is additional time. The priority is mentioned after each feature, the priority scale goes from A to F where A means most important and F being least important. Crossed out features are features that were removed after feedback from either the supervisor or client.

C.3.1. Mirror

- The mirrors shall give an auditory cue when they hit a laser. (A)
- The mirrors shall temporarily deactivate if they are hit often enough within a certain time frame. (B)

C.3.2. Laser pulse

- The laser pulse shall have a looping (3D) sound, so that players can easily determine its location in the game. (A)
- The laser pulse shall have a particle trail to give it a visual representation of the direction it is heading towards. (B)
- The laser pulse shall have the ability to change its speed through power-ups or influences from the audience. (D)
- The laser pulse shall increase in speed each time it hits a mirror. (D)

C.3.3. Playfield

- The playfield shall be based on a spatial map of the real-world playfield. (C)
- The laser should be able to bounce off objects scanned in the spatial map. (C)
- The game shall feature real-life obstacles that will have an effect in the game, these can be moved as well. (F)

C.3.4. Score

- ~~Players shall have an individual score, that will increment through rewarding the player for prestigious actions (e.g. show how many times the player scored a point, how many times he/she hit the laser, show who was the MVP). (F)~~

C.3.5. Targets

- The target shall give an auditory cue when hit by a laser. (A)
- The targets shall have the ability to move around under influence of the audience or through power-ups. (E)

C.3.6. Power-up

- The game shall support multiple lasers at the same time, this can be done as a player pickup or as an audience decision. (C)
- The game shall support power-ups that will change the movement speed of the laser(s). (D)
- Power-up where the laser is split into two, each of these lasers can only hit one predefined target. These lasers can be reflected by all players. (D)
- The game shall support moving targets as variant of the main game or as audience decision. (E)
- The game shall support changes to the size of the targets as power-ups or as audience decision. (E)

C.3.7. Audience

- The audience shall be able to make decisions about the game, these can be the activation of power ups or choosing which power-ups should be next as pickup. (C)
- The audience shall be able to stream an overview of the game on their smartphone. (C)
- The game shall show replays of goals, and alternate between player views in a representation of the game. (E)

C.3.8. UI/UX

- ~~The game shall have a minimap in game to give players a better overview of the game. (E)~~
- The game shall have an announcer that highlights important events in the game. (F)

C.4. Won't Haves

These features will not be developed.

- There shall not be any form of AI in the game.
- The game shall not support single-player.
- The game shall not support a chat system, players are close enough to each other to hear one another.

D

Plan of Action

Roles

Name	Role
Arjen	Meeting organizer Lead Game Designer Lead Network Architect
Shaad	Lead Communication Lead Art Director Minutes Secretary
Jop	Lead Programmer Producer
Niels	Lead CI, Testing and Code Quality Lead Interaction Design and UX Playtest Organization

Responsibilities

Meeting Organizer

- Makes agenda for meetings and presents them to others
- Prints agenda
- Leads the meetings

Producer

- Responsible for sprint planning for the week
- Ensures there's enough work to do, plans meeting if work runs out
- Ensures people add new features throughout the week to the pending-backlog column, and ensures no new features are added to the backlog randomly
- If tasks are too heavy, moves them to next week
- Organizes Waffle board - create according to scrum principles with the inclusion of a pending-backlog column

Lead Communication

- Setting up communication measures
- Moderating communication measures
- Ensure everybody communicates effectively (daily slackup/standup, no important matters in whatsapp unless contacting client)

Lead CI, Testing and Code Quality

- Ensures CI setup (Travis) is functional for our purposes
- Ensures code is tested properly with sufficient coverage
- Ensures all code is of high quality, free of smells
- Ensures code is well-commented

Lead Programmer

- Makes the decisions and choices regarding program architecture
- Ensures code is of high quality
- Documents important design decisions regarding the program

Lead Art director

- Responsible for everything related to decisions regarding sounds, music and visuals of the game
- Additionally responsible for providing the art whenever needed

Lead Game designer

- Explores various game concepts and shares these with the rest during brainstorm sessions
- Responsible for choosing the game's mechanics, direction, and concept

Lead interaction design and UX

- Process feedback from the playtest by
    - Analyzing feedback from users, the good, the bad and the ugly.
    - Creating proposals for improvement based on e.g. playtest feedback
- Ensure that players find the UX / Hololens experience intuitive
- Ensure that UI/Game is: meaningful, convenient, pleasurable, usable, reliable and functional.

Playtest Organization

- Playtesting should be done as early as possible.
- We aim to create a functional prototype by the end of May
- After the functional prototype has been created we aim to do (preferably) a weekly playtest.

Lead network architect

- Chooses a viable network architecture for the game
- Designs the networking infrastructure for the game or finds existing libraries
- Provides documentation for networking implementation or some other way to make important design implications clear to the rest of the team

Roadmap

Important Deadlines

- May 7 end of research phase / research report deadline
- May 11 hand in final paper SHS (13 may is the hard-deadline)
- May 31 target for first playtest
- International Festival of Technology 6-8 June
- **SIG**
    - 1 Juni deadline first upload to SIG
    - 22 Juni deadline second upload to SIG
- Juni 25 deadline report
- Juni 25 deadline executive summary
    - Milestones

Internal deadlines

- May 25 first-playable
- June 8 alpha version
- June 22 beta version

Collaboration

Sprint Planning and organization (SCRUM)

- Daily stand-up
    - Every day (9:30am at its latest) a short meeting, max 15 min. During these team members discuss their plans for the day and the issues they encountered before. If members a working remote the daily should be posted on discord.
    - The discord version will be posted BY EVERYONE if at least ONE person cannot attend the meeting. This is to ensure transparency.
- Weekly meeting first day
    - Previous sprint is evaluated and the new one is planned. Also the overall progress of the project is discussed. After this meeting issues are made and placed on the scrum board.
- Meetings with client/coach
    - We strive to meet these every week.
- “What we do when we have no work for the week anymore”
    - We’ll have a meeting and assign new issues that can be solved in the leftover time. Also we should consider how we should improve our planning.
- “Same as above for too much work”
    - When it is apparent that internal deadlines are not going to be met we should instead plan a short meeting and delay certain issues to a later sprint.
- How sprint planning looks (waffle etc)
    - How issues should look

        - Description of issue
        - Estimated Hours
        - Responsible person
        - Actual Hours
        - Tag (see below)
- PR/Issue behavior
    - Issues should have different tags to describe the issue. The different tags are listed below.
    - User stories always need to be referenced in the pull request, if the pr belongs to a feature.
    - Pull requests should try to be as small as possible, also pr's should (if possible) indicate which classes and methods should get extra attention during the review.
    - PRs should never be greater than 250 (ex test) lines of c# code
    - Should contain the actual amount of effort
- User stories
    - All features should belong to a user story.
- Sprint Management
    - Waffle
        - Will be used as a SCRUM board. The board will have the columns, Backlog, Pending Issues, To do, In Progress, Testing, Done. Pending issues is a special column, issues that are found during development will be placed here, in the weekly we will decide if these issues will make it into the backlog or not. The To do column will contain all issues that should be solved in the current sprint.
        - Descriptions of issue should be clean and only contain an actual description
            - Estimated effort will be assigned through the "size" attribute of Waffle issues
                - Actual effort will still be put in PRs when features are finished
            - Responsible person will be implicitly assumed through the division of roles at the top of this document
            - Everybody will put their issues and user stories on Waffle
        - All issues must have a short name and at least a short and more elaborate description
        - The "feature" label will exclusively be used for product features (see description of labels for their use cases)
        - User stories should be mentioned in their corresponding issues via #issuename
        - When branches are made for issues, name them "name-#relevant_issue", this way the issue is moved to "in progress"
            - For issues where no branch can be made (research etc), the move should be done manually

                - For PRs, put " closes #relevant_issue" in the title at the end, so that the issue is also moved to "In Review" and "Done" when being reviews resp. is done.
            - Interaction with issues on github should be minimized, mainly use Waffle (only PR discussion can happen on github)
        - SCRUM
            - On the first working day of a week, a review will be held of last week
                - How far everything is coming along
                - How many hours were spent on tasks
                - What tasks from the previous sprint need to be moved to the current sprint
                - What we will do to ensure timely sprint completion (or what went wrong)
            - Sprint planning and putting things in Waffle will happen directly after the first meeting
            - A roadmap will be made containing our milestones and completion goals
- Issue tags
    - The following tags are used:
        - User story
        - Feature
        - Test
        - Bug

Communication

- Discord
    - Everyone should have Discord on their laptop/desktop and the phone. It is assumed everyone will stay up-to-date.
- Whatsapp
    - Whatsapp will only be used to talk to the project coach
- Conflict resolution
    - If at least one person is in disagreement with a proposal
        - Responsibilities disagreeing person
            - Try to understand the proposal
            - Explain why you disagree clearly
            - If more than half of the group agrees, be flexible and go with it
        - Responsibilities proposal maker
            - Clearly explain and define the proposal in a way that the relevance is obvious and that it is flexible for adjustment in case
                - Make use of a pro/con list when discussing it

        - If the above does not work after a reasonable amount of time, change the proposal slightly in order to reach consensus
        - If the above does not work after a reasonable amount of time, drop the proposal
    - General for the group
        - If the majority agrees with a proposal, then whoever disagrees has to put up a flexible stance on the matter and go with the proposal
        - Meet face-to-face if discussion is too complex/hard to resolve online
    - Behaviour
        - Take the time to respond and stay calm: don't use an angry tone while speaking and reason sensibly about what you say
        - In particular, never resort to threats
        - Don't execute the proposal anyway unless the majority agrees: doing so anyway will damage the overall trust within the group

Code Quality

- CI
    - We will use Travis CI for continuous integration. The setup will be identical to that of our context project. It is important to remember that the combination of Unity and Travis is a bit iffy. Before a PR, always run the tests locally and check for any ReSharper issues in Visual Studio.
- Unit tests
    - We will try to test as much as possible using the Unity Test Runner, NUnit and the Humble Object Pattern (see old github repo for examples).
- Style checking & Code smells
    - We will use ReShaper and the ReSharper plugin StyleCop as tools to check for code smells. It is important to run this locally and not rely on Travis as there is no integration for StyleCop with Unity using Travis.
- Documentation
    - All code should be well-commented, will be checked by StyleCop.

