Bachelor's project

NTNU
Norwegian University of Science and Technology
Faculty of Information Technology and Electrical Engineering
Department of Computer Science

Mathias Stifjeld
Henrik Trehjørningen
Herman Tandberg Dybing

Creating a social space in VR

Bachelor's project in Bachelor in Programming[Games|Applications]
Supervisor: Christopher Frantz
May 2019

NTNU
Kunnskap for en bedre verden

Mathias Stifjeld
Henrik Trehjørningen
Herman Tandberg Dybing

Creating a social space in VR

Bachelor's project in Bachelor in Programming[Games|Applications]
Supervisor: Christopher Frantz
May 2019

Norwegian University of Science and Technology
Faculty of Information Technology and Electrical Engineering
Department of Computer Science

NTNU
Kunnskap for en bedre verden

Norwegian University of
Science and Technology

Creating a social space in VR

Author(s)

Mathias Stifjeld
Henrik Trehjørningen
Herman Tandberg Dybing

Bachelor in Programming [Games|Applications]
20 ECTS
Department of Computer Science
Norwegian University of Science and Technology,

20.05.2019

Supervisor Christopher Frantz

Sammendrag av Bacheloroppgaven

Tittel:	**Lage et sosialt rom i VR**
Dato:	20.05.2019
Deltakere:	Mathias Stifjeld Henrik Trehjørningen Herman Tandberg Dybing
Veiledere:	Christopher Frantz
Oppdragsgiver:	Progress Interactive
Kontaktperson:	Christopher Frantz, christopher.frantz@ntnu.no
Nøkkelord:	Bachelor, Virituel Virkelighet, VR, Programmering, Spill, Oculus, Vive, Dart, Dam, Rom størrelse, Skalerbarhet
Antall sider:	43
Antall vedlegg:	6
Tilgjengelighet:	Åpen

Sammendrag:

VR teknologi og hardware har blitt en moden teknologi, dette betyr at fokus har bevegd seg over til utvikling av VR applikasjoner. Dette er viktig i vårt tilfelle fordi i oppgaven utforsker vi utvikling av en VR applikasjon i Unity som allerede har bred støtte for VR, detaljrik dokumentasjon og et stort samfunn av utviklere. I denne oppgaven diskuterer vi utviklingsprosessen av et rammeverk for utvikling av sosiale VR rom. Vi har også utført diverse forsøk i forskjellige romstørrelser for å teste spillerenes reaksjon på dette. Det vi fant ut er at for å sosialisere med andre mensker så vil ikke et rom i 1:1 skala være stort nok.

Summary of Graduate Project

Title:	**Creating a social space in VR**
Date:	20.05.2019
Authors:	Mathias Stifjeld Henrik Trehjørningen Herman Tandberg Dybing
Supervisor:	Christopher Frantz
Employer:	Progress Interactive
Contact Person:	Christopher Frantz, christopher.frantz@ntnu.no
Keywords:	Thesis, Viritual Reality, VR, Programming, Oculus, Vive, Dart, Checkers, Room size, Scalability, Game, Bachelor
Pages:	43
Attachments:	6
Availability:	Open

Abstract:

VR technology and hardware is a mature technology, this means the focus has moved over to the development of VR applications. This is notable in our case because our thesis will explore the development of a VR application in Unity which already has wide support for VR, a detailed documentation and a large community of developers. In this thesis we discussed the development process of a framework for developing social VR spaces. We also conducted some experiments on different sized rooms to test the players reactions to different sized rooms, our findings show that in order to socialize with other people a 1:1 scale room is not sufficient.

Preface

We would like to thank Christopher Frantz for being supervisor. He provided answers to our questions, guidance and encouragement during the project.

Contents

List of Figures

Listings

1 Introduction

Virtual reality (VR) is not a new concept. It has been around for decades. Its first widespread commercial releases to the consumer market occurred during the 1990s, but it fell off the radar and out of the general consumers mind in during the 2000s. It has recently experienced a resurgence since 2010 when the first prototype for the Occulus Rift was designed, and later others such as the HTC Vive and PlayStation VR. In addition to these there has been numerous companies developing VR-related products. Looking at the "Gartner hype cycle for emerging technologies", since 2017[1] VR has entered into the mainstream far into the so-called "plateau of productivity" and was off the chart by 2018[2]. With VR becoming increasingly mainstream and widespread among consumers, the time is ripe for VR games.

1.1 Target audience

This thesis is mainly intended for people interested in developing VR games or just people with interest in VR in general. Knowledge about basic programming and Unity as well as familiarity with VR and its concepts is expected.

1.2 Project description

Our project was to make a room-scale environment, like a cottage, for socializing in VR. It should be possible to interact with parts of the environment and do activities such as playing familiar games like checkers and darts with your friends. The project aimed at being integrated with an existing networking solution developed by the Product Owner (company we were working for). We were also going to explore options to add communication functionalities, such as quick chat.

This meant we would need to make a system for scaling and potentially generating the room in which it all were supposed to take place. We would need a generalised framework for board games that would make it easy to implement multiple games (i.e. both checkers and chess), so as to not make unnecessary work for implementing more than one board game.

1.3 Background

The project was initiated by Progress Interactive, a game company in Hamar, with the intent to create a larger game. The finished game will be a Massively Multiplayer Online Role-Playing Game (MMORPG), set in a medieval setting, focusing on interacting with animals with different kinds of challenges and interacting with other players. In the context of the product owner's perspective, our project aims at serving as a social space/hub in which players can hang out with their friends and do crafting and other activities such as play board games between other activities that the full game has to offer.

1.4 Project organization

During the project, we used Scrum, an agile software development model. We chose Scrum due to us being familiar with it, the size of the team and because of the highly agile nature

of game development where features might be added, improved or removed on a regular basis. Because of our team size, however, we could not have anyone dedicated only to any administrative role as everyone would be required to be part of the development team to work on the project itself. The different Scrum roles within the group were therefore of secondary importance and available to be switched around freely if we felt someone was doing a poor job at that particular role. If one role had a particularly heavy workload compared to the other roles, we would switch the responsibility of that around so no one had to do all the work. We will go more closely into scrum and our use of it in section 4.2.

1.4.1 Roles and responsibilities

The initial assignment of roles and the responsibilities we paired with said roles looked like this:

- Internal
  - Scrum Master (Henrik)
    - Clearing obstacles
    - Establishing a good relationship between team and product owner as well as others outside the team
    - Facilitate meeting for the team
    - Planning, retrospective and daily stand-up
    - Convene meetings
    - Prepare meetings
  - Secretary (Mathias)
    - Take notes in meetings
  - Project leader (Herman)
    - Make everyone show up on time
    - Mediate internal conflicts
- External
  - Product owner (Progress Interactive, Richard Barlow)
  - Supervisor (Christopher Frantz)

1.4.2 Group rules

The group agreed on a set of rules for the group so that we would have something to fall back on should an issue occur within the group.

- **Repeated missed attendance** - After not showing up three times within a reasonable time frame, supervisor will be contacted.
- **Not meeting deadlines** - If deadlines are not met three times and it is not due to external factors, supervisor will be contacted.
- **Not performing their role in the group satisfactorily** - Discuss in the group if the role should be switched over to another member. If necessary, supervisor will be contacted.

Working Hours

We decided our base working hours to be Monday to Friday 09:00 - 17:00.

1.5 Report structure

The thesis consists of eight chapters. Here we will briefly go over the chapters following this one and look at what they are about.

2. **Requirements** - This section details requirements and limitations we had to follow.
3. **Technical Design** - In this chapter we will be looking at the design details for the architecture of the solution and the program flow for different components of the project.
4. **Development Process** - Here we will look at some of the major tools/technologies we used. We will also take a look at how we worked, our development model and how we used it, the individual sprints and our plan for the project vs what actually happened.
5. **Implementation** - This chapter will discuss how we created different things, problems we encountered during the implementation, how we solved them.
6. **Testing and User Feedback** - In this chapter we will take a look at the tests/experiments we did and what we got from them.
7. **Discussion** - Here we will be discussing the results from the experiment, what we did and did not do during the project, what could have been done different, changes made during the project.
8. **Conclusion** - In this chapter we will be evaluating the project as a whole and concluding any findings we did. We will also discuss how the work could be continued in the future.

2 Requirements

In this chapter we will discuss the requirements laid down by the employer, both in terms of content of the application and the technical requirements.

2.1 Task description

Virtual reality offers socialization at a distance for physically isolated and anxious or anti-social individuals. The goal of the project is for students to create social VR environment, where players can play familiar games in a room-scale environment, similar to a cottage. The students will work for an external Game Development company based in Hamar, although occasional assistance will be provided on the Gjøvik campus by company staff. Art assets, core systems and feedback on programming implementation will be provided by the company. The programming assignment needs to submitted, checked for appropriateness and then feedback provided to the students at regular intervals.

2.2 Technical Solution

- Develop a bespoke Raknet-based multiplayer solution for Unity3d using a master-client[1] architecture and the external company's remote log in server.
- Investigate voice-based communication options
- C# client and server-side VR implementation of familiar mini games in Unity3d, such as:
    - Checkers
    - Darts
    - Rock, paper, scissors
    - Bespoke card game (external company provides graphics and design)
- Virtual Reality implementation using Steam VR for compatibility with both Oculus VR equipment and the HTC Vive.
- Students submit code to the external company URL in a private git repository. The Computing Department in Gjøvik will also have access to the URL to monitor progress.

[1] The product owner has since clarified that they meant server-client architecture.

2.3 Standards

A requirement given to us by the product owner was their standards, both in terms of code standards as well as more general project standards. These standards were in place to smooth the development and to make sure that there were no misunderstandings due to the structure of the code or the use of external tools. The coding standards specifies things like naming convention, immutability and inheritance, while the project standards specified for example, version of unity, usage of version control(git) and branch philosophy (feature branching) etc.

2.3.1 Code Standards

1. Classes, public/protected member functions and property names use PascalCase
2. Method parameters, private member functions and local method variables are camelCase
3. Fields are prefixed with an underscore and then follow camelCase e.g. fieldName. i.e. No C++ m prefix e,g, m_fieldName.
4. Fields are private and have a public property getter e.g. FieldName get return _field
5. No hungarian e.g. m_iHaveMadeMistake
6. Interfaces prefixed with the letter I e.g. ISaveData
7. Static classes like Singletons should have a suffix e.g. GameManager.
8. Static member variables do not necessarily need to be prefixed with s_, although that is acceptable.
9. Structs should be immutable data containers - Structs should not contain object references, otherwise they should be made into classes.
10. All code should be within the Progress.Rec namespace. (Rec standards for recreation.)
11. Datatype structs and classes should have the suffix _t .
12. Unity serialisable objects should have a script at their root e.g. the provided SceneRoot or PrefabRoot. This can be used for initialisation, asset management and optimization.
13. All Unity classes must inherit from the ManagedBehaviour class instead of the standard MonoBehaviour.
14. Use PropertyDrawers and attributes rather than custom editors based upon types, since this tends to raise the likelihood of serialisation bugs.

2.3.2 Project Standards

1. Use Unity 2018.2.141.
2. Pull, commit, and push regularly.
3. Ensure you merge new changes from master into your feature branch occasionally, so your merges with the mainline are easier and are less likely to cause problems.
4. Ensure you use meaningful commit messages.
5. Avoid making partial or mixed commits. All file changes and dependencies for a coding task should be included within a single commit. Phases of completion are allowed.
6. Merge and test changes before pushing to the master branch.
7. Do not push changes to metafiles where the guids have changed. This will break references within scenes.
8. Force serialise assets to text. Do not use binary serialisation for scene files, which will make tracking changes and merging more challenging.
9. Use the provided google spreadsheet for reserving scene files so that they are locked.
10. Make changes to prefabs separately from instances placed in the scenes. You can drag in a new instance of the prefab from the Project Window, save your changes and then

remove it.

Following the requirements, we will next take a look at the technical design for the project, and after that we will go on to talk about the process of development.

3 Technical Design

In this chapter of the thesis, we are going to be taking a look at the technical design for the project. We will start by taking a look at the solution architecture. We will then move on to talking about the program flow for various components of the project.

3.1 Solution Architecture

The original project, along with networking and further features, was intended to operate using a standard server-client solution. That means every player has/is a client that sends state information to a company-owned server. The server then does any necessary computation (if there is any that is required from the server end) or synchronisation and shares state with other clients in the same instance/area – as shown in Figure 1. In the end we did not manage to integrate networking functionality due to additional evaluation. We will explore this in chapter 7.

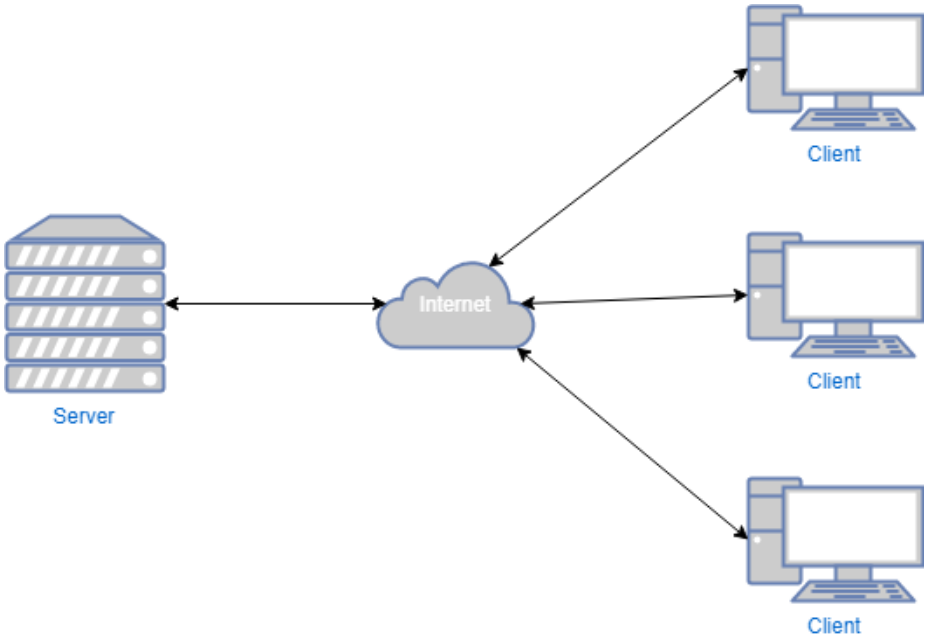


Figure 1: Typical Server client architecture

3.2 Program flow

Rather than developing a single coherent system, we were tasked with developing a framework that could be used to compose a customised system. Thus there is no overarching program flow, but it can be divided into multiple program flows.

3.2.1 Minigames

Because the project aimed to provide an environment where players could play the virtual equivalent of familiar physical games, we consulted with the product owner and together decided that the mechanics of these games would not be programmatically controlled. What we mean by this is that for e.g. the game of checkers, the program does not stop players from making moves prohibited by the rules of checkers; instead, the players enforce the rules themselves (or agree to break them, as the case may be). Any conflicts arising between players would be resolved using a designated method, described in 3.2.1.

Because the program does not enforce the rules of these games, there would be no game logic or program flow for the minigames themselves, merely for setting up the environment for playing said game when the players request it. That would mean, setting up a board with pieces to move for board games like chess and checkers, or a table with cards for the different card games. This is shown in Figure 2. There could potentially be a menu to select different pieces or game boards to instantiate (for chess, checkers etc.) or cards for different types of card games (Boards, pieces and card faces could be provided by the company or even possibly the community / the players themselves if the company wishes to take that approach).

One potentially beneficial side effect of this approach is that players could make up new games themselves, using existing boards/pieces/cards. It would also let the players represent regional or cultural variants of existing games, or popularly accepted modifications to the rules. As long as all players agree what the rules are, the only limiting factors are imagination and available pieces.

The exception to this ruleless approach is the conflict resolution system, described in 3.2.1. It would abide by the "rules of the game", whatever the game may be. Letting this system be customisable by players could introduce unfairness. This system should just start a menu of some kind or just a countdown before the result is shown. It still follows the same flow for setting it up as both players participating in it need to want to "play" it and be ready to do so.

Conflict resolution / decision making

One of the minigames were also going to double as a tool for making decisions in cases where people could not agree with each other on their own, such as splitting loot or the likes. There are many different ways this can be done, such as a coin flip or a dice roll. The product owner wanted it to be rock paper scissors for this game, despite being aware that it is not truly random and is affected by skill[3].

Rock paper scissors is a game that is mostly standardised and has minimal possibility for customizing by players, and as mentioned, it could introduce unfairness if left to the players how this minigame works[3]. As such, it would be implemented with the standard rules of the game. Rock paper scissors would still follow the same basic flow of setting up the environment / starting the game as the other minigames like detailed in the above, though using UI elements instead of physical pieces. The players would then pick what they wanted, and the game would count down and reveal the picks of the two players as well as who won at the same time. Figure 3 shows how a regular game of rock paper scissors is played.

Ultimately, we did not implement rock paper scissors for decision making (or any other decision making mechanism for that matter), nor the framework for card games. We will discuss why we did not implement these features in section 7.7.

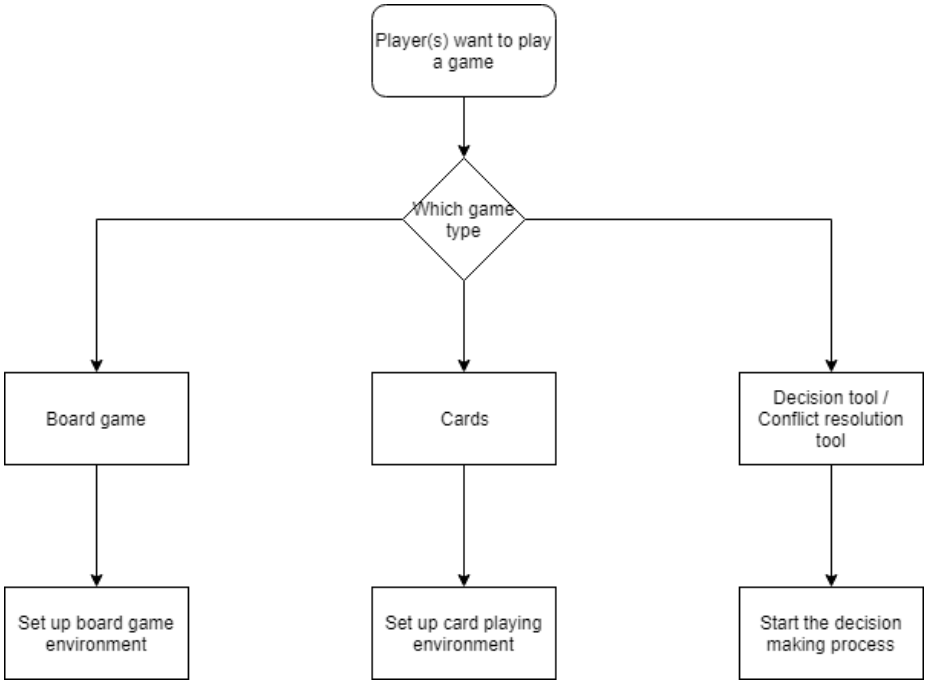


Figure 2: Program flow for activities in the room.

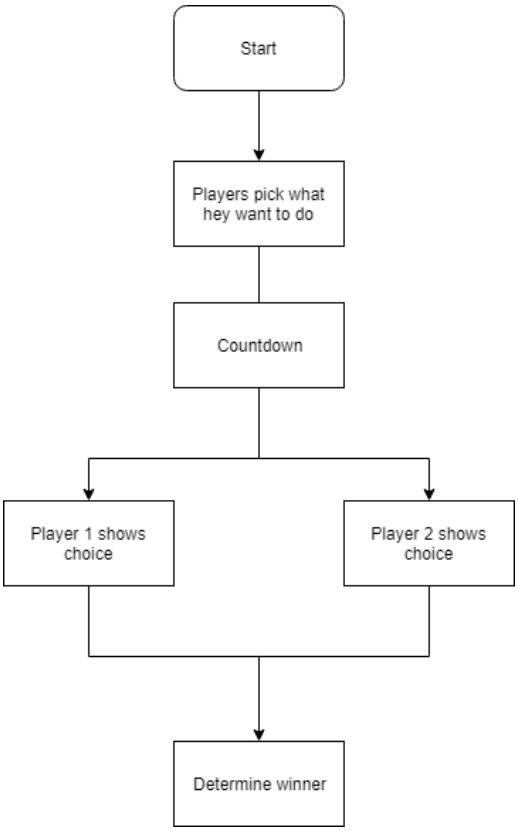


Figure 3: Program flow for the Rock paper scissors minigame.

3.2.2 Scaling the room

Figure 4 shows a possibility for the program flow of a system for scaling the room like we were tasked with at first. This might not be good as it would keep on checking for players in the room which, would most often be unnecessary work for the system, and it could then potentially randomly rescale the room when no one is expecting it.

Another solution could be to have it just stop after scaling the room and then scale again if a new player enters the room or if someone exits it can check again in case it was the player with the smallest play area that left. It could also just run again the next time the room loads, but that would mean no one with a smaller play area could join mid session, so it would perhaps be best to run the check when someone enters or exits the room.

In order to explore the different options systematically, we decided (in agreement with the product owner) to divert from the original plan and devised an experiment to evaluate whether it was at all feasible to make this project with 1:1 room scaling and movement and for multiple people. We describe the details of the experiment in section 6.1

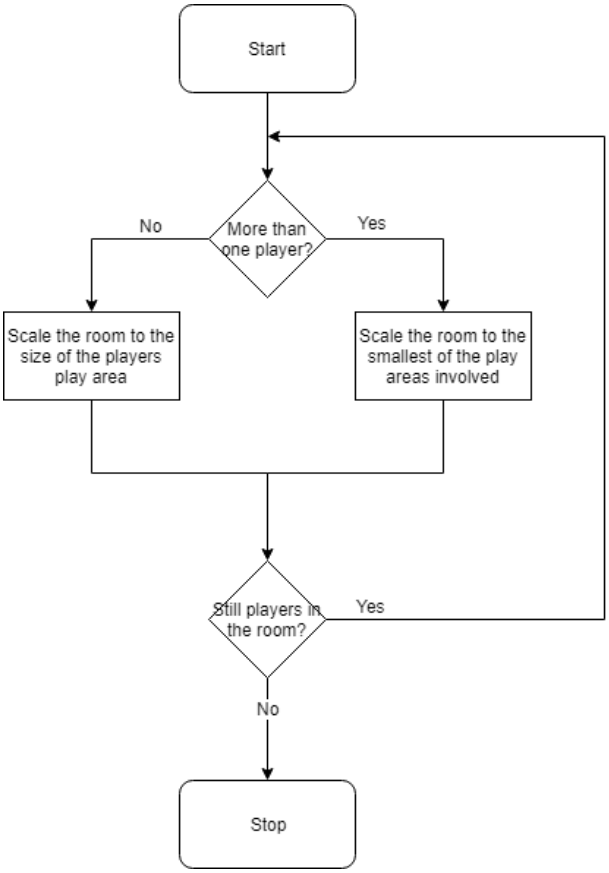


Figure 4: Program flow for scaling the room.

Before discussing implementation-related aspects in greater detail, we discuss the development process we followed and describe changes in greater detail.

4 Development Process

This chapter is about the process by which we worked on our thesis. We will talk a little about the major tools we used during development. We will also talk about our choice of model for software development and how we used it. Following that we will look at our workflow and go into our planned schedule and what actually happened. We will then go into some detail about what we did during the different sprints.

4.1 Development tools

This section will talk about some of the major tools and programs we used while developing the project.

4.1.1 Unity

We used Unity version 2018.3.4f1[4] for reasons explained in section 7.4. We spent the first few days / week becoming familiar with Unity since not all of us had much experience working with it prior to the project. With Unity, we mainly wrote code in C# using Microsoft Visual Studio.

4.1.2 Version control

For version control we mainly used git bash with Github when working on our own stuff in a temporary repo. When dealing with the company's repo for the project, we used Bitbucket and *Sourcetree*, a git desktop client and GUI.

4.2 Scrum

As touched upon in the introduction, we chose to use Scrum for this project. The reason being that we were familiar with it, a small team and that developing games is a highly agile process. In this section we will take a closer look at scrum and how we used it during the project.

When using Scrum, you work in iterations of a set length called sprints. After some discussion, we decided on a sprint length of two weeks as the time frame for the project was not that long and we wanted to get more than just a couple sprints in before the end.

Scrum traditionally has a meeting at the beginning of each sprint called *sprint planning meeting*, and one at the end of each sprint divided into two, the *sprint review* and *sprint retrospective*. The sprint planning meeting is used to discuss the upcoming sprint and decide what should be done during the coming sprint. For the end of sprint meeting, the sprint review is to show what we did or did not do during the sprint, and the sprint retrospective is for the team to reflect on the past sprint, what went well and what can be improved upon.

We decided to merge these meetings into one meeting at the start of each sprint in order to minimize the number of meetings and make it easier for the company we were working for to coordinate with us.

Scrum also has short daily meetings called *Daily scrum* or *Daily stand-up*. This is a meeting in which people say what they did yesterday and what they are planning to do today. They also say if they see any problems with what they are going to do or that day. In the end, we did not make much use of these because we were always in the same room working together and therefore we felt that we had little need for these meetings.

You can read more about the philosophies of scrum and the various aspect of it at *The Scrum Guide*[5].