Tooling

- Github
    - Link to old repo, we will fork this if we can reuse the codebase, otherwise we will create a new repository. We will not use the github SCRUM board, we use Waffle for this.
- Discord
    - Both laptop/desktop and mobile version.

- Visual Studio 2017 community (comes standard with Unity)
    - We will use the newest Visual Studio 2017 version available and keep up to date with the updates. We will use the Unity debugger plugin that is automatically installed with Unity.
- Resharper + Stylecop (see guide on the github wiki)
    - Plugin for Visual Studio that checks for code quality issues, style issues, code smells etc. Please check locally for any errors, before creating a PR, since Travis can't detect these issues.
- Unity
    - We will use the newest Unity version available and keep up with the Unity updates.
- Waffle
    - Waffle will be used as our SCRUM board. See this guide how to automatically use Waffle
- Google Drive
    - Link to drive
- Windows 10 pro/edu/enterprise
    - Only these versions support the HoloLens dev kits from Microsoft

Hardware

- HoloLens
- Laptops

E

Research Paper

Research Paper HoloLens Game Design

Shaad Alaka 4287851 Niels de Bruin 4375440 Arjen Miedema 4447247
Jop Vermeer 4462734

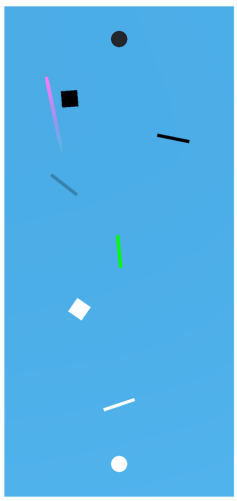
May 2018

1 Introduction

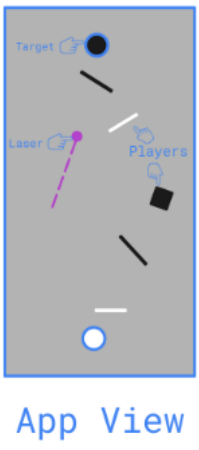
In July of 2018 the Superhuman Sports Design Challenge (SHS), an international symposium celebrating the next generation of inventors, is held. It is organized by the International Superhuman Sports Society [1]. The SHS aims to create sport-like experiences where the focus lies on augmenting human ability with technology and involve physical fitness and skills. These sports will be played for fun and physical health.

This project, proposed by Dr. ir. A. R. Bidarra and Dr. S. G. Lukosch, aims to make an HoloLens/AR version of League of Lasers, a game designed for the TU Delft context project in 2017.

League of Lasers' core mechanics revolves around reflecting a laser pulse to guide it towards a target [2]. Each team has its own target, so players of one team want to hit the opposing team's target while defending their own (see figure 1a and 1b). When a team's target is hit by the laser, the opposing team scores one point. The team that scored the most points at the end of the game wins; all in all, the general idea resembles football (soccer) and Pong.



(a) An overview of the in app playfield.



(b) An abstraction of the app view.

The new version of the game will use the mechanics of League of Lasers but should be regarded as a continuation of the original game. The game will be developed to be played using the Microsoft HoloLens [3]. Moreover the game needs to use a method to track the position and orientation of players in the game field, preferably without the use of sombreros and a overhead camera, which were needed in the original League of Lasers. The game needs to be fast paced, meaning that players should be able to move freely without being hindered by hardware limitations. Another requirement is to add new gameplay elements to league of lasers that are fitting to the Superhuman Sports context.

In addition to the requirements of the project the game will be judged at the Superhuman Sports Design Challenge. The competition committee will look at the following aspects of the game [1]:

- *Human augmentation:* How much does the superhuman sport augment human senses and capabilities?
- *Fitness and skills:* How much does the superhuman sport require or train physical fitness and skills?
- *Fun and engagement:* How engaging and fun is it for participants to play?
- *Innovation:* How innovative is the superhuman sport?
- *Audience:* How much fun is it for the audience to watch the superhuman sport?
- *Inclusiveness:* Can participants with different backgrounds practice the superhuman sport?

1.1 Research Question

Our research question is as follows:

> *How can the game design of League of Lasers be augmented for usage on the Microsoft HoloLens and meet the require-*

ments of the project and the Superhuman Sports Design Challenge?

This question can be split into the following subquestions:

- How can we integrate tracking and orientation handling?
- How does the HoloLens function in a Multiplayer setting (2 or more HoloLenses)?
- What are the hardware limitations of the HoloLens?
- How can the design of League of Lasers be improved through usage of the HoloLens?
  - What can we do with real life objects in augmented reality?
  - Are there any alternative control schemes that improve the gameplay?
- How can the design of League of Lasers be adapted to fit the requirements of the Superhuman Sports Design Challenge?
  - How to include physical fitness and skill in the design?
  - How to engage the audience?
  - How to make the game accessible (inclusive)?
  - How to make the game fun and engaging?

1.2 Structure

The paper is structured as followed: first the research methodology is discussed. Secondly the methods of tracking player are discussed. Then the multiplayer capabilities and requirements for the game are explored. Next, the hardware capabilities and limitations of the HoloLens are analyzed and finally the game design is discussed. After the subquestions have been answered a conclusion is given, which will provide recommendations for adaptions of the design of League of Lasers necessary in an AR setting.

2 Methodology

To answer the research question a mixture of literature research and experimentation was be used. For the literature overview several scientific papers and sources were used. Do note however, that due to the technical nature of the project and the relative young age of the technology, papers discussing the HoloLens hardware are limited. Therefore the choice was made to use technical manuals and documentation as sources as well.

The experiments were mostly relevant for the game's design, to get a clear overview of the capabilities of the Microsoft HoloLens and how well certain game mechanics of League of Lasers would translate to the HoloLens. For each experiment a small research question was created and a small report was written afterwards, these can be found in the design chapter of this paper. The code can be found in the following repositories: https://github.com/jopenmolles/BouncingLaser, https://github.com/nielsdebruin/mirror-hololens. The tools and hardware used for these small experiments are discussed in the following section.

2.1 Development Tools And Materials Used

For creating Windows mixed reality (HoloLens) applications, Windows 10 is required (Pro, Enterprise or Education versions are preferred due to them including Hyper-V, which is necessary for Microsoft's HoloLens emulator) [4]. Therefore all experiments were performed on Windows 10 Education. The team was using the most recent version of windows 10 Education, namely version: 1709.

Microsoft recommends the use of Unity as an easy way to build HoloLens applications. Unity is a game engine, which primarily uses C# as programming language [5]. As code editor, Visual Studio 2017 is recommended and is Unity's default editor. Microsoft has released a toolkit, which includes a collection of scripts and components which accelerates mixed reality development with Unity, called the MixedRealityToolkit-Unity [6] [7, p. 81–93]. This toolkit includes scripts for input, sharing (using multiple HoloLenses together), spatial mapping and a lot more. Because of this toolkit and the team's previous experience with Unity, the choice was made to develop all experiments in Unity using the MixedRealityToolkit. In more detail Unity 2017.4.2f2 was used and MixedRealityToolkit-Unity version: 2017.2.1.4 was used.

Unity has a holographic emulator, which can simulate a HoloLens in the unity editor [8] [7, p. 7–8]. One important note is that a game controller or gamepad is necessary to be able to control the player (e.g. Xbox 360 controller). The holographic emulator of Unity does not require Hyper-V, meaning it will run on operating systems other than Windows 10 Pro, Enterprise or Education. The holographic emulator is primarily used for quick testing and debugging, as it runs directly in the Unity editor without the need to build and deploy the game to an HoloLens.

Microsoft's own HoloLens emulator can also be used to simulate the HoloLens. It requires Hyper-V and thus needs Windows 10 Pro, Enterprise or Education [7, p. 7–8] [9]. The HoloLens emulator is more powerful than Unity's built-in holographic emulator, since it includes all the UI elements of the HoloLens and access to the Windows Store, Settings menu, Device Portal, etc. Unlike the Unity holographic simulator, you have to build and deploy the application to Microsoft's HoloLens emulator, so it is recommended to only use it near the end of the project for performing a final check, before installing the app on an actual device. The team did not use this emulator all that much during the experiments, as the usage of the built in Unity emulator was easier and faster to use and HoloLenses were usually present to deploy the builds on.

3 Tracking Possibilities

In this chapter the question: "How can we integrate tracking and orientation handling?" will be answered. To answer this question, we will consider several methods to determine a player's orientation, a player's position and the positions of other objects within the game in real-time, after that a conclusion giving the proposed methodology is given.

3.1 Determining A Player's Orientation

In this section, several methods to determine a player's orientation in the virtual world from their actual orientation in the real world will be discussed.

3.1.1 Gyroscope

The gyroscope is part of the HoloLens' Inertia Measurement Unit (IMU) [10]. Since the HoloLens is a head-mounted device, a HoloLens' user can move its head around to change the orientation of its representation in the virtual world. This functionality is already put to use in the basic environment of the HoloLens, where Windows' menus can be opened and the user always faces them [11]. Additionally, the user's head orientation also partially controls the central pointer's location that is used for interacting with user-interface elements, which is known as Gaze tracking [12].

Devices such as Android smartphones also contain gyroscopes [13]. The gyroscope's output can be accessed by running a native app on the smartphone through the Android SDK and sending this information to the game. This form of tracking is desirable if the game clients run on the smartphones, as they can directly access the gyroscope, such as with the game's former implementation [2]. However, the data the app provides could be supplementary to the main device of the game client.

Purely using data from a gyroscope to determine an orientation is prone to drift: the accumulation of position/orientation related error over time [14]. The HoloLens has its own solution for this phenomenon, which is also used for position tracking and will be elaborated in detail in section 3.2.1.

3.1.2 Camera

By utilizing image processing techniques on the input of an overhead camera, the players' orientation could also be determined. The players would be given a hat (e.g. a sombrero) that has an obvious front-facing direction, either by painting an arrow on it or through its elongated shape [15]. Then, the camera placed above the playing field facing downwards would capture the hats and derive their direction through image orientation analysis, thus finding the dominant orientation [16].

While this approach was not used for orientation tracking in the game's former implementation, it was used for position tracking. It was able to track players accurately, but it was sensitive for lighting conditions and had some mapping errors related to boundary behaviour (perspective effects, pin-cushion effect at the camera's edges of field of vision) that did not affect positional tracking adversely [2]. The expectation is that the same will hold for orientation tracking, and that boundary artifacts will be more visible for orientation tracking as it relies on working with shapes which may deform easily.