4.3 Workflow

The workflow for the project went like this. We would start the sprints with the sprint planning meeting on Monday with the company. There we would decide what we would work on for the coming sprint and add it to our sprint backlog. Once that was done, we would start working on the things that needed to be done. We worked on separate things, or in groups / pair programming if needed. If we meet problems we could not solve by ourselves, we would ask the others for help and collectively take a look at it. At the end of the sprints (and the beginning of the next sprint), we would present what we had done at the sprint review with the company.

4.4 Schedule

Here we will take look at what our plan for the project was. We will also take a look at what actually happened, the contents of the each of the sprints, and why the things that happened did happen.

4.4.1 Planned schedule

We made an initial Gantt chart for the project. It was very high level and really only detailed the sprints and when we were going to stop developing and start writing the thesis. This was because we did not know what the company wanted when and had no idea for what the different sprints would contain, see Figure 5.

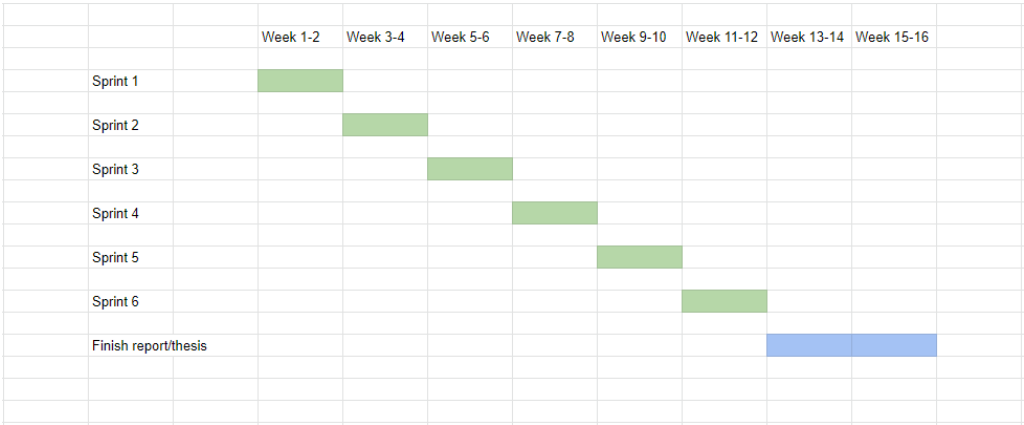


Figure 5: Original Gantt chart.

4.4.2 Actual schedule

Reality did not turn out quite as we had planned however. We ended up doing an unforeseen experiment that really threw the original Gantt chart into shambles. The experiment also took more time than we first had anticipated, which led to time constraints on other things.

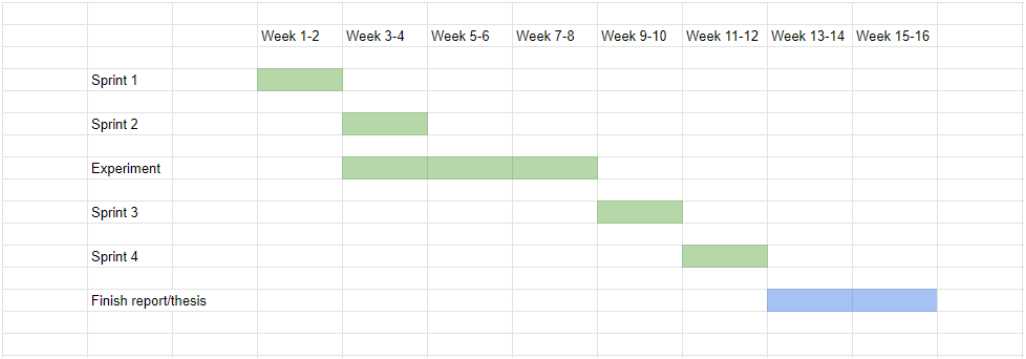


Figure 6: Actual Gantt chart.

From the Gantt chart in Figure 6 we can see the impact the experiment had on the planned project timeline. It effectively took up two and a half sprints, which in turn left us with three and a half sprints to actually work on the project. This in turn led to time constraints and having to prioritize some features over others. The timeline for the experiment itself was something like Figure 7. We will not go into detail about the experiment here, you can read more about that in section 6: testing.

Figure 7: A Gantt chart showing the experiment broken down to different components.

Looking at Figure 7 we see that most of the time for the experiment was used in preparing all that we needed. It took more time than we had imagined as we were not aware of all that needed to be done in order to actually conduct an experiment like formalities such as information and consent forms and needing to check if we had to apply for approval from *Norsk senter for forskningsdata*(NSD). That further shrunk our remaining time compared to what we had recalculated for. When everything was ready, conduction the experiment went fairly smoothly with no major roadblocks. Analyzing the data did not take all that much time either and we quickly turned it into usable graphs and charts.

4.4.3 Sprint 1

The first sprint we started by getting to know Unity and how it works. We then started on the task of generating a room using the size of the available play area. We quickly noticed the fact that the smallest room were not very big and not all of the planned activities would

fit into an area that small, let alone more than one person. We talked to the Product Owner about it at the sprint retrospective, and he gave some suggestions that could offset this while still keeping in line with the original idea of 1:1 scale and movement. These were:

- Windows.
- Counters at the edge of play area (you get extended view so that it does not feel as cramped, but not more actual play space).
- Able to see/throw over the "boundary" between rooms but not walk.
- Boundaries themselves have the content.
- Teleportation between multiple "rooms"/modules (suggestion from our side).

We had a bit of a hard time visualising exactly what we were supposed to do and got that cleared up during the retrospective. For this sprint we spent quite some time familiarising ourselves with Unity. Because of that we did not finish all the tasks we were given. We finished most, but the big one that was unfinished was the room generation tool.

4.4.4 Sprint 2

At the sprint meeting, the Product Owner decided that we should keep using the old version of SteamVR (for more on that see section 7.5).

We finished the tool for generating rooms and tried making some of the room sizes and fill them with furniture to see how we felt about the size. We concluded that we clearly needed to do something different than what we had originally been tasked with, and so we needed proof that this was the case. We decided to do an experiment on this and got it green lighted by the Product Owner. We started preparing for the experiment. We finished filling the rooms with furniture and placed some objectives and made a teleport between multiple modules the size of the play area that always kept you in your relative position in the play space. And started working on the required formalities for the experiment.

At the retrospective for this sprint, we had yet to finish some things that were on the list. These things were:

- Interactable objects.
- Different movement modes (other than walking and our specialized teleportation).
- Nested prefabs (i.e. tiles with furniture in them).
- Doors.

It was largely due to the experiment taking half the sprint. We ended up using the next couple originally planned sprints for the experiment too.

4.4.5 Sprint 3

We presented our findings and the data from the experiment to the Product Owner and he agreed that we should rather do it in some other way than the one he had initially wanted. The networking was not yet ready from their side, but perhaps soon. Thus we started working on the minigames. During this sprint we made both darts and checkers and prepared a little demo where you could try out the darts. We tried it and let some colleagues try it as well in order to get some feedback on it.

At this point we were done with the experiment and could therefore focus on the sprints. Because of that we managed to complete most of the tasks planned for the sprint. We did not do the rock paper scissors minigame as we were still holding off on that (see subsection 7.7.1), and we did not manage to do the card game during this sprint either.

4.4.6 Sprint 4

We were nearing the end of the active development period and it was a possibility that the networking would never come. We started researching communication as there would be limited to no time to actually implement any form on communication, but we wanted to at least have had a look at it even if we did not get to implement any. We were ready for the possibility of a last minute networking crunch to get that up and running, but in the end that did not happen.

We decided to take some time off during Easter, after we had been a weekend at Progress Interactive working (more on that in section 7.6). That meant we had only one week left which we spent looking at communication. We still had not done any networking, but that was out of the question at this point. Other than that, we completed what we had been assigned to do this sprint, namely the dive into communication options.

Next we will go into the details of implementing the features of the project which we have been outlining in chapter 2 and 3.

5 Implementation

In this chapter, we provide a detailed description of our implementation of various features, motivate the decisions made during development, and discuss potential improvements.

5.1 Approximating aerodynamics for throwing darts

Without anything resembling aerodynamics, there was no way to ensure that the thrown darts hit with the tip first. Since the project was part of a larger game, and games need a degree of verisimilitude, it was important that we made the darts behave reasonably similar to real darts while ensuring that the darts throwing was fun. So we needed to create an approximation of aerodynamics that was accurate enough to be believable. We have simplified and summarized the effects of aerodynamics on a moving object as follows:

- It slows the object down by exerting drag, which increases with velocity
- It turns the object so that the center of drag is behind the center of mass

As we were only going to use this simplified model of aerodynamics to throw darts at low speeds over distances shorter than 5 meters, any loss of velocity we applied to the darts would not be very noticeable. We therefore decided to only implement the second point. We originally implemented this by linearly interpolating the dart's forward vector towards its velocity vector, taking both its speed and the elapsed time into account. When the tip of the dart struck something, we would check to see if the sine of the angle between the dart's forward vector and its velocity vector would be less than 0.5. If it was, the dart had hit more or less straight on, and it would stick. However, because Unity's internal physics update happens before OnTrigger functions, darts moving faster than a certain speed would bump into the darts board and bounce off before we compared the forward vector and the velocity vector, causing the dart to not stick in the board even when it hit straight on.

In addition, some of our testers commented that they would pick up the dart with one controller and use the other controller to ensure the dart was pointing forward before they threw it. It detracted from the experience and took additional time. We solved this problem and the collision problem mentioned above by making a small change to our aerodynamics and collision. Instead of linearly interpolating the darts forward vector towards its velocity vector, any moving dart would simply change its forward vector to match its velocity vector. This means that players will not have to adjust the orientation of the dart prior to throwing, and also that we can simply assume that the dart will always hit straight on, solving both a technical problem and a design problem in one stroke. The final code is shown in Figure 5.1

5.2 Checkers

When it came to the development of the checkers board game we discussed different approaches to how we wanted board games to behave.

- Have strict rules of movement of pieces. (e.g. not allowed to place piece unless valid move)

Listing 5.1: Simplified aerodynamics and collision for darts.

```
public class Aerodynamics : Monobehaviour
{
    public Transform aeroTransform;
    public Rigidbody aeroRigidbody;

    void Update()
    {
        float speed =
            aeroRigidbody.velocity.magnitude *
                Time.deltaTime;

        if (speed > 0.002f)
        {
            // Ugly line breaks to fit the page
            // Not present in actual code
            aeroTransform.rotation =
                Quaternion.LookRotation
                    (aeroRigidbody.velocity);
        }
    }

    private void OnTriggerEnter (Collider other)
    {
        if (other.isTrigger == false)
        {
            aeroRigidbody.isKinematic = true;
        }
    }
}
```

- grid based or not. (e.g. pieces snap to the grid of the board game when placed)
- have no rules of movement, players move pieces and place them where they want to. (rules enforced by the players them self)

While discussing these different approaches we decided that the first approach of strict rules sort of enforced grid based placing for the user to get a picture of allowed and disallowed moves.

For the third approach we felt like it would not make sense to implement grid based placing because one of the main advantages to this approach is that it replicates real life the best, and a grid based system would interfere with this.

We then started to discuss which approach to go for, and ended up with the third (no rules) approach because of a couple of reasons

- Feels most natural
- more flexible (easier to implement similar gametypes)
- Easier to implement (no grid based placing or enforcing of rules)
- more reliable (no edge cases of rules to handle)

but we also realized that this approach had some obvious drawbacks

- no cheat prevention
- misplacing of pieces

As we were going for a system where we could with minimal effort create other board games we started to break down features that would be reusable with minimal refactoring.

1. An interchangeable playing board where the game would be played.
2. A system to place playing pieces to the specification of the game.
3. A system of interacting with the pieces
4. A way to reset the game to the starting position

Firstly, we identified that the first and fourth system is in all practical applications the same system and this system is reliant on a specific function for each game played, we therefore separated this into three distinct functions. One for calculating where to place a piece, one for getting the piece from a prefab and one for moving said prefab to its proper location. Using the local scale of the game board we then scale both the pieces and the locations of these accordingly so that we can scale the game board to whatever size makes sense for the setting. With this system in place, the two latter functions are interchangeable and only the prefabs for the pieces and the algorithm to place these in a grid is unique for different games, in this case the GenerateCheckers is the function for generating a checkers game, while the two other functions are usable for other board games.

Generating Checkers (Listing 5.2)

For the main logic of the checkers game we have the GenerateCheckers function, which in the case of another board game would be substituted by another generator function, the main part of this function is to place the pieces on every other tile for the three first rows, first for white and then for black and it will call on the PlacePiece function which takes the prefab of the piece as well as the x,y coordinates.

Getting and Instantiate the pieces (Listing 5.3)

For the placing of the object, we create its game object with the prefab and sets its parent to be the game board, as well as placing it in the array and calling the movePiece function to move it

Moving the pieces to the right place (Listing 5.4)

In the last function we start by scaling the piece to the scale of the game board and then transforming its position according to the scale and x,y coordinates. The flowchart of this logic can be seen on this flowchart: Figure 8

1. Generate Checkers, this is the checkers main logic
2. int y = 0, this is the loop for the white pieces, iterating over row 0,1 and 2
3. PlacePiece(int x, int y, GameObject color), initiates the piece with the prefab given (in this instance it is the color of the checkers piece, but it could be a black rook or a white queen for chess)
4. MovePiece(p, x, y). with the x,y coordinates and a reference to the gameobject, we move the piece to its location
5. int y = 7 t, this is the loop for the black pieces, iterating over row 7, 6 and 5. When y

Listing 5.2: Algorithm for placing of the checkers pieces

```
private void GenerateCheckers()
{
    //white
    for (int y = 0; y < 3; y++ )
    {
        bool oddRow = (y % 2 == 0);
        for (int x = 0; x < 7; x += 2)
        {
            PlacePiece((oddRow) ?
                x : x + 1, y, whitePiecePrefab);
        }
    }
    //black
    for (int y = 7; y > 4; y--)
    {
        bool oddRow = (y % 2 == 0);
        for (int x = 0; x < 7; x += 2)
        {
            PlacePiece((oddRow) ?
                x : x + 1, y, darkPiecePrefab);
        }
    }
}
```

> 4 is false. The program exits

6. same as 3
7. same as 4

5.3 Room generator

When we decided to run the experiment discussed in 6.1, The need to be able to generate different rooms with differing size and arrangement became apparent. after some discussion, we decided on these key functions:

1. The ability to insert any tileable prefabs for walls and floor
2. Be able to create several rooms in one scene, defining the corners x,y,z coordinates
3. specifying the size, both in terms of height and floor area, specified in tiles.
4. The ability to use different sized prefabs with the use of a factor, where 1 is 1x1m

If we take a look on the Gantt chart for the experiment Figure 7, the breakdown of the experimenting, we can see that it took 3 weeks to preparing the experiment. It was during these weeks that we created the rooms and figured that we needed a room generation tool, The experiment initially required 10 rooms and this would be tedious to implement by hand, plus for futureproofing, we wanted a systematic way to create rooms. After creating the rooms, we can then manually add furniture, but there is nothing stopping us to add furniture to modules (each floor-tile, 2x2m) beforehand so that the rooms generated would already be fitted.

Listing 5.3: Getting the prefabs and Instantiate them

```
private void PlacePiece(
    int x,
    int y,
    GameObject color)
{
    GameObject piece = Instantiate(color)
        as GameObject;
    piece.transform.SetParent(transform);
    Piece p = piece.GetComponent<Piece>();
    pieces[x, y] = p;
    MovePiece(p, x, y);
}
```

Listing 5.4: Moving piece prefabs according to the generator function

```
private void MovePiece(Piece p, int x, int y)
{
    p.transform.localScale =
        Vector3.Scale(
            p.transform.localScale,
            transform.localScale);

    Vector3 xPos = Vector3.right *
        x *
        transform.localScale.x;
    Vector3 yPos = Vector3.forward *
        y *
        transform.localScale.y;

    p.transform.position =
        (xPos) +
        (yPos) +
        Vector3.up *
        0.1f;
}
```

UI

The UI for the room generation is quite simple: Figure 10 but it has the features needed and as it is for development purposes we felt that adding the UI in VR did not make sense, as the rooms were created in a mouse/keyboard environment. The program lets you create several spaces in a single Unity Scene, but there is not a function for "merging" rooms, this will have to be done manually as of now.

functionality

When we create a room with the room generator, it spawns the tiles and names them based on their coordinates in the scene and for the wall tiles, it also names them based on its direction. Figure 9 The functionality of the room generator is based on what we needed for running the experiments, and not what we would like to have in it for a tool in the actual framework including features such as:

- change the shape of the room (e.g. rectangular rooms)

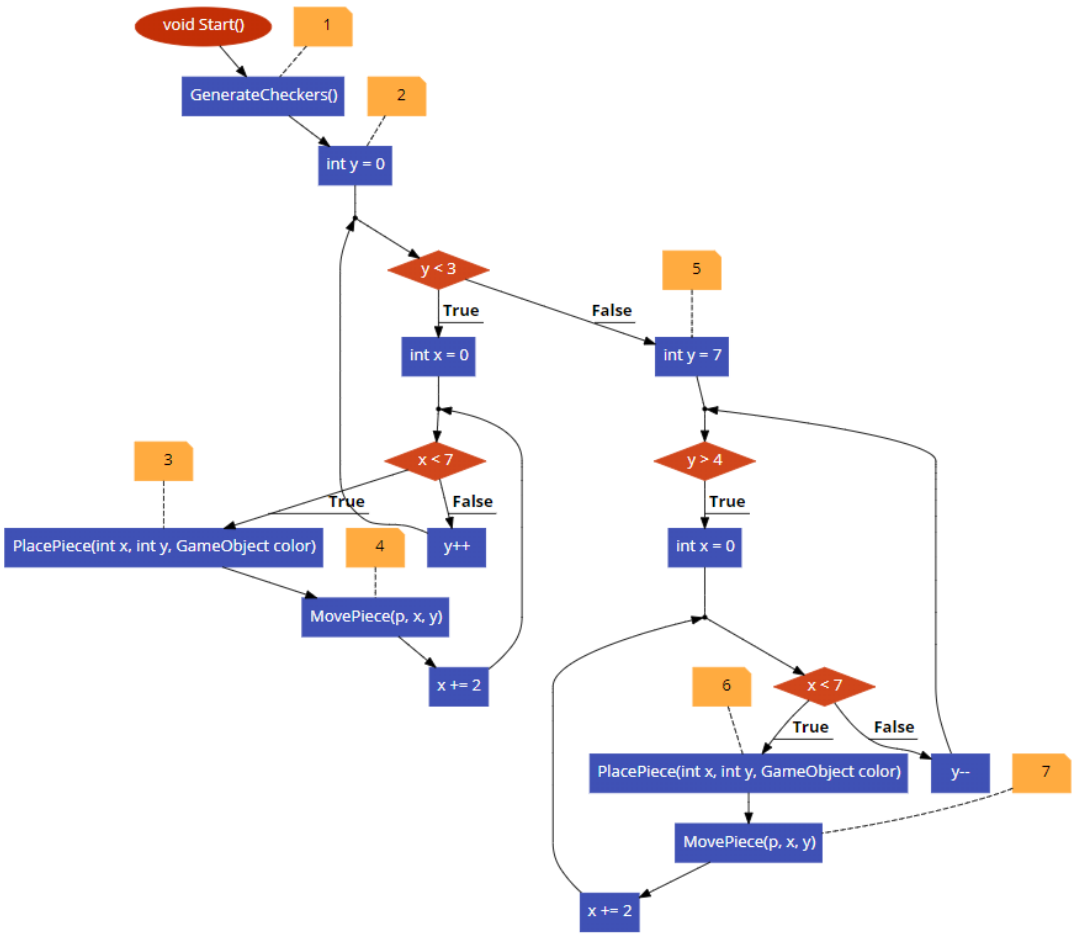


Figure 8: Flowchart of checkers logic

- "merging rooms" meaning that if two rooms overlap, the overlapping tiles would be removed so that the two rooms become one
- create furnished rooms, possibly ability to choose the theme of the interior

Another approach we had was to look around for other unity room generators, for example the works of DunGen [6] Which has a lot of cool features and good reviews, but for our purpose most of the implemented features were not applicable in our case, plus the fact that the software cost 75 USD made us just create it ourselves.

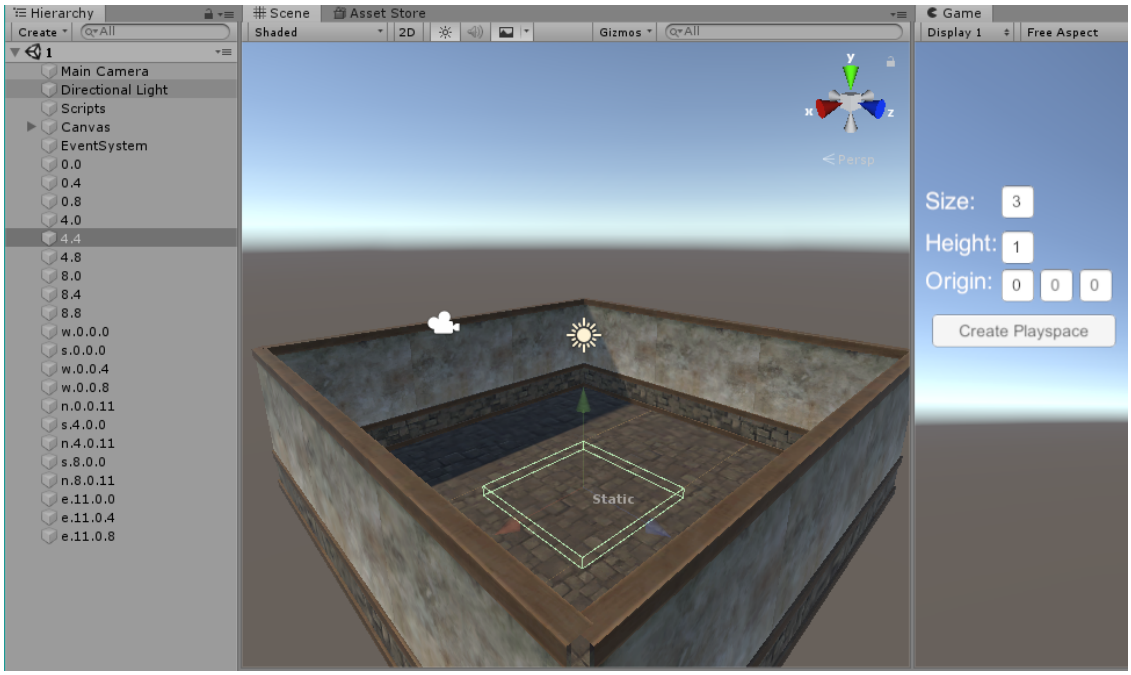


Figure 9: example of room generation

Figure 10: UI for creating Room

5.4 Implementing a movement system

As written in the project plan, the product owner initially wanted the virtual environment to have the same dimensions as the player's play area. This meant that we would not need to implement any movement system, as the player could simply walk around their play area.

However, when we ran our experiment (discussed in 6.1), we were using rooms larger than the play area we had available. Because of this, we needed a way to move around the entire room, so we started looking at options for a movement system. The options we considered were:

- **Variable movement speed.** We discussed the feasibility of making the player's movement speed proportional to the room size, so that moving from one side of the play area to the opposite side would always take the player across the virtual room. The product owner was in favor of this solution, claiming that teleportation could damage immersion. However, this would create a cognitive dissonance between how far the player feels they are moving and how far they see that they are moving. This cognitive dissonance would be further reinforced if the play area is rectangular, as movement along the long sides would be faster than movement along the short sides. Such cognitive dissonance is a known cause of motion sickness [7], which arguably would be a significantly greater obstacle to immersion than teleportation in a game that has magic. We therefore decided this option was unsuitable for our needs.
- **Traditional movement.** Movement controlled by pressing buttons or moving analog sticks. The player is stationary, and moves the in the virtual reality by using the hand controllers. This also causes a slight cognitive dissonance, though motion sickness does not seem to be very common.
- **Teleportation.** The SteamVR asset pack for Unity includes a fully fleshed out teleportation system, allowing developers to designate areas or points as a valid destination for teleportation. Players can teleport by pointing at a valid destination and pressing a button. This instant movement generally does not cause motion sickness. Our product owner suggested we change this system so that the player is moved to the destination gradually instead of instantly, but we believe the resulting cognitive dissonance would likely cause motion sickness.

Because the SteamVR asset pack for Unity has a teleportation system we could use, teleportation would be faster to implement than a traditional movement system. The earlier we could run the experiment, the more time we would have to make use of the data, so we decided we would implement teleportation. However, there was a design problem: with the addition of teleportation, players could walk to the virtual wall at one end of their play area, teleport to the opposite wall, and walk to the other side of the play area, going through the virtual wall. This would undermine the game's verisimilitude, causing players to be less invested in the game world.

Our solution requires some explanation:

1. When the player teleports using the SteamVR framework for Unity, the virtual position of the play area moves with them.
2. Each of our rooms was composed of one or more modules, and each module had the same dimensions and orientation as the play area.
3. The SteamVR framework keeps track of the player's position in the play area, and this information is accessible through a public function.
4. By giving one of these modules the same starting position as the play area, the players position relative to the play area would always equal their position relative to the module - prior to teleportation.