3.2 Determining A Player's Position

In this section, several methods to determine a player's position in the virtual world from their actual position in the real world will be discussed.

3.2.1 HoloLens

For positional tracking, the HoloLens makes use of its "spatial mesh" and its IMU [17]. The spatial mesh is a 3D model from the environment around the user that is generated by the HoloLens through four cameras as the user looks around with the HoloLens, creating a 3D virtual representation of the real world environment through a Simultaneous Localization And Mapping (SLAM) algorithm [17] [18, Section 1]. This mesh is continuously updated

and refined, but these updates occur with some latency and the resulting captures are noisy [19].

Through the IMU, changes in the HoloLens' position can be measured. This measurement is used in combination with a comparison of the scene captures of the cameras before and after the movement [17]. Through this comparison and the IMU data it can be determined whether the environment or the user has moved, and by how much. The IMU on its own can provide an approximate answer with little latency and noise by adding up differences, though this approach has relatively large errors through the phenomenon called drift, as explained in section 3.1.1. This is where the scene captures come in: by using the feature comparison between the two scene captures, which have relatively high noise and latency, but no drift, the player's position can be determined accurately in the order of millimeters, thus nullifying the error build-up by drift [17]. This mechanism is called inside-out tracking and is a very powerful feature of the HoloLens, allowing accurate position tracking without using additional tools (e.g. antennae).

While this approach works well for the position of the HoloLens, maintaining the positions of static holograms on the spatial coordinate system requires additional measures [20]. Microsoft recommends staying within 5 meters of the scanned area to prevent any drifting artifacts from becoming apparent. However, for applications that require a larger environment, spatial anchors may be used. These are important points in the scanned area that the HoloLens will remember. Holograms close to these will be free of drifting issues.

3.2.2 Camera

Similar to how orientation tracking would work using image processing, positional tracking is also possible using a camera as described in subsection 3.1.2. It is only necessary to detect and identify the users. The game's former implementation used large round hats with either a black or white rim, with a colored dot in the center [2]. The color was used for identifying the player, while the rim was used for identifying the team. These hats were detected using blob detection, through a multiplication of the value and saturation channels in the HSV color-space of the image, which would retrieve the centers of the colored dots [15], [21]. Since the shape does not matter much, it is only necessary to detect the specific combination of rim and dot color to determine some player's position on the playing field relative to the field of view of the camera [2]. Therefore any errors related to the shape of the hat are not significant. The perspective and camera boundary artifacts, can causing a relatively large error, but did not significantly impact the gameplay. however lighting condition did affect the implementation's ability to detect the colored dots properly.

3.2.3 WiCapture

It is also possible to use Wifi access points for position tracking that is resistant to occlusion and has a fairly long range [22]. The method for this described by Manikanta Kotaru and Sachin Katti called "WiCapture" makes use of the multipath problem in Wifi routing, making it work in favor of the problem at hand, by looking at phase-differences in the various propagation paths of the same packet. More Wifi access points or antennae can be added at stationary points to make the system more accurate. This will generate many false estimates from reflected paths, but also a few true estimates that resulted from the direct paths to the access points, which will generally lie close together. While existing Wifi localization systems can determine positional changes with an error of 40 cm, WiCapture is able to bring this down to a few millimeter.

One large benefit of this system is that it works with any commodity Wifi router, making use of packet meta-data called CSI which is present in any packet sent by any commodity Wifi router [23]. As mentioned above, the system is resistant to occlusion, has a large working range and is accurate down to a few millimeter. This all makes it suitable for VR/AR application.

For tracking multiple devices simultaneously though, it is not clear how the tracking accuracy would be affected, as the experiments that had demonstrated the high accuracy were only performed for tracking a single device. On top of that, the method was only tested with rather bulky WiFi routers attached to VR headgear: it is unclear whether the networking hardware in the HoloLens is powerful enough to substitute for such routers. These uncertainties, along with the uncertain implementation details make this a rather experimental endeavour that may be too time-consuming or impractical to set up properly.

3.3 Determining The Position Of Other Objects

Several image processing libraries exist that may aid in tracking real-world objects within the user's view. In this section two of these libraries will be discussed.

3.3.1 Vuforia

Vuforia is an object tracking library, it is able to recognize and track pre-determined visual markers robustly with very little latency [4], [24]. The robustness comes from the fact that not all features of some visual marker need to be visible in order to detect the object and when a marker is detected once, it can be tracked even when the marker becomes mostly occluded [24] [25, Section 6]. Because of this robustness and the overall speed at which Vuforia operates it can give users a great sense of immersion [26]. The Vuforia engine is natively integrated with Unity, which provides an easy development workflow [27].

Vuforia utilizes VuMarks for object detection. These VuMarks are similar to QR-codes in that they can encode information, but they allow more designer-freedom in terms of how they look, which is useful for creating logos with encoded URLs in them, or in this case, encoding object identifying data in an aesthetically pleasing way [28], [29].

3.3.2 Object Tracking Using OpenCV

OpenCV is another image processing library that may be utilized for object tracking, although it is a more general purpose image-processing library [30], whereas Vuforia focuses on object tracking specifically [24]. Object tracking in OpenCV can be approached in a multitude of ways, one example is the blobbing approach that was mentioned before in section 3.2.2. Another example is SIFT, which uses scale-invariant features on objects to identify them regardless of their size or screen-perpendicular orientation [31].

3.4 Conclusion

The team is fairly confident that it would be most convenient to initially start off with using the inside-out tracking of the HoloLens to determine a player's orientation and position. If issues with these methods arrive during development, a multi-modal solution could be implemented using one of the other discussed techniques. If the position of other objects would need to be determined in the game, it is proposed to use Vuforia as it is fully integrated within Unity.

4 Networking

This section will analyze how the HoloLens functions in a multiplayer setting. First some core concepts are given, which can be used to share a mixed reality scene between HoloLenses. After that, some multiplayer solutions compatible with the HoloLens are discussed.

4.1 Core Concepts HoloLens Sharing

The major challenge with creating a networked game with multiple HoloLenses is sharing the same virtualized world between them. The holograms should be located at the same position in both the game and the real world for all players. To enable this, HoloLenses make use of *spatial anchors* [20]. Spatial anchors are positions in the real world to which we want to attach a hologram. These holograms will then always be rendered at that particular position, and the rendering of these holograms will not be relative to the HoloLens itself, meaning that the hologram will not move if the HoloLens would lose track of its position or shift its internal coordinate system. Anchors should only be used for static objects that never move.

Anchors can be shared between HoloLenses, thus allowing different HoloLenses to see the same hologram at the same position even if their internal maps of the environment might differ [32]. This is useful for rendering static holograms at the same position for all players. Dynamic holograms such a the laser and mirrors could be parented to these anchors to synchronize their position, parenting these objects means that their position will be relative to the anchor until they move outside of the anchor's range (after that point they should be parented to another anchor). Static holograms such as walls and goals could be implemented as anchors [20]. A game like League of Lasers will depend on synchronized holograms for the laser, mirrors and other game objects as players should be playing in the same room and thus share the same representation of the virtual world. Therefore it is necessary to choose a networking solution that easily integrates into Unity and that supports the usage of anchors and anchor sharing.

4.2 Multiplayer Frameworks

In this section several networking solutions are briefly discussed, followed by a conclusion about which solution should be used. Frameworks that are incompatible with Unity and/or the HoloLens setup have been omitted.

4.2.1 Photon

The team is not familiar with this service but its features are promising. Photon uses a client to

server architecture [33]. It is cloud based, cross-platform and has features such as match-making. The Photon servers are room-based which is ideal for a game as league of lasers. Photon advertises its ease of use however these features do come at a cost, Photon is free for usage with less than 20 concurrent users, but everything after this point is paid. Another disadvantage is that Photon cannot be hosted on a user's own device. Moreover, Photon has no built-in convenience methods to share spatial anchors between players, these would have to be developed.

4.2.2 UNET

UNET is Unity's own networking solution, it features matchmaking and cloud-hosting, is room-based and fully integrated in Unity, and is cross-platform [34]. From experience, the team can tell that it is not the most intuitive networking solution, the system has a somewhat incomplete documentation and sometimes does not work as documented. However, it offers the option to deploy your own servers which are free of a concurrent player limit. UNET has support for sharing HoloLens anchors as well [35].

4.2.3 Microsoft Sharing Server

Microsoft's MixedRealityToolkit-Unity comes with a sharing package, which contains a console based server, called the Microsoft Sharing Server, that can be used to communicate world anchors between HoloLenses. Moreover, it can synchronize changes between players and send custom messages to other clients [36]. The main problem with this framework is the poor (almost non-existent) documentation and the fact that the server is not extendable, meaning modifications and extensions cannot be made to the source code of the server. It is the recommended networking solution provided by Microsoft and thus mentioned here.

4.2.4 Conclusion

Due to our previous experience with UNET and the full support of the HoloLens from within UNET, it seems like the most logical choice for this project. UNET has more and better documentation than the Microsoft sharing server and more flexibility when it comes to deployment than Photon. Local deployment is an important aspect for the team as it makes debugging easier and gives the ability to host on a local network which may reduce latency.

Another feature of UNET that is not present in the Microsoft sharing server, is that it allows the server to have a graphical interface, meaning that the game can be visualized in the same application instance as the server. This makes debugging locally more intuitive and also allows for displaying the game on a projector. Using the other frameworks, displaying the game on a projector would be more difficult, as a special client would have to be created.

All in all the team's previous experience and familiarity with UNET plus its full support of the HoloLens makes it the preferred choice for this project.

5 Hardware Limitations And Capabilities

In the following subsections, the capabilities and the limitations of the HoloLens hardware will be discussed. First the Holographic Optics are discussed. Second, The HoloLens' capabilities and limitations in terms of processing power are discussed. Third, the available sensors are described. Finally, methods for human interaction are discussed.

5.1 Capabilities And Limitations Of The Holographic Optics

The HoloLens boasts some impressive optic specifications on paper, especially when compared to the current state of the art of AR headsets [10]. However, as with all new technology, there will be some challenges during development. In this subsection, we will highlight some of the HoloLens' optics capabilities but also discuss its limitations.