Listing 5.5: Moving the teleportation points corresponding to the player.

```
public class TeleportPointMovement : Monobehaviour
{
    private Vector3 startPosition;
    private Vector3 playerPosition;

    void Start()
    {
        startPosition = this.transform.position;
    }

    void Update()
    {
        // Ugly line breaks added to fit the page
        // Not present in actual code
        playerPosition =
            FindObjectOfType<SteamVR_Camera>
                ().head.localPosition;
        float newX =
            startPosition.x + playerPosition.x;
        float newZ =
            startPosition.z + playerPosition.z;
        Vector3 newPosition = new Vector3
            (newX, startPosition.y, newZ);
        this.transform.SetPositionAndRotation
            (newPosition, this.transform.rotation);
    }
}
```

In order to keep point 4 valid after teleportation, we had to ensure that the play area would always have the same position as one of the modules. We did this by placing a SteamVR teleportation point - a single valid teleportation destination - in each module at the same position relative to that module as the player's starting position relative to the play area. We also gave each of these teleportation points a simple script. Every frame, this script would ask for the position of the player relative to the play area, and move the teleportation point to the same position relative to its module. The full script is in Listing 5.5.

Admittedly, having every single teleportation point ask for the player's position every frame is not very efficient. We could instead have had a single script moving every teleport point. Making the teleport points children of the modules they are in would also save us the memory needed to keep track of the starting positions, and it would let us have the room generator discussed in 5.3 create the teleport points for us instead of placing them all manually.

5.5 Script for Testing

For the Testing of the different sized rooms we first discussed having a automated script shuffling the order of the rooms and then spawn the player in those rooms in sequence. However after some discussing about the player being influenced by the size of the previous room we decided to break the immersion in between rooms, as discussed in chapter 6. Because of this decision the scope of the script behind the testing was drastically reduced and we could save some development time. All our script needed to do was control the objectives and record the time spent, the objectives we are referring to are white sphere manually placed in the level that the player is instructed to touch as seen in Figure 32.

First for the preparation, we used random.org [8] to randomize the sequence (1,2,3,4,5) 50 times and importing it to a excel sheet so that we could report the time spent while the participants were answering the questionnaire (C), as well as loading up the next scene manually. That means that all that was left was reporting the time spent from touching the first objective until the last objective was touched and this could be achieved by two classes, one for each objective, destroying itself when touched by a vr controller, the other one counting time spent and counting the remaining objectives.

Pseudocode

As we can see in the pseudocode 5.6 we have structured the code into two classes, one is attached to the objective prefab and the other one is attached to a generic gameobject containing scripts. We will not be discussing the first class too much, as its only function is to look for collisions and destroy itself if collided with the players controller. The second class does a couple of things, first we have some variables, ints and floats, containing time spent and objectives.

In unity there are several base functions that are called at specific times during runtime, these scripts is a function from MonoBehaviour, MonoBehaviour is the base class from which every Unity script derives. In this script we will be using the void start() and void update(). Start is called on the frame when a script is enabled just before any of the Update methods are called the first time, and update is ran once every frame as long as the program runs.

1. Start In the start function, we count all the objectives
2. update We check once per second if any objectives has been touched, and if all objectives has been finished we stop the application and log the time spent

while the performance of the test most likely would not be greatly impacted by checking each frame (we group the objectives under one gameobject and count its children instead of checking the whole scene) we still do the check once per second as Unity will run as many frames as possible so even a small impact per check will be a large impact. As we can see in the pseudocode 5.6, we do every check once per second in the update function, in the Objective class we do the check in a OnTriggerEnter function, which checks every frame, but for the other checks it is sufficient to do once per second, as we will only be reporting time spent in whole seconds, and this will make unity have to do these checks approcimatly 1/90 as often.
In the next chapter we will discuss why this experiment was needed, what the goals of the experiment was and also, of course our findings.

After implementation, we will now talk about an experiment we did and then go into

Listing 5.6: Script for testing roomscale

```
Class Objective
        check if anything collides
                check if what collided is tagged "hand"
                        Destroy objective

Class Counterscript
        float timer
        int objectivecount, seconds and totalobjectives

        start (runs once at start of application)
                count all gameobjects tagged "objectives"
                store the amount of objectives in totalobjectives
                set objectivecount to equal totalobjectives

        update (runs once per frame)
            timer += time.DeltaTime;
            check if timer is larger than 1
                recount the objectives
                reduce timer by one
                check if objectivecount is smaller than 1
                    log the seconds spent
                    quit the application
                check if first objective is touched
                    increase seconds by one
```

some testing of what was implemented and then we will take a look at results of these.

6 Testing and Experimenting

In this chapter we will discuss tests and experiments we conducted during the development process. These have provided us with valuable quantitative and qualitative feedback, which we have used to inform our design choices. The results, as well as our interpretation of the results, will be presented and discussed in Chapter 7.

6.1 Testing the scale of the room

Around the middle of March, our product owner said they wanted the game's "home base" level, with up to 4-players, to scale 1:1 to match the play area of the player with the smallest play area, to a minimum of 2x2 meters. We believed that 2x2 meters would not be nearly enough to have four players performing the activities available in this level, but we needed solid data to back up our claim. We conducted an experiment between the 21st of March and the 25th. The motivation was to find out how large the level needed to be to fulfill the needs of the product owner.

The questions we wanted to answer were:

1. How many people can comfortably fit in these rooms, while performing day-to-day activities such as the minigames described in the project requirements (see Appendix A)? In the finished game, this level serves as a home base, so as level designers we need to ensure that it is not only large enough for four people, but also that it is suitable for the kind of activities you might do with guests at your home.
2. Are these rooms comfortable to move around in? Why/why not? To assist in our level design, it would be beneficial to understand how factors such as ceiling height, percentage of floor area taken up by furniture, windows etc. impacts how comfortable people feel in a room.
3. Does any prior experience with virtual reality the participants may have impact the results? How? For the finished game, having data on how humans adapt to virtual reality after repeated exposure could help predict how new players change their behaviour as they become familiar with the environment, which could potentially improve the quality of level design.

We looked at earlier work to find answers to these questions, and found numerous studies that dealt with perception of distance and/or scale in immersive virtual environments. In particular, we found the following studies interesting:

- Gooch and Willemsen, 2002[9], *Evaluating Space Perception in NPR Immersive Environments*. Particularly, this study found that in a walking task, subjects perceived distances as roughly 66% of the real distance, and that linear perspective cues can convey absolute distance.
- Messing and Durgin, 2005[10], *Distance Perception and the Visual Horizon in Head-Mounted Displays*. Particularly, this study found that the compression of perceived distance is a function of the head-mounted display. It does not change with distance, and

is unaffected by graphical quality, persisting even viewing a live-feed of the real world.

- Sinai, Krebs, Darken, Rowland and McCarley, 1999[11], *Egocentric Distance Perception in a Virutal Environment using a Perceptual Matching Task*. Particularly, this study found that a perceptual matching task may result in significantly more accurate distance judgment than other methods, and that ground texture had significant effect on accuracy, though it does not establish an exact cause for this.
- Willemsen and Gooch, 2002[12], *Perceived Egocentric Distances in Real, Image-Based, and Traditional Virtual Environments*. Particularly, this study found that graphical quality may not have any significant effect on distance compression, and suggests that the head-mounted display is one cause of compression.
- Witmer and Kline, 1998[13], *Judging Perceived and Traversed Distance in Virtual Environments*. Particularly, this study suggests that traversing a distance improves the ability to estimate that distance, and that the type of movement does not affect the estimation.
- Witmer and Sadowski, 1998[14], *Nonvisually Guided Locomotion to a Previously Viewed Target in Real and Virtual Environments*. Particularly, this study suggests that practicing distance estimation in a virtual environment can impair real world distance estimation, and that practicing real-world distance estimation can improve distance estimation in a virtual environment.

While it was interesting to us to see that all of the above-mentioned studies reached the conclusion that distances are perceived as shorter in immersive virtual environments, none of them explored how that impacts how crowded a room feels or what role, if any, experience with virtual reality plays in this. As for how comfortable the rooms are to move around in, we did not find any studies that explore how this compressed depth perception affects how cramped a room feels. Studies on color and furniture arrangement in interior design were deemed too inconclusive to be of significant use to us. Studies on lighting in interior design were deemed not applicable, as the lighting models used for real-time lighting in games usually differ greatly from real light.

In preparation for the experiment, we developed a system for generating unfurnished rooms (discussed in 5.3), and with it prepared five models of rooms of varying shape and size, composed of one or more 2x2 meter modules, fully furnished and with objectives to be gathered. The rooms we prepared were as follows:

1. One module (2x2 meters, 4 square meters)
2. Two modules, linked with teleportation (2x4 meters, 8 square meters)
3. Three modules arranged to resemble a corridor, linked with teleportation (2x6 meters, 12 square meters)
4. Four modules arranged in a square, linked with teleportation (4x4 meters, 16 square meters)
5. Three modules arranged in an L shape, linked with teleportation (4x4 meters minus 2x2 meters, 12 square meters)

You can find screenshots of the rooms showing their layout as well as where the objectives in the rooms are in Appendix E.

We prepared a set of questions (see Appendix C) for the participants, starting with asking how much VR experience they had, followed by two questions for each room and finally ending in a free text question where they could write any comments or thoughts they had

about the experiment. We gave them a short verbal introduction to what they were going to do (try to get a feel of the room and touch the objectives we had placed around the rooms), and how to move around in our rooms (Walking and a variation of the standard SteamVR teleportation 5.4). After the introduction, the process was as follows:

1. The rooms are arranged in a random order. The order is recorded and associated with the participant number, so that we know which of the now random room numbers corresponds to which room.
2. The participant answers the question of how much experience they have with virtual reality.
3. The participant puts on the equipment and explores the first room of their random room order. A script measures how much time it takes the participant to finish exploring the room.
4. The participant takes off the equipment and answers two questions about the room they just explored.
5. Repeat step 3 and 4 for the remaining four rooms.
6. Finally, the participant writes down any comments, questions or general feedback.

We wanted to reduce the impact of deviations caused by comparing rooms to earlier rooms as well as preempting experimental fatigue. This is why we had the participants take off the equipment and answer questions between rooms, instead of letting them explore every room before answering the questions. It also ensured that the memory of each room was fresh when the questions were answered. Minimizing deviations is also why we randomized the order in which the rooms were presented, so any deviations would hopefully cancel each other out.

We will present the results and discuss our interpretation of the results in the next chapter.

6.2 Testing the darts throwing

Prior experience with implementing throwing in virtual reality led us to believe that implementing a system for throwing in a way that is satisfying for the players is significantly more complicated than simply letting the player pick things up and fling them. With that in mind, we quickly implemented a prototype of throwable darts with an approximation of aerodynamics (5.1), and started gathering qualitative feedback. The people at Progress Interactive supplied us with a Unity prefab of a dartboard they had made. We wrote down verbal feedback, but we did not measure or record the exact results. The process was more or less as follows:

1. The player throws three darts at a darts board from a distance of 2.5 meters.
2. The player fetches the darts and repeats the process from step 1, until they do not want to throw darts anymore.
3. Verbal feedback is provided at any point during or after this process.

Among our colleagues, throwing in virtual reality is widely considered to be significantly less accurate compared to throwing in reality. We believe this is partially due to the lack of precision manipulation of the projectile during the release, using the fingers. In virtual reality, the projectile is either held or not held; there is no middle ground. It could also be partially because computer simulated physics in games is an approximation sacrificing accuracy for

the ability to run with only the limited computing power of the computers available to the average consumer[15]. Nevertheless, we did discover a technique that substantially improved accuracy, which we will discuss as part of the results of this experiment in the next chapter.

Following this description of the performed experiments, we will now turn to the presentation and discussion of the experimental results.

7 Discussion

In this chapter we will discuss the experiment and tests we did and also any major decisions done over the course of the project. We will start by looking at the results from the room scale experiment, and any findings we did when looking at the data. Then we will look at flaws and potential improvements for our experiment. Following that we will talk a little about future research that could be done with basis in our experiment. After that we will go over the feedback from the testing of the darts and what we did with that. We will then go into detail about options for communication, and what would be best for this type of game. We will touch upon the reasons for changing the Unity version and why we kept using the old SteamVR framework. Finally we will talk about why did not do networking and some minigames that we originally were going to do.

7.1 Results of the room scale experiment

In this section we will present the results of the performed experiment and discuss our interpretations of the results. We further propose how the experiment could be improved, and sketch further experiments we would like to perform based on these results. The complete results can be found in Appendix D. To reiterate, these were the questions we tried to answer:

1. How many people can comfortably fit in these rooms, while performing day-to-day activities such as the minigames described in the project requirements (see Appendix A)?
2. Are these rooms comfortable to move around in? Why/why not?
3. Does any prior experience with virtual reality the participants may have impact the results? How?

7.1.1 Results

To respond to question 1 and 2, we explored the results across different room sizes as shown in figures 11, 12 and 13 below. Figure 11 shows the results for question 1, figure 13 shows the results for question 2, and figure 12 shows how much time players spent in each room.

From the results depicted in the graphs, we can see that the data seems to scale mostly linearly with the area of the room. Room three and room five have the same area, but different shapes, and the results for these two rooms are nearly identical. This suggests that the shape of the room might not play any significant role, but more testing is needed. If the results continue to scale linearly with room area, we would need a room of roughly twenty square meters to satisfy the product owner's desire to have four players in the room. This addresses question 1.

Had we more time, we could have prepared multiple variants of each room, each variant deviating from the standard room in a single factor such as light level, presence of windows, ceiling height etc. Changing only one factor at a time would have let us attribute any changes in the participants comfort level to that attribute, which would provide an understanding of why the room is or is not comfortable. We have looked for similar work in research papers on

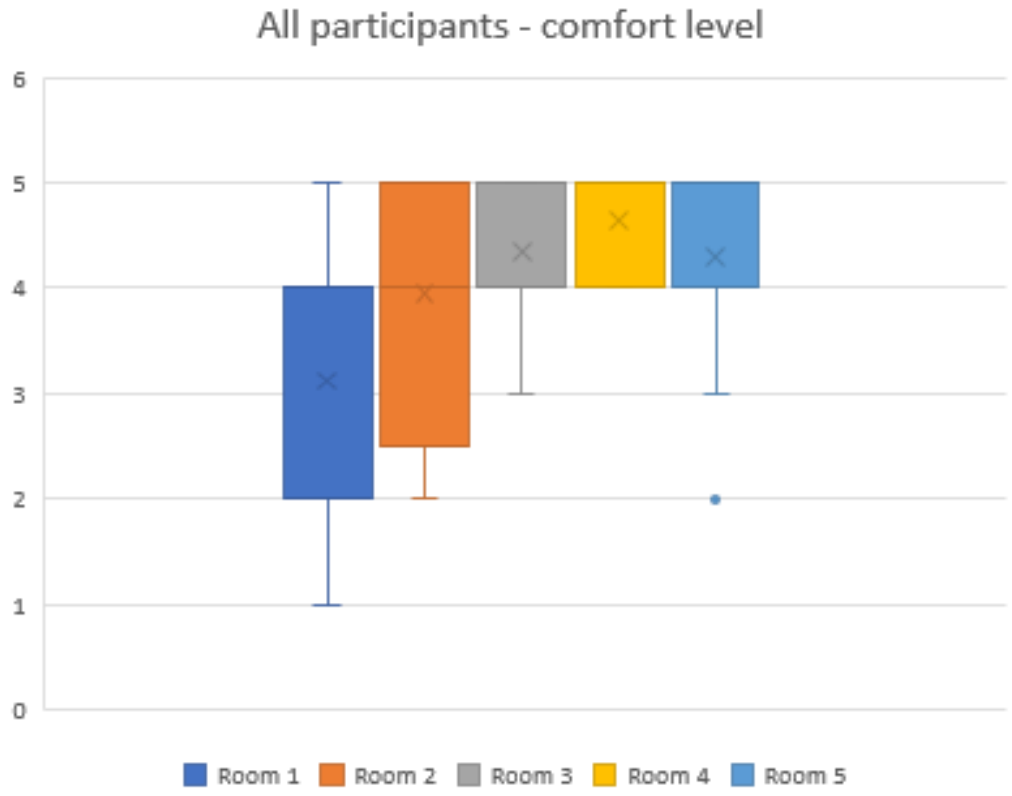


Figure 11: Comfort level, all results

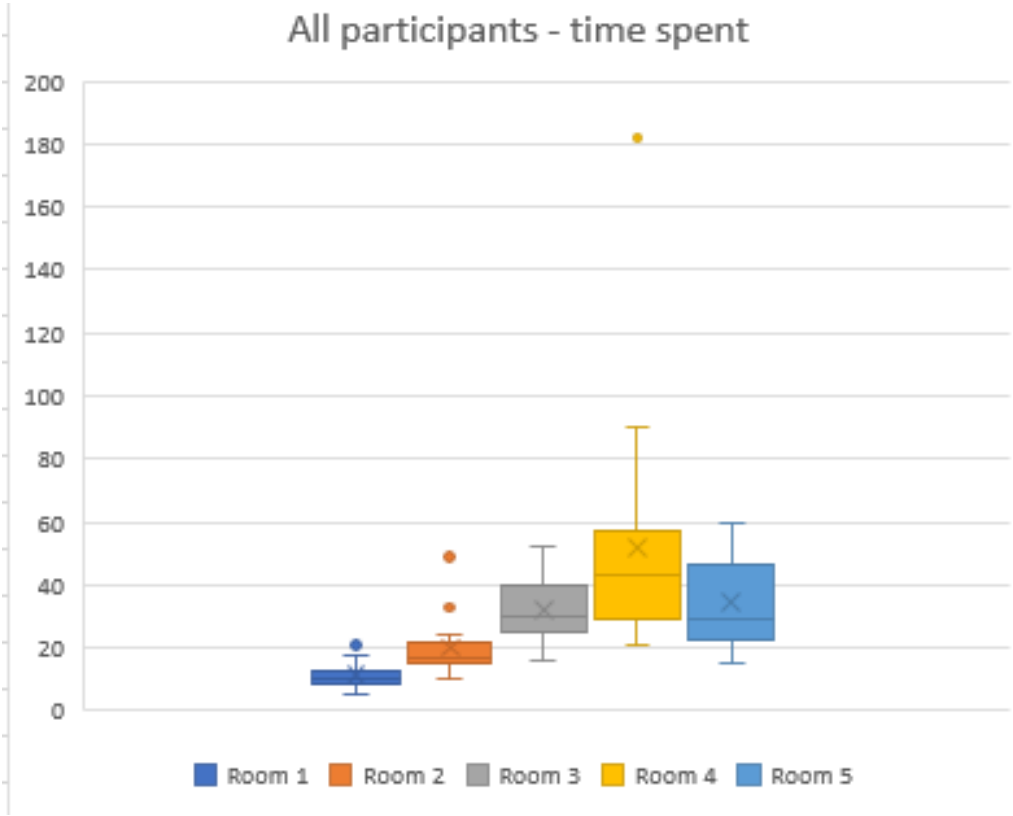


Figure 12: Time spent, all results

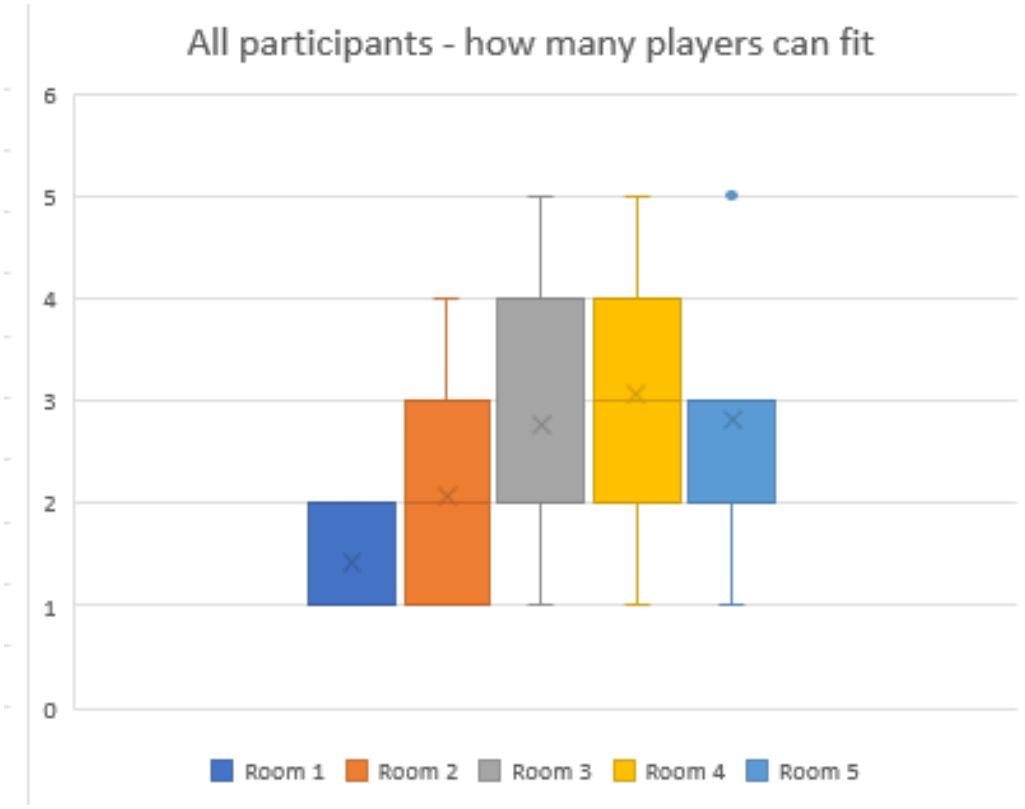


Figure 13: Player capacity, all results

interior design, but have not found anything exploring the impact of each factor separately. As it stands, we have only measured what difference changing the area of the room makes, and so we have not sufficiently addressed question 2.

To address question 3, we divided the results into three groups based on the participants experience with virtual reality. The groups were as follows:

- **Group 1:** No prior experience, five participants.
- **Group 2:** One to four hours prior experience, six participants.
- **Group 3:** Five or more hours prior experience, six participants.

Arranged in this manner, we found the following results:

- Group 2 spent slightly less time exploring the rooms than the other two groups. We think this is because group 1 needed time to adjust to their first encounter with virtual reality, and some participants in group 3 spent some additional time actively looking for flaws.
- Group 1 perceived room one and two, the smallest rooms, to be significantly less comfortable compared to the results of the other two groups. They also perceived the three other rooms to be more comfortable.
- Group 3 found every room to be able to fit fewer people than the other two groups, suggesting that more experienced players feel they need more room than less experienced players. In the smallest room, 2x2 meters (the size the product owner wanted four people in), every single participant in group 3 reported that the room could only fit one person. For comparison, 60% of group 1 and 66.67% of group 2 reported that the room could fit two people, and the remaining participants all reported that it could fit one person.
- Group 1 found room three and five, both twelve square meters, to be able to fit more people than the larger room four, which was sixteen square meters. This may be because walking from one end of the room to the other is longer.

Interestingly, the data suggests that players with little or no experience need more room to feel comfortable, and more experienced players can feel comfortable in smaller rooms but do not tolerate as many actors (other players/non-player characters) before they feel the room is cramped. In any case, we have clearly established that experience does make a measurable and significant difference, and we have some data on what difference it makes.

7.1.2 Suggested improvements for the experiment

For the question of how many players could comfortably fit in a room, when we wrote the questionnaire, it did not occur to us that anyone would want to answer zero or above five. In hindsight, instead of presenting options from one to five, it might have been better to let the participants write any number. This may have skewed the results somewhat, since for any answer of one or five it is possible that the participant would have answered zero or above five instead, had that been an option.

Of the five rooms we used, room three and room five had the same area, but different shape. In hindsight this was useful because the results suggest that the shape of the room might not play any significant role in how comfortable and spacious the room is perceived to be. However, since we do not have similar comparisons for the other rooms and our sample size is relatively small, this might be a fluke. If we ever run this experiment again we would

like to have multiple sets of three rooms, with all rooms in a set sharing the same area but different shapes. This would make any resulting findings more credible.

As discussed in 7.1.1, we should have prepared multiple variants of each room, in order to measure how factors such as light level, ceiling height and presence of windows affect the participants comfort level and possibly other metrics as well.

Several participants, and a few of our colleagues, have commented that the ceiling of the rooms should have been higher (the ceiling was fixed at two meters). This may have made the rooms feel less comfortable than they would have been otherwise, especially for taller participants. We did not record the height of participants, so we do not know exactly what difference, if any, this would have made.

In the sprints following the experiment, we implemented darts throwing and a checkers game. Having those activities in the rooms used in the experiment could have given the participants a better idea of how much space they would need to perform activities, resulting in more accurate data.

7.1.3 Future research

We still do not know what factors other than room area affect comfort level, or how those factors could affect other metrics. In order to find out, we would like to run the experiment again, implementing the changes suggested in 7.1.2 and with a significantly larger sample size. We believe that the results would be interesting for both level designers and interior designers.

7.2 Feedback from testing the darts throwing

Looking at the second experiment, we gained tractable practical insights for the improvement of physical games in VR. While testing different ways of throwing darts, we found a technique that seems to improve accuracy greatly, depicted in figure 14. By holding the dart between the darts board and the thrower's chest, rapidly accelerating the hand towards the darts board and flicking the wrist to give the dart extra speed, some testers were able to consistently land all three darts within 20 centimeters of each other. One tester commented that this is very similar to how professional darts players throw.

Figure 14: Dart throwing technique

Before discovering the better throwing technique, testers who felt they were not able to throw hard enough tried throwing in an arc. Some of these testers commented that the ceiling was too low (again, it was only 2 meters) and that the darts were striking the ceiling. This was after the experiment discussed in 7.1, so there was nothing stopping us from simply increasing the height of the ceiling by 20 centimeters. We did not get any more complaints about the ceiling height after that.