The HoloLens has been equipped with the following optic capabilities:

- See-through holographic lenses
- 2 HD 16:9 light engines
- Automatic pupillary distance calibration, allowing the device to adjust the hologram display according to the users' interpupillary distance. Doing this makes sure holograms do not become unstable or appear at an incorrect distance [37].
- Holographic Resolution: 2.3M total light points
- Holographic Density: >2.5k radiants (light points per radian)

5.1.1 Limited Field View

One of the HoloLens' most prevalent visual limitations is the limited field of view. Currently, the

HoloLens supports a field of view of 30x17.5 degrees (horizontal x vertical) [38]. This limitation means that the image projected does not fully cover the human field of view which is 210x150 degrees [39]. It is possible that more experienced HoloLens users will get used to this limitation and will compensate for it by relying more on head movement instead of eye movement. However, for players just testing out the game (e.g. on a public event) this might severely impact their first impression of a game that makes extensive use of head movement.

5.1.2 Holograms Will Only Appear At A Certain Distance

The Holograms that the HoloLens projects appear at a distance of 80 centimetres [40], and moving closer to a Hologram will cause it to fade out. This might be a problem for League of Lasers since the laser beams will disappear before reaching the players. From testing with the HoloLens, we found that holograms fully disappear at a distance of approximately 50 centimetres. For optimal user experience, holograms should be placed between 2 and 5 meters from the HoloLens user [40] (see figure 1). At this point, the holographic displays will start to overlap which yields the best hologram quality. Moving holograms too far or close can cause strain on the user's eyes.

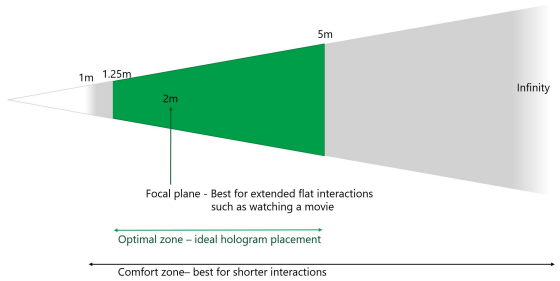


Figure 1: Optimal zone for hologram placement

5.2 Capabilities And Limitations In Terms Of Processing Power

The HoloLens makes use of an 32 bit Intel chip and has 2GB of RAM [10]. Given some of the more complex game demos such as RoboRaid [41] that we have seen so far we do not expect to be severely limited by processing power when implementing the basic League of Laser features (mirrors, lasers, collisions). However, we cannot estimate the impact of any potential new or modified gameplay features.

5.3 Available Sensors

Numerous sensors [10] are available to provide information about the user or the environment:

- 1 Inertial Measurement Unit
- 4 environment understanding cameras, used for tracking the head position and inside out tracking.
- 1 depth camera
- 1 2MP photo / HD video camera
- Mixed reality capture
- 4 microphones
- 1 ambient light sensor

Even though we anticipate using each sensor we do not need to interact with them directly. For instance, the Spatial Mapping API [42] will access the depth camera for us and abstract most of the low-level logic.

5.4 Available Methods For Human Interaction

There are multiple ways in which users can interact with the HoloLens. We will discuss these in this subsection.

5.4.1 Spatial Sound

The HoloLens supports spatial sound, which allows the user to experience sounds generated by a hologram, from the direction the hologram is located [43]. This in itself is not a new or revolutionary technology, however, it could be used to help the user generate additional spatial awareness [44]. Spatial sound is easy to implement using Unity [45] and will thus greatly increase the spatial awareness.

5.4.2 Gaze Tracking

Gaze tracking can be used to determine where a user is looking [12]. In our experience controlling an application through gaze tracking tends to be intuitive, but is not very accurate, since it is hard for a user to hold his head steady in place. Gaze tracking can be a powerful tool, some of its use cases are [12]:

- Intersect user's gaze with holograms to determine the focal point of user's attention.
- Target gestures e.g. control a menu, targeting system.

- Determine when a user is *not* looking in a certain direction. The latter could be used to alert the user about incoming enemies or other objects, through visual or audio cues when the player is not looking at them.

5.4.3 Gesture Input

Gestures are a supported input method, though these are limited. Essentially there are only two gestures: air tap (figure 2) and bloom (figure 3) [46]. Air tap can be used to interact with holograms after targeting it with Gaze. Air tap is the equivalent of a "click". Bloom is generally used to open en closed the HoloLens' main menu.

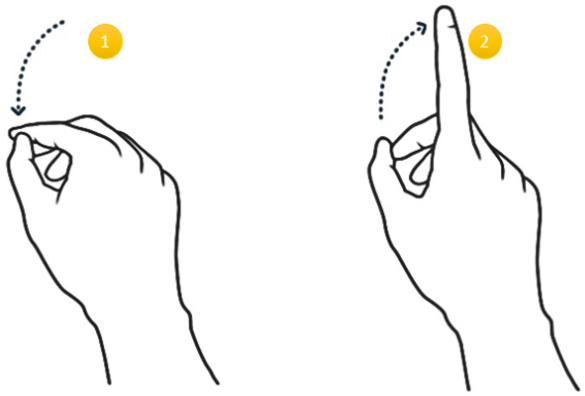


Figure 2: The Air Tap gesture

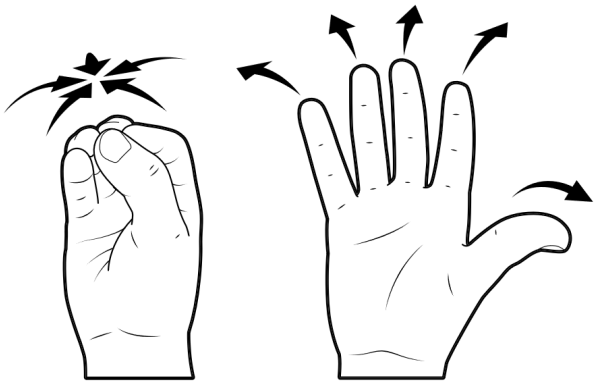


Figure 3: The Bloom gesture

5.4.4 Voice Support

The HoloLens comes with build in Cortana Support [47] and uses the same voice recognition technology as is used in the Windows Universal access apps. For our game voice recognition will probably be infeasible, since we will be using multiple HoloLenses in close proximity of one another. In such a setting, voice commands will start to interfere with one and another since user-specific voice recognition is not supported.

6 Game Design

In this section the research questions: "How to adapt the design of League of Lasers to fit the requirements of the Superhuman Sports Design Challenge?" and: "How to improve the design of League of Lasers by using the HoloLens?" are answered. To tackle these questions, the core mechanics of League of Lasers were explored in small programming experiments. The experiments and the results will first be discussed, then several extensions to the gameplay that take these results into account will be presented.

6.1 Experiments

Two small experiments (spikes) have been developed to test and research some of the core mechanics of the game. The first spike was about the spatial mapping of the HoloLens: can we bounce a laser around a spatial map of a room? The second spike was about the control scheme: how playable and intuitive would a mirror be that is projected in front of the player's view and follows the player around?

6.1.1 Spatial Mapping Experiment

The goal of this experiment was to research how spatial mapping works with the HoloLens. There are multiple ways to implement spatial mapping: using Unity's own spatial mapper or using the spatial mapper of the MixedRealityToolkit for Unity [48]. To use the spatial understanding capabilities, the MixedRealityToolkit's spatial mapper should be used. We wanted to know whether the HoloLens could map a room for use as a playfield and if we could let game objects interact with the mesh generated from the spatial.

Methodology
A small Unity project was developed where we used the MixedRealityToolkit's spatial mapping and understanding to map the room. When the game starts, the player has to move around to scan the room. The spatial mesh is given a physics material with the bounciness parameter set to 1, which causes colliding game objects to bounce off the mesh. After the scan has been completed, a ball is spawned with a starting velocity towards a random direction, with the velocity being maintained.

When it hits the spatial mesh, it will bounce off. This will cause the ball to bounce around in the room, since this spatial mesh contains the walls, the ceiling, the floor and other objects in the room.

Result
The mapping worked and spatial understanding resulted in a fully-closed spatial mesh of the room, where the ball could bounce around in. Any holes in this mesh are filled up and if necessary virtual walls are created, so that the spatial mesh is fully encapsulated and the ball cannot escape. This means that it will create a virtual wall in order to still have a fully-closed playfield, if the playfield is missing walls or a ceiling.

During the scanning process, it is recommended to not have moving objects in the playfield, otherwise they will remain in the spatial map, since it retains old meshes for a rather long time. This means that the game thinks there is an object that, in the real-world, is not there anymore. After the scanning is done, the room is mapped and the spatial mesh is not updated anymore.

We also noticed that the HoloLens had difficulties tracking large black surfaces (e.g. a projector screen, black table), they would not be mapped in the game. This finding corresponds with the spatial mapping documentation [19]. Here several factors that can create errors in the spatial map are listed, one of which being the fact that dark surfaces do not reflect enough light, so they can create holes in the mesh. This is something to keep in mind when choosing/creating a playfield.

This experiment let us know it is possible to map the real-world playfield to a mesh in the game. This means the laser can interact with real-world objects (e.g. bounce off walls and obstacles). We now also know that we can let the laser bounce in 3d space since it will just bounce off the ceiling and floor.

6.1.2 Mirror Experiment

The goal of this experiment was to determine the feasibility of moving a mirror through head movements using the HoloLens. Within League of Lasers the movement of mirrors is the way players interact with the laser. The simplest way to do this is to connect the mirror to the players head. We wanted know how well this mechanic translates to actual gameplay, so this experiment was designed. We will measure the feasibility through three criteria that will be elaborated upon below. Then the results, with respect to these criteria, will be discussed and it will be determined whether a mirror, controlled in the way described, is a feasible solution.

Methodology
A small Unity project was created, making use of the MixedRealityToolkit project and scene settings for the basic setup, and then adding a box with rigidBody and collision components to act as a mirror. This mirror was attached to the in-game camera and placed directly in front of it. It follows the camera around by changes its position and/or orientation, while preserving the same facing direction as the camera.

A laser object represented by a collision enabled sphere is spawned and launched in the direction of the mirror from a stationary point, with the HoloLens Air Tap gesture. The "laser" collides with the mirror and bounces in the new direction depending on the orientation of the mirror. Since the in-game camera is controlled using the position and orientation of the HoloLens, the mirror moves and rotates as the player moves and rotates. This allows the player to re-position himself in order to try and hit the laser with the mirror and make it travel in a specific direction towards a point of interest (such as a closet that is visible in the real world).

We used the following criteria to evaluate the results of our experiments.

- **Accuracy** consists of several components: First, It should be possible for a player to control the position/rotation of the mirror without any noticeable input lag. Second, it should not be difficult to target the laser into a certain direction using the mirror.
- **Intuitiveness** allows players to be able to move the mirror the way they intend in a relatively short playtime. Additionally, it also means that moving the mirror should not be a cumbersome act.
- **Physical fitness and skill** should provide an advantage, allowing players to actively engage with competing in the game.

Result
The implementation seems to be accurate: players can control the mirror without noticing input lag. It is trivial to reflect the laser towards a pre-defined target when its direction is not far from the incoming direction of the laser. When a target further away (in terms of angular change required for the laser) has to be reached, the player has to move a bit more, by walking and rotating a bit just before the laser hits the mirror. This clearly involves physical fitness and skill, hence satisfying that criteria

as well. The prototype was also intuitive: for users it felt as if they were holding a transparent sheet in front of them and coming up with the positioning to reflect the laser into the intended direction seemed like a task that was straightforward, but also challenging enough to remain interesting for larger angles. All in all, the results seem to be strictly positive, according to the specified criteria, meaning that this is a viable design direction for the game.

6.2 Adapting the Design of League of Lasers for the Superhuman Sports Challenge

In this section different aspects of the Superhuman Sport Design Challenge (see 1) and how League of Lasers could fit these aspects are discussed. Also additions to the design that fit within the Superhuman Sport Design challenge are given.

6.2.1 Human Augmentation

Human augmentation was something League of Lasers already did to some extent: humans were given a virtual representation that they were able to control through their movement and orientation in the physical world, while perceiving these changes on a screen. This representation can be conceived as an extension of their being into a virtual world, where they can be given abilities that are not limited by their physical form (such as deflecting a laser off their "body"). However, it can be argued that this is not a form of augmentation that is "convincing" enough, since especially in terms of augmenting the senses it is not all that different from the ordinary video game that is played on a PC.

The usage of the HoloLens will definitely bring the aspect of human augmentation to League of Lasers. In addition to being able to control extensions of themselves in a virtual world, users will be able to perceive a virtual world overlaid on top of the physical world. This is done by directly augmenting the players' vision and hearing with information that is necessary to play the game, instead of doing this through a detached screen and audio source. This will allow players to reason about the virtual world as they would do in the real world, which is a large gain from the former implementation.

6.2.2 Physical Fitness And Skill

League of Lasers was already a skill based game, however, due to the reliance on mobile phones for orientation and visualization of the playfield and the need to wear a sombrero for player tracking, the gameplay was slow paced. This was done as players needed more time to safely move around as they were watching their phones. Running could also cause players to lose their sombrero. The slow pace meant that, while players were moving and standing, their physical fitness was not tested by the game.

The usage of the HoloLens could potentially improve the speed of the gameplay, as players will no longer have to look down at their screens nor will they need a sombrero on their head. The expectation is that the increase in speed will stimulate players to move faster and challenge their fitness more, though research is needed to proof this. As the rotation of the player is still measured and the team mechanics are still present in the AR version of the game, the skill aspect should remain unchanged.

During the second experiment (section 6.1.2) it was found that controlling the mirror with head movement is more natural than expected. Additionally, the distance between the mirror and the player ensured that the laser hologram could not fade out as a result from (virtually) getting too close to the player.

While this control scheme seems satisfactory, peripherals could also be implemented, by for example equipping players with handheld devices that could act as mirrors in-game. The idea here is to use an arm mounted device which contains a gyroscope and accelerometer, such as a WiiMotion plus [49], to act as a virtual shield in the game. The advantage to this method would be that, if the hardware is accurate, players can reflect the laser more freely. The disadvantage is the need for more hardware and calibration steps in the game [49]. The usage of peripherals such as the above mentioned shield should add another skill aspect to the game, if needed.

6.2.3 Fun And Engagement

To define fun, the following description proposed by R. Koster [50, p. 90] is used. Koster defines fun as "the act of mastering a problem mentally". He later expands on this definition by stating that fun is the feedback the brain gives when we are starting to understand patterns for learning purposes [50, p. 96].

Predicting the laser's movement pattern is straightforward under normal circumstances, the laser always responds in the same manner under normal circumstances (power-up could change these), moreover from previous playtests we learned

that the rules and goals of the original version of League of Lasers as described in section 1 are easy to understand [2], players will quickly learn how to play. Therefore the game is extremely suitable for casual play due to its ease of entry.

While the game is easy to learn, mastering the game is difficult. The interaction between team mates and opponents ensures a great degree of variance and uncertainty in what's going to happen in an instance of the game, much like soccer for example.

As shown above the game is easy to learn, yet it is hard to master, due to the cooperative nature and the interaction between players. This means players will be stuck in the "act" of mastering the game and thus the game should be and stay fun.

6.2.4 Innovation

League of Lasers in itself was a quirky idea: there are not a lot of games like it, that combine team sports with AR. It is a good example of a case where mixed reality can really make a difference in innovation and uniqueness of a game concept. Mixed reality has some unique properties, the spatial mapping experiment (section: 6.1.1) showed the prospects of scanning environments and using them as the play field in the game. The usage of the ceiling and floor of the a allowed for some interesting interactions where the laser can bounce in all three dimensions. This behaviour could be extended by allowing players to build a unique playfield by placing objects in a room that will then be scanned into the geometry of the game. A challenge with this approach will be the synchronization of these objects across different HoloLenses.

Another interesting mechanic would be to place and move objects in the field that will then be tracked by the game. These objects will then change into mirrors or other objects in the game. The difference with the previous idea is that these objects can be moved after being scanned. Players would need to build paths for the laser to score a point, while the other team would try to destroy these. To make these kinds of interactions possible, alternative tracking methods will likely need to be used in the game. All in all, there is enough potential to further innovate within the formula of League of Lasers.

6.2.5 Audience

The previous version of League of Lasers already has an audience aspect, as the game overview can be shared on either a television or a projector and spectators can watch the match just like any other sport. With the addition of the HoloLens, the players' feeds could be streamed as well, using Mixed Reality Capture [51], [52]. This feature was tested and a latency of roughly 4 seconds was observed as well as performance issues with the HoloLens, which makes this unsuitable for live streaming footage, however, it could still be used for showing replays of game highlights, such as scoring a point.

Another option is to engage the audience by having them participate in the game. Giving audience members the option to help and/or hinder players during a game greatly increases their engagement and interaction [53]. This concept could also be implemented in League of Lasers, by giving spectators the option to select which power ups (if any) should be active or to alter parameters in the game to make the game easier or harder (laser speed, obstructed areas, etc.).

6.2.6 Inclusiveness

Accessibility and inclusiveness of games is a matter mostly discussed in relation to traditional video games. The SHS will look at how well people from different backgrounds can play the game. In the case of League of Lasers there are no discriminatory factors present that could exclude a person on the basis of cultural background, as the game resembles football, the most popular sport in the world [54], which is played all over the world. Moreover, playtests showed that adults and children both have no trouble playing the game [2]. Therefore it makes sense to look at accessibility to judge whether a person can play the game or not, instead of focusing on inclusiveness.

Guidelines exist to ensure a game has high mechanical accessibility [55]. Some of these guidelines relate to the input method of the game, in the case of a HoloLens game using the orientation and location determining abilities of the HoloLens, players would only need to be able to move their bodies and rotate their head to be able to play. This excludes people with poor mobility from playing.

One could look further and view accessibility as more as just mechanical accessibility but also as an extension of game design [56]. In this case accessibility is about how easy it is to understand and learn the game. As discussed previously (see section: 6.2.3) the game is easy to learn due to its simple mechanics. This was also supported by the play tests [2], which showed that children were able to understand the game rather quickly and had fun playing it. This leads to the conclusion that the game should be playable for most people, as it is easy to understand and it only excludes people with poor mobility.

6.3 Conclusion

As shown, the core mechanics of League of Lasers should translate well to the HoloLens. Moreover it was discussed that the game can implement the mentioned Superhuman Sports Design Challenge criteria and suggestions were made to further improve the game in order to better match the criteria. In the following section a final game design is given that combines the lessons learned from all individual research questions.

7 Conclusion

We propose to modify the original game design of League of Lasers in such a way, that the core concepts will stay the same, but the weaker elements, also with respect to the Superhuman Sport Design Competition criteria, will be enhanced. First, we want to change the way tracking is done. The original version of League of Lasers uses a camera with sombreros to track players' positions and their smartphone's gyroscope to track the players' orientation. The camera was sensitive to light changes and required sombreros. The gyroscope orientation tracking implementation was also prone to drift. Using the HoloLens, these issues should be resolved, as it can accurately track its own position and orientation.

The HoloLens can project a first person view of the playfield, where the players can see the laser, the targets, the mirrors, etc. This removes the need of smartphones and gives a first person perspective instead of a top-down perspective, which makes the gameplay a more intuitive experience. The audience would still see a top-down representation of the game on a projector.

Using the HoloLens' spatial mapping capabilities, we can map the real-world playfield to a virtual one and let the laser interact with it (e.g. bounce off real objects and walls). This virtual playfield can be visualized, letting the players easily see where the bounds of the playfield are and where the laser is. The spatial mapping also enables us to change the playfield by placing or moving obstacles, in the real world, even while the game is in progress.

The core concept of providing each player with a mirror, which can be used to guide the laser towards the opponent's target, also translates rather well to the HoloLens. During the experiments it was found that projecting the mirror in front of the player and moving it around with the player's head, was a natural and intuitive way to control the mirror, even more so than using the gyroscope found in the player's smartphone. An extension could be to use an arm mounted device with an accelerometer and gyroscope (e.g. a sort of shield with a smartphone mounted to it) as the mirror. Such a mechanic would allow players to bounce the laser without the necessity to adjust their heads orientation for operating the mirror.

Since the core gameplay of League of Lasers was fun and players liked it, we want to mostly extend and enhance the game's design, instead of making radical changes. Using the HoloLens, we can reduce or even remove the weak points of the original design, while further improving the good points, also taking the SHS criteria into consideration.