Some of the people we tested with claimed that the target zones on a modern darts board are much too small to reliably hit in virtual reality, even with professional darts throwing techniques. In response to this, the artists at Progress Interactive redesigned the darts board to match a finding of a medieval darts board. This new board had significantly larger target zones, but this came too late for us to test what difference this made.

7.3 Communication options and quick chat

The product owner wanted the finished game to allow players to communicate with each other, but it had to be appropriate for minors as well as adults. This meant that we could not use a traditional text chat, nor Voice over Internet Protocol (VoIP, also called IP telephony), as both would allow players to potentially share inappropriate content. In short, any open system in which the users can say, write or depict whatever they want would not be suitable. This leaves us with the following options:

- **A moderated text chat with a swear filter.** Most MMORPG's have some functionality of this kind. However, it is not very practical in virtual reality. The players do not have a physical keyboard, and though it would be possible to have an in-game keyboard, typing out messages by pointing and clicking at one button at a time would take a lot of time, making it unsuitable for all but the most relaxed of situations.
- **An emote system.** Since a text chat is not suitable for our needs, we will not explain emotes in the context of text based chat systems. In non-text based chat systems, an emote is an animation used to communicate via the body language of the player's avatar. Emotes may or may not be accompanied by sound effects or prerecorded dialogue. Games such as From Software's *Dark Souls*[16] have used emotes as the only means of real-time communication with success, but one major flaw of this option is that since the communication often takes place entirely via body language, it can take quite some time to make oneself understood. In addition, moving in any way such as walking or fighting while the animation plays will change the body language, making it more difficult to interpret correctly. Since combat is a major element of MMORPG's, this makes emotes without accompanying text unsuitable for our needs.
- **A Quick Chat system.** In a Quick Chat system, users communicate via short messages predefined by the developer. These can be text messages or prerecorded voice messages. In either case, the messages can have additional effects, e.g. telling another player to move to a given location could create a marker at that location, providing the other player with clear directions in an instant. These systems emphasize near-instant communication, so it must be designed in such a way that players can navigate to any of the predefined messages quickly. Some games have implemented this by presenting a wheel encircling the middle of the screen, divided into sections containing one message per section. By moving the cursor and clicking (or moving the analog stick if using a controller) one of these sections, players can select the message they want in the span

of a second. This implementation would map nicely to the circular touchpads on the HTC Vive's controllers or the analog sticks of the Oculus Rift. An example of a quick chat system is shown in figure 15.

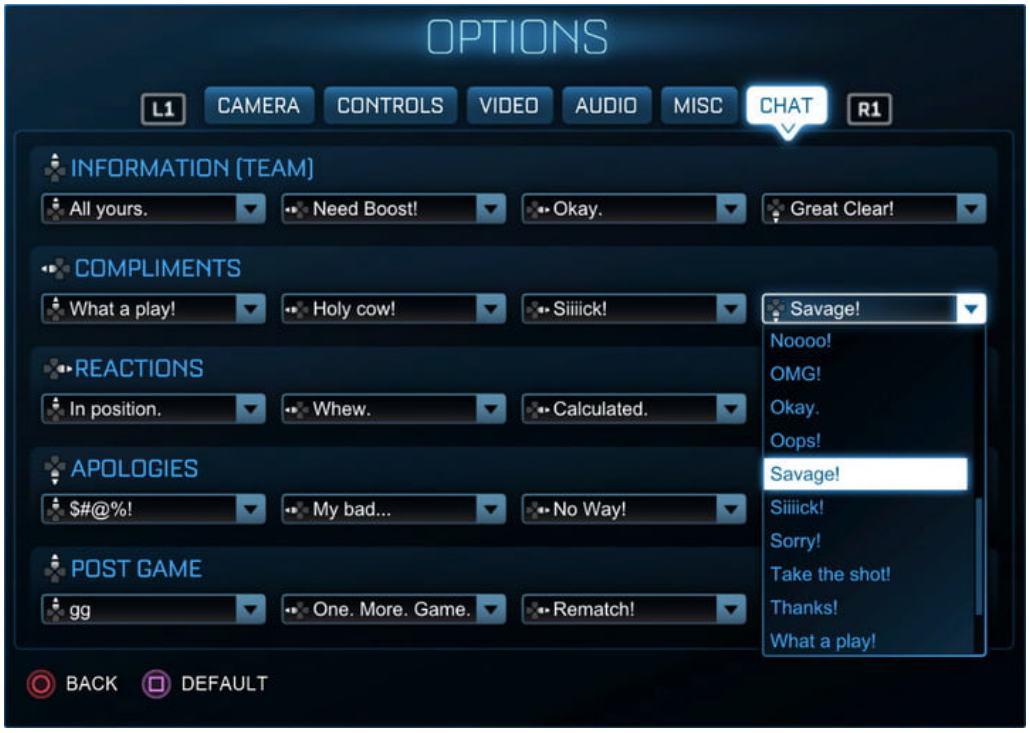


Figure 15: Customisable configuration of quick chat, from *Rocket League*. Note that any of these messages can be sent with only two key presses.

7.4 Changing Unity version

The project plan originally said we were to use Unity version 2018.2.14f1, because that was what Progress Interactive was using for this project. However, less than a week into development the product owner wanted us to change to Unity version 2018.3.4f1, primarily because it had nested prefabs and prefab variants. We will briefly explain what these terms mean, and why they are useful to us.

From the Unity documentation[17]: "*You can include Prefab instances inside other Prefabs. This is called nesting Prefabs. Nested Prefabs retain their links to their own Prefab Assets, while also forming part of another Prefab Asset*". Before nested prefabs, if you wanted a prefab to contain an instance of another prefab, the child would become a part of the parent prefab and would no longer be an instance of its prefab asset. Because of this, any changes to the child would not be applied to its former prefab asset or other instances of said prefab asset unless you manually made the same changes to the prefab asset. And even then, further changes to the prefab would not affect any other prefab that contains an instance of it. They would also have to be changed manually. With nested prefabs, changes to a prefab can be applied to instances that are children of other prefabs and vice versa. This is significantly less time consuming and less prone to human error.

From the Unity documentation[18]: "*A Prefab Variant inherits the properties of another*

Prefab, called the base. Overrides made to the Prefab Variant take precedent over the base Prefab's values". Before prefab variants, if you wanted three different versions of the same thing, you would have to create every shared attribute three times. In order to change one of these shared attributes, you would have to make the change in every version manually. Alternatively you could create a tool for creating new versions and changing existing ones, but that would assume you to be a skilled programmer. With prefab variants, you can simply change the base prefab. This feature lets non-programmers create new variants fast without relying on programmers. For smaller companies, the cost of employing skilled programmers and developing their own tools can be insurmountable. By increasing the productivity of non-programmers and letting small companies get away with fewer employees, this makes it easier for small companies to turn a profit, which in turn promotes growth in the games industry.

As for why we did not update to an even newer Unity version, the product owner made it clear that though they wanted us to integrate with a company-provided networking solution, they did not want us to worry about integrating with the larger game. This project was more of a proof of concept, and as such the highest priority was to get it playable fast so we could use it to playtest and gather data that would be useful in designing the final version. The extra work needed to update to an even newer version of Unity would take precious development time.

7.5 Reverting to a previous version of the SteamVR framework

Valve released an update to the SteamVR framework (SteamVR 2.0) that Progress Interactive's existing systems were not compatible with, and Valve did not update the documentation, so we could not figure out how to get everything to work in the new version. The product owner figured that reverting to the version they had been using previously would be significantly less work than trying to update everything to the new version. Since our highest priority was to get the prototype playable fast, it made sense to revert to the previous version.

7.6 Networking

In the project plan, the project was going to integrate with an existing networking solution from Progress Interactive, which the product owner told us would be finished around Easter. We were invited to come to Progress Interactive's offices to work over the first weekend of Easter. In a sprint meeting two weeks prior, we had been told that this weekend we were to work on integrating with the networking solution, but it was not yet finished. Instead, the product owner wanted us to work on refactoring the Unity project we used for our experiment (6.1) and integrating it with the project's primary repository. As we were nearing the end of the sixth and final sprint, Progress Interactive had still not provided a finished networking solution. We will therefore instead describe in broad terms one way we could have used an example networking solution, based on our impressions of what Progress Interactive had planned.

As described in 2.2, we were to develop "a bespoke RakNet-based[19] multiplayer solution for Unity3d using a master-client architecture and the external company's remote log in server". Using RakNet makes this job significantly easier, since it already has Unity integration and many features that we would otherwise have to implement ourselves. Object replication, security, and an extensive lobby system is already implemented and ready for us

to use. Progress Interactive's remote login server would authenticate players before letting them into the lobby system.

7.7 Minigames we did not implement

There were some minigames in the project requirements that we ended up not implementing for various reasons. We will discuss the reasons for why we did not implement these minigames here.

7.7.1 Rock, paper, scissors

An integral part of the rock paper scissors game would be the synchronisation of the two players involved, the game itself is not something one can play alone. Because of this, we could not implement this feature until after we had the networking in place. We instead planned for this feature to be one of the first things implemented after we got the networking up and running as a proof that the networking solution was actually working properly. Because we did not receive the networking solution before our implementation deadline, we could not implement this feature.

7.7.2 Bespoke card game

According to the project description (see Appendix A), the graphics and design of the card game was to be provided by Progress Interactive. However, this was a "nice to have"-feature, to be implemented only if there was enough time (see project plan, Appendix Put project plan in appendix). We left this feature in the project backlog until the start of the final sprint, when we decided to abandon it due to time constraints.

8 Conclusion

At the start of this project, we set out to make a social space in VR where friends could hang out and do activities together. To do this, we had to make a social space for them to be in and flesh it out with activities to do.

The social space were to be in the form of a room with the feeling of a small cottage. The original idea from the Product Owner was to have this room be the size of the play area of the player with the smallest play area in the group. This proved to be too small and so we had to gather conclusive evidence of this. We did that in the form of an experiment where we asked how people felt about the size of a few different rooms. We go into detail about how we made and conducted the experiment in section 6.1. We managed to gather data that pointed to the fact that this was indeed the case and we needed another solution. We discuss the results in section 7.1.

The activities to do were in the form of minigames. These were relatively straightforward to make and there were no major hurdles from our side. Nevertheless there were problems. Mainly that we didn't get what was promised from the Product Owner, which led to the postponing and eventual abandonment of some minigames. See section 7.7 for a closer look at why.

We ran a test on one of the minigames, darts, to get feedback on how people felt about it and if there were anything we could do to make it better. It turned out to be a little too difficult and we figured the best way to make it easier was to change the dart board. More on the dart experiment in section 6.2 and the results in section 7.2.

8.1 Future Work

From here the project could get networking in place and completion of the minigames we didn't do.

Future work on our research would be testing more/other factors than just the room size. That would be as alluded to in subsection 7.1.2, light level, ceiling height and windows and the things suggested by our Product Owner that we listed in subsection 4.4.3, counters at the edge of the play area to stop you from moving but extending your view beyond and being able to see over boundaries between rooms. It would also be interesting to test more closely whether the shape of the room had any impact.

there are many more things to say

Bibliography

[1] Gartner. Top trends in the gartner hype cycle for emerging technologies, 2017, 2019.

[2] Gartner. 5 trends emerge in the gartner hype cycle for emerging technologies, 2018, 2019.

[3] Wikipedia contributors. Rock, paper, scissors — Wikipedia, the free encyclopedia, 2019. [Online; accessed 18-May-2019].

[4] Unity. Unity download archive, 2019.

[5] Jeff Sutherland and Ken Schwaber. The scrum guide, 2019.

[6] Aegon Games LTF. Dungen, May 2019.

[7] Craig R Sherman. Motion sickness: Review of causes and preventive strategies. *Journal of travel medicine.*, 9(5):251–256, 2002.

[8] Random.org. Random.org, May 2019.

[9] Amy Ashurst Gooch and Peter Willemsen. Evaluating space perception in npr immersive environments. In *Proceedings of the 2Nd International Symposium on Non-photorealistic Animation and Rendering*, NPAR '02, pages 105–110, New York, NY, USA, 2002. ACM.

[10] Ross Messing and Frank H. Durgin. Distance perception and the visual horizon in head-mounted displays. *ACM Trans. Appl. Percept.*, 2(3):234–250, July 2005.

[11] Michael J Sinai, William K Krebs, Rudy P Darken, JH Rowland, and JS McCarley. Egocentric distance perception in a virutal environment using a perceptual matching task. In *Proceedings of the Human Factors and Ergonomics Society Annual Meeting*, volume 43(22), pages 1256–1260. SAGE Publications Sage CA: Los Angeles, 1999.

[12] Peter Willemsen and Amy A Gooch. Perceived egocentric distances in real, image-based, and traditional virtual environments. In *Proceedings IEEE Virtual Reality 2002*, pages 275–276. IEEE, 2002.

[13] Bob G Witmer and Paul B Kline. Judging perceived and traversed distance in virtual environments. *Presence*, 7(2):144–167, 1998.