References

[1] Superhuman Sports Society. *Superhuman Sports Design Challenge*. URL: `http://superhuman-sports.org/delft/` (visited on 04/30/2018).

[2] Jop Vermeer, Shaad Alaka, Niels de Bruin, et al. "League of Lasers: A superhuman sport using Motion Tracking". In: *Superhuman Sports Design Challenge: First International Symposium on Amplifying Capabilities and Competing in Mixed Realities Proceedings*. Vol. 1. ACM. 2018.

[3] Microsoft. *Microsoft HoloLens*. URL: `https://www.microsoft.com/en-us/hololens` (visited on 04/30/2018).

[4] Matt Zeller, Chris White, Cosmos Darwin, et al. *Install the tools*. Mar. 2018. URL: `https://docs.microsoft.com/en-us/windows/mixed-reality/install-the-tools` (visited on 04/30/2018).

[5] Adam Tuliper. *Developing Your First Game with Unity and C#*. Aug. 2014. URL: `https://msdn.microsoft.com/en-us/magazine/dn759441.aspx` (visited on 05/07/2018).

[6] Microsoft. *Microsoft/MixedRealityToolkit-Unity*. URL: `https://github.com/Microsoft/MixedRealityToolkit-Unity` (visited on 04/30/2018).

[7] Sean Ong. "Beginning Windows Mixed Reality Programming". In: ().

[8] Unity Technologies. *Holographic Emulation*. URL: `https://docs.unity3d.com/Manual/windowsholographic-emulation.html` (visited on 04/30/2018).

[9] JonMLyons, Matt Zeller, and Brandon Bray. *Using the HoloLens emulator*. Mar. 2018. URL: https://docs.microsoft.com/en-us/windows/mixed-reality/using-the-hololens-emulator (visited on 04/30/2018).

[10] Matt Zeller and Brandon Bay. *HoloLens hardware details*. Mar. 2018. URL: https://docs.microsoft.com/en-us/windows/mixed-reality/hololens-hardware-details (visited on 05/01/2018).

[11] Addison Linville, Matt Zeller, rwinj, et al. *Billboarding and tag-along*. Mar. 2018. URL: https://docs.microsoft.com/en-us/windows/mixed-reality/billboarding-and-tag-along (visited on 05/04/2018).

[12] Alex Turner, Matt Zeller, rwinj, et al. *Gaze*. Mar. 2018. URL: https://docs.microsoft.com/en-us/windows/mixed-reality/gaze (visited on 05/01/2018).

[13] Android Developers. *Motion sensors*. Apr. 2018. URL: https://developer.android.com/guide/topics/sensors/sensors_motion (visited on 05/04/2018).

[14] N. O-larnnithipong and A. Barreto. "Gyroscope drift correction algorithm for inertial measurement unit used in hand motion tracking". In: *2016 IEEE SENSORS*. Oct. 2016, pp. 1–3. DOI: 10.1109/ICSENS.2016.7808525.

[15] H. Kong, H. C. Akakin, and S. E. Sarma. "A Generalized Laplacian of Gaussian Filter for Blob Detection and Its Applications". In: *IEEE Transactions on Cybernetics* 43.6 (Dec. 2013), pp. 1719–1733. ISSN: 2168-2267. DOI: 10.1109/TSMCB.2012.2228639.

[16] Sravan Bhagavatula and Nashlie Sephus. "Estimating the Dominant Orientation of an Object Using Image Segmentation and Principal Component Analysis". In: *Advances in Visual Computing*. Ed. by George Bebis, Richard Boyle, Bahram Parvin, et al. Cham: Springer International Publishing, 2015, pp. 243–252.

[17] Paul Aaron, Matt Zeller, and Matt Wojciakowski. *Inside-out tracking*. Dec. 2017. URL: https://docs.microsoft.com/en-us/windows/mixed-reality/enthusiast-guide/tracking-system (visited on 04/30/2018).

[18] Reid Vassallo, Adam Rankin, Elvis C. S. Chen, et al. *Hologram stability evaluation for Microsoft (R) HoloLens TM*. Mar. 2017.

[19] Brandon Bray, rwinj, Matt Zeller, et al. *Spatial Mapping and Design*. Mar. 2018. URL: https://docs.microsoft.com/en-us/windows/mixed-reality/spatial-mapping-design (visited on 05/04/2018).

[20] Alex Turner, Matt Zeller, Eliot Cowley, et al. *Spatial anchors*. Mar. 2018. URL: https://docs.microsoft.com/en-us/windows/mixed-reality/spatial-anchors (visited on 05/02/2018).

[21] S. Sural, Gang Qian, and S. Pramanik. "Segmentation and histogram generation using the HSV color space for image retrieval". In: *Proceedings. International Conference on Image Processing*. Vol. 2. 2002, II-589-II-592 vol.2. DOI: 10.1109/ICIP.2002.1040019.

[22] Manikanta Kotaru and Sachin Katti. "WiCapture: Motion Capture UsingWiFi". In: (2015). URL: http://stanford.edu/class/ee367/Winter2015/report_kotaru.pdf.

[23] Manikanta Kotaru and Sachin Katti. "Position Tracking for Virtual Reality Using Commodity WiFi". In: *CoRR* abs/1703.03468 (2017). arXiv: 1703.03468. URL: http://arxiv.org/abs/1703.03468.

[24] Vuforia. *Attach digital content to specific objects*. URL: https://www.vuforia.com/features.html (visited on 04/30/2018).

[25] Alexandro Simonetti Ibañez and Josep Paredes Figueras. "Vuforia v1. 5 SDK: Analysis and evaluation of capabilities". MA thesis. Universitat Politècnica de Catalunya, 2013.

[26] Fuguo Peng and Jing Zhai. "A mobile augmented reality system for exhibition hall based on Vuforia". In: *Image, Vision and Computing (ICIVC), 2017 2nd International Conference on*. IEEE. 2017, pp. 1049–1052.

[27] Vuforia. *Getting Started*. URL: https://library.vuforia.com/.

[28] International Organization for Standardization and International Electrotechnical Commission. *Information Technology – Automatic Identification and Data Capture Techniques – QR Code 2005 Bar Code Symbology Specification*. International standard ISO.: International Organization for Standardization. ISO IEC, 2006. URL: https://books.google.nl/books?id=Ga4PMQAACAAJ.

[29] Vuforia. *VuMark*. URL: https://library.vuforia.com/articles/Training/VuMark (visited on 05/03/2018).

[30] Kari Pulli, Anatoly Baksheev, Kirill Kornyakov, et al. "Real-time Computer Vision with OpenCV". In: *Commun. ACM* 55.6 (June 2012), pp. 61–69. ISSN: 0001-0782. DOI: 10.1145/2184319.2184337. URL: http://doi.acm.org/10.1145/2184319.2184337.

[31] T. Lindeberg. "Scale Invariant Feature Transform". In: *Scholarpedia* 7.5 (2012). revision #153939, p. 10491. DOI: 10.4249/scholarpedia.10491.

[32] Alex Turner, Matt Zeller, Eliot Cowley, et al. *Shared experiences in mixed reality*. Mar. 2018. URL: https://docs.microsoft.com/en-us/windows/mixed-reality/shared-experiences-in-mixed-reality (visited on 05/01/2018).

[33] Exit Games. *PUN*. URL: https://www.photonengine.com/en-US/PUN (visited on 05/01/2018).

[34] Unity. *Services - Multiplayer*. URL: https://unity3d.com/unity/features/multiplayer (visited on 05/01/2018).

[35] Unity Technologies. *Anchor Sharing*. Apr. 2018. URL: https://docs.unity3d.com/Manual/windowsholographic-anchorsharing.html (visited on 05/01/2018).

[36] Microsoft. *MixedRealityToolkit-Unity*. URL: https://github.com/Microsoft/MixedRealityToolkit-Unity/blob/master/Assets/HoloToolkit/Sharing/README.md (visited on 05/01/2018).

[37] BrandonBray and Matt Zeller. *Calibration*. URL: https://docs.microsoft.com/nl-nl/windows/mixed-reality/calibration (visited on 05/03/2018).

[38] *HoloLens and Field of View in Augmented Reality*. Aug. 2015. URL: http://doc-ok.org/?p=1274 (visited on 05/01/2018).

[39] Harry Moss Traquair. *An Introduction to Clinical Perimetry*. Henry Kimpton, 1938.

[40] Mark Hachman. *We found 7 critical HoloLens details that Microsoft hid inside its developer docs*. Mar. 2016. URL: https://www.pcworld.com/article/3039822/consumer-electronics/we-found-7-critical-hololens-details-that-microsoft-hid-inside-its-developer-docs.html (visited on 05/01/2018).

[41] Microsoft Corporation. *Get RoboRaid - Microsoft Store*. Mar. 2018. URL: https://www.microsoft.com/en-us/store/p/roboraid/9nblggh5fv3j (visited on 05/05/2018).

[42] Matt Zeller, Kelly Bakker, and Brandon Bray. *Voice input*. Mar. 2018. URL: https://docs.microsoft.com/en-us/windows/mixed-reality/spatial-mapping (visited on 05/01/2018).

[43] Hak0n, Matt Zeller, Kelly Bakker, et al. *Spatial sound*. Mar. 2018. URL: https://docs.microsoft.com/en-us/windows/mixed-reality/spatial-sound (visited on 05/01/2018).

[44] V. Sundareswaran, Kenneth Wang, Steven Chen, et al. "3D Audio Augmented Reality: Implementation and Experiments". In: *Proceedings of the 2Nd IEEE/ACM International Symposium on Mixed and Augmented Reality*. ISMAR '03. Washington, DC, USA: IEEE Computer Society, 2003, pp. 296–. ISBN: 0-7695-2006-5. URL: http://dl.acm.org/citation.cfm?id=946248.946841.

[45] alex turner, Matt Zeller, Eliot Cowley, et al. *Spatial sound in Unity*. Mar. 2018. URL: https://docs.microsoft.com/en-us/windows/mixed-reality/spatial-sound-in-unity (visited on 05/04/2018).

[46] Matt Zeller, rwinj, and Brandon Bray. *Gestures*. Mar. 2018. URL: https://docs.microsoft.com/en-us/windows/mixed-reality/gestures (visited on 05/01/2018).

[47] Hak0n, Matt Zeller, and Brandon Bray. *Voice input*. Mar. 2018. URL: https://docs.microsoft.com/en-us/windows/mixed-reality/voice-input (visited on 05/01/2018).

[48] Alex Turner, Matt Zeller, Eliot Cowley, et al. *Spatial mapping in Unity - Mixed Reality*. Mar. 2018. URL: https://docs.microsoft.com/en-us/windows/mixed-reality/spatial-mapping-in-unity#holotoolkit.spatialunderstanding (visited on 05/02/2018).

[49] Michael Erickson, Romulo Ochoa, and Cris Ochoa. "The Wiimote on the Playground". In: *The Physics Teacher* 51.5 (2013), pp. 272–275. DOI: 10.1119/1.4801352. eprint: https://doi.org/10.1119/1.4801352. URL: https://doi.org/10.1119/1.4801352.

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

[51] Jonathan Lyons, Matt Zeller, Kelly Baker, et al. *Using the Windows Device Portal*. Mar. 2018. URL: https://docs.microsoft.com/en-us/windows/mixed-reality/using-the-windows-device-portal#Mixed_Reality_Capture (visited on 05/01/2018).

[52] Wguyman, Matt Zeller, Eliot Cowley, et al. *Mixed reality capture*. Mar. 2018. URL: https://docs.microsoft.com/en-us/windows/mixed-reality/mixed-reality-capture (visited on 05/01/2018).

[53] Joseph Seering, Robert Kraut, and Laura Dabbish. "Shaping pro and anti-social behavior on twitch through moderation and example-setting". In: *Proceedings of the 2017 ACM Conference on Computer Supported Cooperative Work and Social Computing*. ACM. 2017, pp. 111–125.

[54] FIFA.com. *2014 FIFA World Cup^{TM} reached 3.2 billion viewers, one billion watched final*. Dec. 2015. URL: http://www.fifa.com/worldcup/news/y=2015/m=12/news=2014-fifa-world-cuptm-reached-3-2-billion-viewers-one-billion-watched--2745519.html (visited on 05/04/2018).

[55] Game accessibility guidelines. *Full list*. URL: http://gameaccessibilityguidelines.com/full-list/ (visited on 05/04/2018).

[56] Eitan Glinert. *Motivation for Accessibility in Games*. URL: https://www.gamasutra.com/view/feature/3538/designing_games_that_are_.php?print=1 (visited on 05/04/2018).

F

Playtest Questionnaire

League Of Lasers PlayTest Questionnaire

*Vereist

1. **What is your age?**

_____

2. **How would you rate the game (alpha)** *
   Markeer slechts één ovaal.

	1	2	3	4	5	6	7	8	9	10	
awful	◯	◯	◯	◯	◯	◯	◯	◯	◯	◯	awesome

3. **What do you think of the speed of the laser** *
   Markeer slechts één ovaal.

   - ◯ Should be faster
   - ◯ Just right
   - ◯ Should be slower
   - ◯ Anders: _____

4. **What do you think of the responsiveness of the mirror.** *
   Markeer slechts één ovaal.

   - Perfect
   - Lags a bit, not a problem for gameplay
   - Unresponsive I was not able to control the game

5. **Was something unclear?**

6. **Did you encounter any bugs?**

7. **Suggestions?**

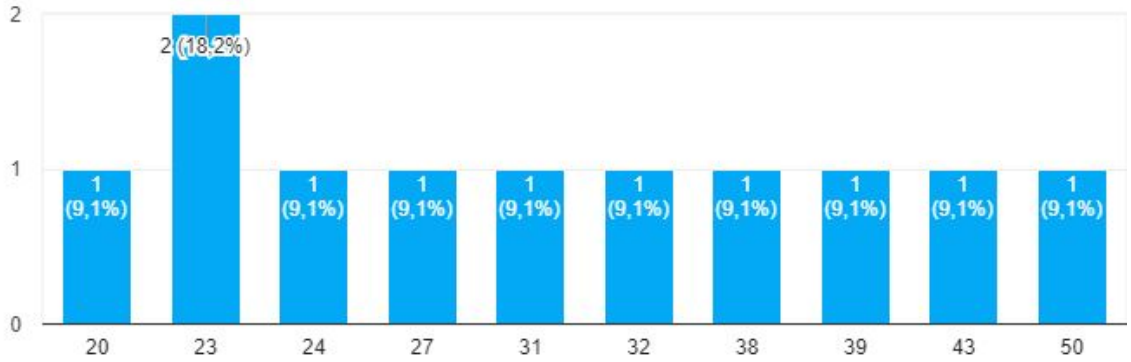
G

Playtest Results

Virtual Playground Questionnaire results

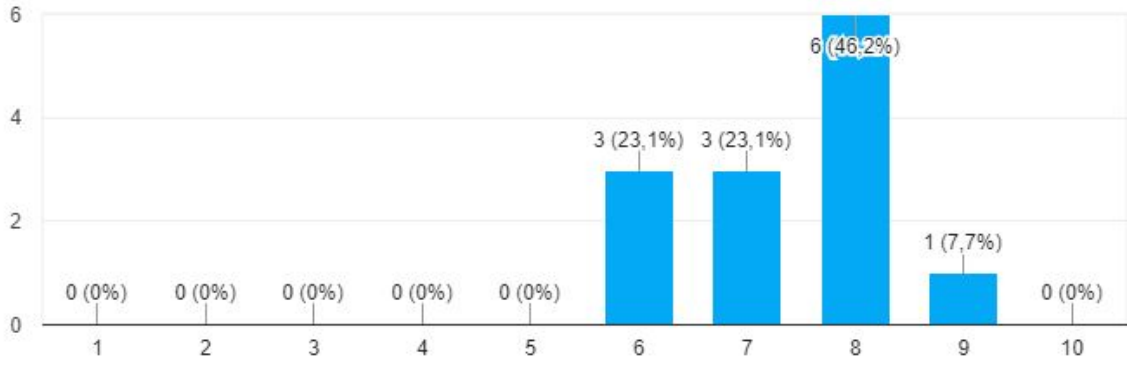
What is your age?

11 reacties



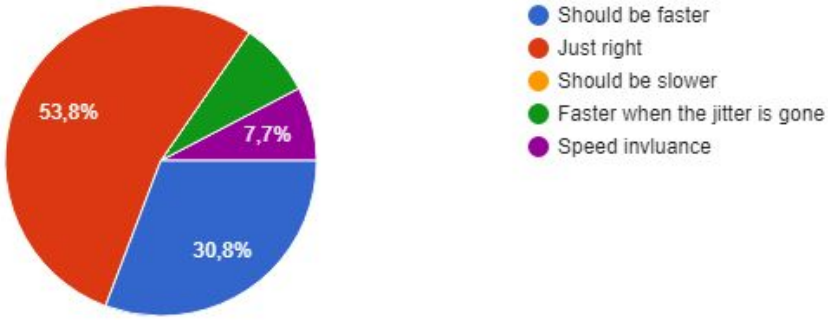
How would you rate the game (alpha)

13 reacties



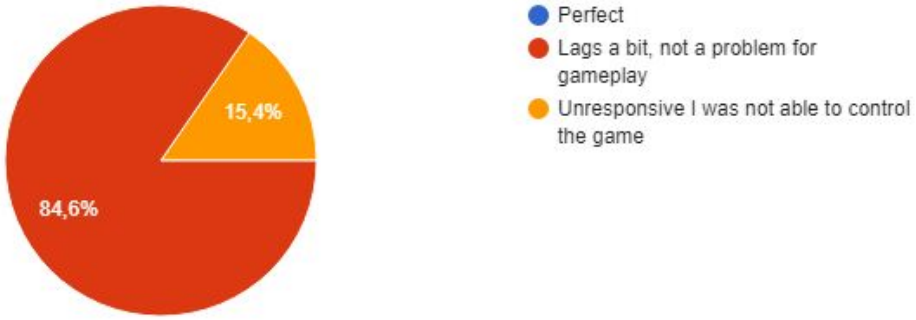
What do you think of the speed of the laser

13 reacties



What do you think of the responsiveness of the mirror.

13 reacties



Did you encounter any bugs?

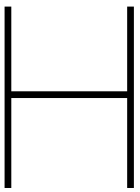
11 reacties

no (2)
Sometimes the mirror and the environment seem to tremble or vibrate.
problem with collision detection with the mirror
the ball went through the mirror several times
Ball seems to sometimes clip through the mirror
Yes, the ball went through the mirror multiple times.
Couldn't see the ball all the time, also not after "hitting it" so wouldn't know where it went.
nop
just lag
Ball sometimes bounces opposite direction

Suggestions?

8 reacties

when you aim give the bat -the mirror- a colored border
The goal was very high, difficult to protect the top of it.
make sure the base game functions robustly, reinforce scoring more and wait a bit before starting the next round, perhaps with a countdown
curious about how the focus is formed ahead of the device
adding a visual cue for the ball
sometimes you loose orientation and sight of the ball
maybe the mirror and the door can change color when touch the laser.
nope



Info Sheet

Superhuman Laser Game:
League of Lasers

Organization: TU Delft Sports Engineering Institute

Date Final Presentation: 04-07-2018

Challenge: Creating a Superhuman sport similar to 3D-Pong/Soccer on the HoloLens with multiplayer for the Superhuman Sports Design Competition.

Research: We learned about hardware limitation of the HoloLens and about the available approaches for multiplayer experiences on the HoloLens. From this, we developed an approach to consistently display Holograms at the correct location for multiple devices, using spatial anchors. We also created several prototypes for design elements of the game, such as bouncing off a ball hologram with a mirror hologram in front of the player, where we found high potential for the game we intended to make.

Process: We worked as a game studio, where each member had a specific role/responsibility. We used the SCRUM methodology for the game's development and pull based development. The game was created using Unity and Visual Studio 2017. We used a custom CI setup using Travis/Unity Cloud Build.

Product: The product consists of a HoloLens Unity app that includes the game client, a Unity game server and a web server responsible for transferring anchors between the clients. The product was tested using playtests on the HoloLens and unit tests via the humble object pattern and parameterized tests among other things. The framework for anchor management could easily be extracted for use for future HoloLens projects

Outlook: Technically the game functions well, and the codebase was written with maintainability in mind. Future recommendations would be to add game mechanics and systems that involves audience participation and further polish of the visuals, UI and audio. League of Lasers will be used for the Superhuman Sports Design Challenge.

The Team:

Member	***Roles and Interests***
Arjen Miedema	**Roles:** Meeting organizer, Lead Game Designer, Lead Network Architect **Contribution:** Anchor relative transform system, client side web service and spectator view **Interests:** Game design, game development, network programming.
Shaad Alaka	**Roles:** Lead Communication, Lead Art Director, Minutes Secretary **Contribution:** Game implementation, art, network interpolation **Interests:** Shader programming, game development, sound design and visual design
Jop Vermeer	**Roles:** Lead Programmer, Producer **Contribution:** Anchor management system and playfield creation **Interests:** Game development
Niels de Bruin	**Roles:** Lead CI & Code Quality, Lead Interaction Design/UX, Playtest Organization **Contribution:** Custom CI and Web server **Interests**: Artificial Intelligence, devops and web development

Contact:

Rafael Bidarra	***Coach***	Computer Graphics & Visualization Group	R.Bidarra@tudelft.nl
Stephan Lukosch	***Client***	TU Delft Sports Engineering Institute	s3_delft_design@superhumansports.org
Shaad Alaka	***Team***	Lead Communication	shaad1@live.nl

The final report for this project can be found at: http://repository.tudelft.nl

Glossary

API *Application Programme Interface.* 7, 17, 20, 30, 31

AR *Augmented Reality.* 1, 3, 4, 6, 9, 10, 19, 39, 41

CI *Continuous Integration.* 1, 17, 19, 20

DLL *Dynamic-link library.* 30

eTA *ethical Technology Assessment.* 1, 35, 37

Go *GoLang.* 13

HLSL *High-Level Shader Language.* 25

IDE *Integrated Development Environment.* 7

IP *Internet Protocol.* 14

IVF *In Vitro Fertilisation.* 36

NIPT *Non-Invasive Prenatal Testing.* 36

PR *Pull Request.* 20

REST *Representational State Transfer.* 14

RPC *Remote Procedure Call.* 32

SHS *Superhuman Sports.* v, 1, 3, 4, 7, 41

SIG *Software Improvement Group.* 1, 17, 21

TCP *Transmission Control Protocol.* 6

UI *User Interface.* 7

UML *Unified Modelling Language.* 23, 24, 25, 26, 27, 28, 29, 30, 32

UNET *Unity Networking.* 7, 11, 14

UWP *Universal Windows Platform.* 20

VR *Virtual Reality.* 1, 4, 19, 41

WAN *Wide Area Network.* 35

Bibliography

[1] Megan Allyse, Mollie A Minear, Elisa Berson, Shilpa Sridhar, Margaret Rote, Anthony Hung, and Subhashini Chandrasekharan. Non-invasive prenatal testing: A review of international implementation and challenges. *International journal of women's health*, 7:113–26, 01 2015.

[2] Astrogee. Unit tests cannot find my namespaces/classes, 2 2018. URL https://forum.unity.com/threads/unit-tests-cannot-find-my-namespaces-classes.515742/.

[3] Michel Beerens. Virtual playground, 2018. URL http://virtualplayground.tudelft.nl/.

[4] Jim Blascovich and Jeremy Bailenson. *Infinite Reality: Avatars, Eternal Life, New Worlds, and the Dawn of the Virtual Revolution.* William Morrow & Co, 2011. ISBN 0061809500, 9780061809507.

[5] jnm2 CharliePoole, ChrisMaddock. Testcase attribute, 2017. URL https://github.com/nunit/docs/wiki/TestCase-Attribute.

[6] Travis CI. Core concepts for beginners, 5 2018. URL https://docs.travis-ci.com/user/for-beginners/.

[7] Agile Business Consortium. Moscow prioritisation, 10 2014. URL https://www.agilebusiness.org/content/moscow-prioritisation.

[8] Discord. What features does discord have?, 2018. URL https://discordapp.com/features.

[9] Discord. Screen sharing & video calls, 2018. URL https://support.discordapp.com/hc/en-us/articles/115000982752-Screen-sharing-Video-Calls.

[10] Matt Ellis and Andy Reeves. Stylecop by jetbrains, 5 2018. URL https://resharper-plugins.jetbrains.com/packages/StyleCop.StyleCop/.

[11] ElvisAlistar. Sonarqube and unity (code quality), 12 2016. URL https://forum.unity.com/threads/sonarqube-and-unity-code-quality.444490/#post-2878642.

[12] Famatech. Advanced port scanner, 2018. URL https://www.advanced-port-scanner.com/.

[13] Glenn Fiedler. Snapshot interpolation, 2014. URL https://gafferongames.com/post/snapshot_interpolation/.

[14] GitHub. Github features: the right tools for the job, 2018. URL https://github.com/features.

[15] Google. Suggest edits in google docs, 2018. URL https://support.google.com/docs/answer/6033474?hl=en&co=GENIE.Platform=Desktop.

[16] Saisang Cai Paul Chapman Miku Jones Tglee Mike Blome Colin Robertson Gordon Hogenson, Genevieve Warren. Develop apps for the universal windows platform (uwp), 10 2017. URL https://docs.microsoft.com/en-us/visualstudio/cross-platform/develop-apps-for-the-universal-windows-platform-uwp.

[17] Software Improvement Group. Better code hub, 2018. URL https://bettercodehub.com/.

[18] JBD. Go cross compilation, 2016. URL https://rakyll.org/cross-compilation/.

[19] JetBrains. Resharper: Visual studio extension for .net developers, 2018. URL https://www.jetbrains.com/resharper/.

[20] JonMLyons, Matt Zeller, and Brandon Bray. Using the hololens emulator, 3 2018. URL https://docs.microsoft.com/en-us/windows/mixed-reality/using-the-hololens-emulator.

[21] Lucien King. *Game on: The History and Culture of Video Games*. Universe Publishing, Incorporated, 2002. ISBN 0789307782.

[22] Peter Koch. Hololens contest – (9 weeks to go) – networking, web server, shared experiences, 2017. URL http://talesfromtherift.com/hololens-contest-week-9/. Broken network discovery mentioned somewhere below.

[23] Yang Liu, Haiwei Dong, Longyu Zhang, and Abdulmotaleb El Saddik. Technical evaluation of hololens for multimedia: A first look. *IEEE Multimedia*, 2017.

[24] Microsoft. Udp-communication [solved], 2017. URL https://forums.hololens.com/discussion/7980/udp-communication-solved. Marked as solved while issue persists.

[25] Microsoft. Microsoft/mixedrealitytoolkit-unity, 2018. URL https://github.com/Microsoft/MixedRealityToolkit-Unity.

[26] Microsoft. Microsoft hololens - development, 3 2018. URL https://docs.microsoft.com/nl-nl/windows/mixed-reality/development-overview.

[27] N.A. Miedema. Example sharingwithunet still relevant?, 2018. URL https://github.com/Microsoft/MixedRealityToolkit-Unity/issues/2102.

[28] Glenford J. Myers and Corey Sandler. *The Art of Software Testing*. John Wiley & Sons, Inc., USA, 2004. ISBN 0471469122.

[29] Sean Ong. *Beginning Windows Mixed Reality Programming*. Apress, 2017.

[30] Elin Palm and Sven Ove Hansson. The case for ethical technology assessment (eta). *Technological Forecasting and Social Change*, 73:543–558, 06 2006.

[31] Lucio Tommaso De Paolis. Virtual and augmented reality application. University Lecture, 2015. URL http://avrlab.it/wp-content/uploads/2015/03/lez-1-introduction.pdf.

[32] Tomek Paszek. Unit testing part 2 – unit testing monobehaviours, 3 2014. URL https://blogs.unity3d.com/2014/06/03/unit-testing-part-2-unit-testing-monobehaviours/.

[33] Superhuman Sports Society. Superhuman sports design challenge, 2018. URL http://superhuman-sports.org/delft/.

[34] Tak. Cant activate personal license???, 8 2016. URL https://forum.unity.com/threads/5-4-0p1-fails-to-launch-due-to-license.425277/#post-2753911.

[35] CA Technologies. Developer-first project management for teams on github, 2018. URL https://waffle.io/.

[36] Unity Technologies. Holographic emulation, 2017. URL https://docs.unity3d.com/2017.4/Documentation/Manual/wmr_testing.html.

[37] Unity Technologies. Worldanchortransferbatch, 2017. URL https://docs.unity3d.com/2017.4/Documentation/ScriptReference/XR.WSA.Sharing.WorldAnchorTransferBatch.html.

[38] Unity Technologies. Worldanchor, 2017. URL https://docs.unity3d.com/2017.4/Documentation/ScriptReference/XR.WSA.WorldAnchor.html.

[39] Unity Technologies. Monobehaviour, 5 2018. URL https://docs.unity3d.com/ScriptReference/MonoBehaviour.html.

[40] Unity Technologies. Unity test runner, 5 2018. URL https://docs.unity3d.com/Manual/testing-editortestsrunner.html.

[41] The Wireshark Team. Wireshark, 2018. URL https://www.wireshark.org/.

[42] Alex Turner, Matt Zeller, Eliot Cowley, and Brandon Bray. Hologram stability - mixed reality, 2018. URL https://docs.microsoft.com/en-us/windows/mixed-reality/hologram-stability.

[43] Alex Turner, Matt Zeller, Eliot Cowley, and Brandon Bray. Spatial anchors, 3 2018. URL https://docs.microsoft.com/en-us/windows/mixed-reality/spatial-anchors.

[44] Unity. Unity cloud - build, 6 2018. URL https://unity3d.com/unity/features/cloud-build.

[45] Unity. Unity cloud - build, 6 2018. URL https://docs.unity3d.com/Manual/CommandLineArguments.html.

[46] Jop Vermeer, Shaad Alaka, Niels de Bruin, Nico Arjen Miedema, Nick Winnubst, Cyril Trap, and Rafael Bidarra. League of lasers: A superhuman sport using motion tracking. In *Superhuman Sports Design Challenge: First International Symposium on Amplifying Capabilities and Competing in Mixed Realities Proceedings*, volume 1. ACM, 2018.

[47] Tom Warren. Microsoft planning to unveil hololens 2 this year, Jun 2018. URL https://www.theverge.com/2018/6/13/17458168/microsoft-hololens-2-details-rumors.

[48] Matt Zeller, Chris White, Cosmos Darwin, and Brandon Bray. Install the tools, 3 2018. URL https://docs.microsoft.com/en-us/windows/mixed-reality/install-the-tools.