[14] Bob G Witmer and Wallace J Sadowski Jr. Nonvisually guided locomotion to a previously viewed target in real and virtual environments. *Human factors*, 40(3):478–488, 1998.

[15] C. Hecker. Physics in computer games, 2000.

[16] From Software. Dark souls: Prepare to die edition, 2019.

[17] Unity. Nested prefabs, 2019.

[18] Unity. Prefab variants, 2019.

[19] RakNet. Raknet 4, 2019.

A Project Description

Title: **Resocialization for Vulnerable Individuals in VR multiplayer**
Level: BSc
Study programme: BPROG
Owner: Richard Barlow
Department: IDI Gjøvik
Project relevance: excited
Category: Health
Team size: 1-3

Description

Virtual reality offers socialisation at a distance for physically isolated and anxious or antisocial individuals. The goal of the project is for students to create social VR environment, where players can play familiar games in a room-scale environment, similar to a cottage.

The students will work for an external Game Development company based in Hamar, although occasional assistance will be provided on the Gjøvik campus by company staff. Art assets, core systems and feedback on programming implementation will be provided by the company.

The programming assignment needs to submitted, checked for appropriateness and then feedback provided to the students at regular intervals.

Technical Solution

- Develop a bespoke Raknet-based multiplayer solution for Unity3d using a master-client architecture and the external company's remote login server.
- Investigate voice-based communication options
- C# client and server-side VR implementation of familiar minigames in Unity3d, such as:
    - Checkers
    - Darts
    - Rock, paper, scissors
    - Bespoke card game (external company provides graphics and design)
- Virtual Reality implementation using Steam VR for compatibility with both Oculus VR equipment and the HTC Vive.
- Students submit code to the external company URL in a private git repository. The Computing Department in Gjøvik will also have access to the URL to monitor progress.

B Project Agreement

Sometimes you need to include a PDF document in the appendix. It is perfectly acceptable to have most of this page blank as it acts as a cover page for the appendix. We use pdfpages for this include. It lets you specify which pages to include, currently we have left it blank with `[pages={-}]` but you could use `[pages={1-3}]` for example

NTNU

Vår dato Vår referanse

Norges teknisk-naturvitenskapelige universitet

Prosjektavtale

mellom NTNU Fakultet for informasjonsteknologi og elektroteknikk (IE) på Gjøvik (utdanningsinstitusjon), og

_____

_____ (oppdragsgiver), og

_____

_____

_____ (student(er))

Avtalen angir avtalepartenes plikter vedrørende gjennomføring av prosjektet og rettigheter til anvendelse av de resultater som prosjektet frembringer:

1. Studenten(e) skal gjennomføre prosjektet i perioden fra _____ til_____ .

   Studentene skal i denne perioden følge en oppsatt fremdriftsplan der NTNU IE på Gjøvik yter veiledning. Oppdragsgiver yter avtalt prosjektbistand til fastsatte tider. Oppdragsgiver stiller til rådighet kunnskap og materiale som er nødvendig for å få gjennomført prosjektet. Det forutsettes at de gitte problemstillinger det arbeides med er aktuelle og på et nivå tilpasset studentenes faglige kunnskaper. Oppdragsgiver plikter på forespørsel fra NTNU å gi en vurdering av prosjektet vederlagsfritt.

2. Kostnadene ved gjennomføringen av prosjektet dekkes på følgende måte:
   - Oppdragsgiver dekker selv gjennomføring av prosjektet når det gjelder f.eks. materiell, telefon/fax, reiser og nødvendig overnatting på steder langt fra NTNU på Gjøvik. Studentene dekker utgifter for ferdigstillelse av prosjektmateriell.
   - Eiendomsretten til eventuell prototyp tilfaller den som har betalt komponenter og materiell mv. som er brukt til prototypen. Dersom det er nødvendig med større og/eller spesielle investeringer for å få gjennomført prosjektet, må det gjøres en egen avtale mellom partene om eventuell kostnadsfordeling og eiendomsrett.

3. NTNU IE på Gjøvik står ikke som garantist for at det oppdragsgiver har bestilt fungerer etter hensikten, ei heller at prosjektet blir fullført. Prosjektet må anses som en eksamensrelatert oppgave som blir bedømt av intern og ekstern sensor. Likevel er det en forpliktelse for utøverne av prosjektet å fullføre dette til avtalte spesifikasjoner, funksjonsnivå og tider.

4. Alle bacheloroppgaver som ikke er klausulert og hvor forfatteren(e) har gitt sitt samtykke til publisering, kan gjøres tilgjengelig via NTNUs institusjonelle arkiv hvis de har skriftlig karakter A, B eller C.

Tilgjengeliggjøring i det åpne arkivet forutsetter avtale om delvis overdragelse av opphavsrett, se «avtale om publisering» (jfr Lov om opphavsrett). Oppdragsgiver og veileder godtar slik offentliggjøring når de signerer denne prosjektavtalen, og må evt. gi skriftlig melding til studenter og instituttleder/fagenhetsleder om de i løpet av prosjektet endrer syn på slik offentliggjøring.

Den totale besvarelsen med tegninger, modeller og apparatur så vel som programlisting, kildekode mv. som inngår som del av eller vedlegg til besvarelsen, kan vederlagsfritt benyttes til undervisnings- og forskningsformål. Besvarelsen, eller vedlegg til den, må ikke nyttes av NTNU til andre formål, og ikke overlates til utenforstående uten etter avtale med de øvrige parter i denne avtalen. Dette gjelder også firmaer hvor ansatte ved NTNU og/eller studenter har interesser.

5. Besvarelsens spesifikasjoner og resultat kan anvendes i oppdragsgivers egen virksomhet. Gjør studenten(e) i sin besvarelse, eller under arbeidet med den, en patentbar oppfinnelse, gjelder i forholdet mellom oppdragsgiver og student(er) bestemmelsene i Lov om retten til oppfinnelser av 17. april 1970, §§ 4-10.

6. Ut over den offentliggjøring som er nevnt i punkt 4 har studenten(e) ikke rett til å publisere sin besvarelse, det være seg helt eller delvis eller som del i annet arbeide, uten samtykke fra oppdragsgiver. Tilsvarende samtykke må foreligge i forholdet mellom student(er) og faglærer/veileder for det materialet som faglærer/veileder stiller til disposisjon.

7. Studenten(e) leverer oppgavebesvarelsen med vedlegg (pdf) i NTNUs elektroniske eksamenssystem. I tillegg leveres ett eksemplar til oppdragsgiver.

8. Denne avtalen utferdiges med ett eksemplar til hver av partene. På vegne av NTNU, IE er det instituttleder/faggruppeleder som godkjenner avtalen.

9. I det enkelte tilfelle kan det inngås egen avtale mellom oppdragsgiver, student(er) og NTNU som regulerer nærmere forhold vedrørende bl.a. eiendomsrett, videre bruk, konfidensialitet, kostnadsdekning og økonomisk utnyttelse av resultatene. Dersom oppdragsgiver og student(er) ønsker en videre eller ny avtale med oppdragsgiver, skjer dette uten NTNU som partner.

10. Når NTNU også opptrer som oppdragsgiver, trer NTNU inn i kontrakten både som utdanningsinstitusjon og som oppdragsgiver.

11. Eventuell uenighet vedrørende forståelse av denne avtale løses ved forhandlinger avtalepartene imellom. Dersom det ikke oppnås enighet, er partene enige om at tvisten løses av voldgift, etter bestemmelsene i tvistemålsloven av 13.8.1915 nr. 6, kapittel 32.

12. Deltakende personer ved prosjektgjennomføringen:

NTNUs veileder (navn): _____

Oppdragsgivers kontaktperson (navn): _____

Student(er) (signatur): _____ dato _____

_____ dato _____

_____ dato _____

_____ dato _____

Oppdragsgiver (signatur):_____ dato _____

Signert avtale leveres digitalt i Blackboard, rom for bacheloroppgaven.
Godkjennes digitalt av instituttleder/faggruppeleder.

Om papirversjon med signatur er ønskelig, må papirversjon leveres til instituttet i tillegg.
Plass for evt sign:

Instituttleder/faggruppeleder (signatur): _____ dato _____

C Experiment questionnaire

Questionaire for VR-Test

How much previous experience do you have with VR (Oculus, Vive or similar)

- [ ] zero or very little experience
- [ ] 1-4 hours
- [ ] 5 hours or more

Room number 1

Do you feel that the room is big enough to comfortably move around in?

- [ ] Strongly agree
- [ ] Somewhat agree
- [ ] No opinion
- [ ] Somewhat disagree
- [ ] Strongly disagree

How many players do you feel would comfortably fit in the room?

- [ ] 1
- [ ] 2
- [ ] 3
- [ ] 4
- [ ] 5 or more

Room number 2

Do you feel that the room is big enough to comfortably move around in?

- [ ] Strongly agree
- [ ] Somewhat agree
- [ ] No opinion
- [ ] Somewhat disagree
- [ ] Strongly disagree

How many players do you feel would comfortably fit in the room?

- [ ] 1
- [ ] 2
- [ ] 3
- [ ] 4
- [ ] 5 or more

Room number 3

Do you feel that the room is big enough to comfortably move around in?

- [ ] Strongly agree
- [ ] Somewhat agree
- [ ] No opinion
- [ ] Somewhat disagree
- [ ] Strongly disagree

How many players do you feel would comfortably fit in the room?

- [ ] 1
- [ ] 2
- [ ] 3
- [ ] 4
- [ ] 5 or more

Room number 4

Do you feel that the room is big enough to comfortably move around in?

- [ ] Strongly agree
- [ ] Somewhat agree
- [ ] No opinion
- [ ] Somewhat disagree
- [ ] Strongly disagree

How many players do you feel would comfortably fit in the room?

- [ ] 1
- [ ] 2
- [ ] 3

- [ ] 4
- [ ] 5 or more

Room number 5

Do you feel that the room is big enough to comfortably move around in?

- [ ] Strongly agree
- [ ] Somewhat agree
- [ ] No opinion
- [ ] Somewhat disagree
- [ ] Strongly disagree

How many players do you feel would comfortably fit in the room?

- [ ] 1
- [ ] 2
- [ ] 3
- [ ] 4
- [ ] 5 or more

Any comments on the experiment or anything you would like to share

D Results of the experiment

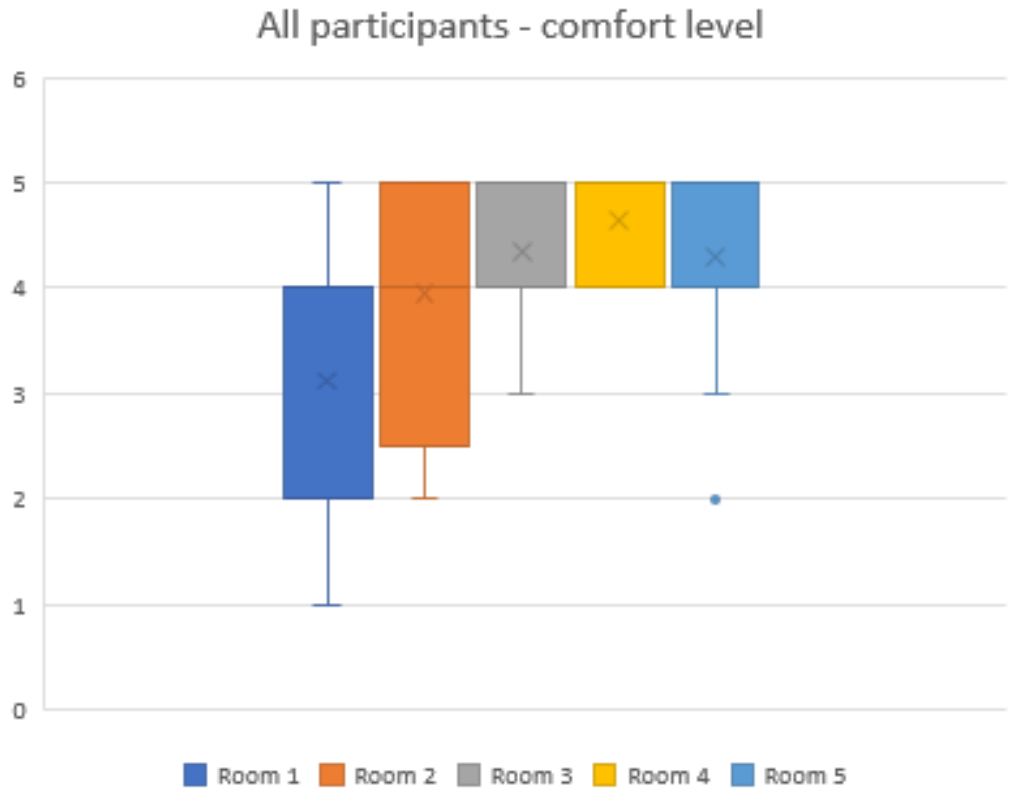


Figure 16: Comfort level, all results

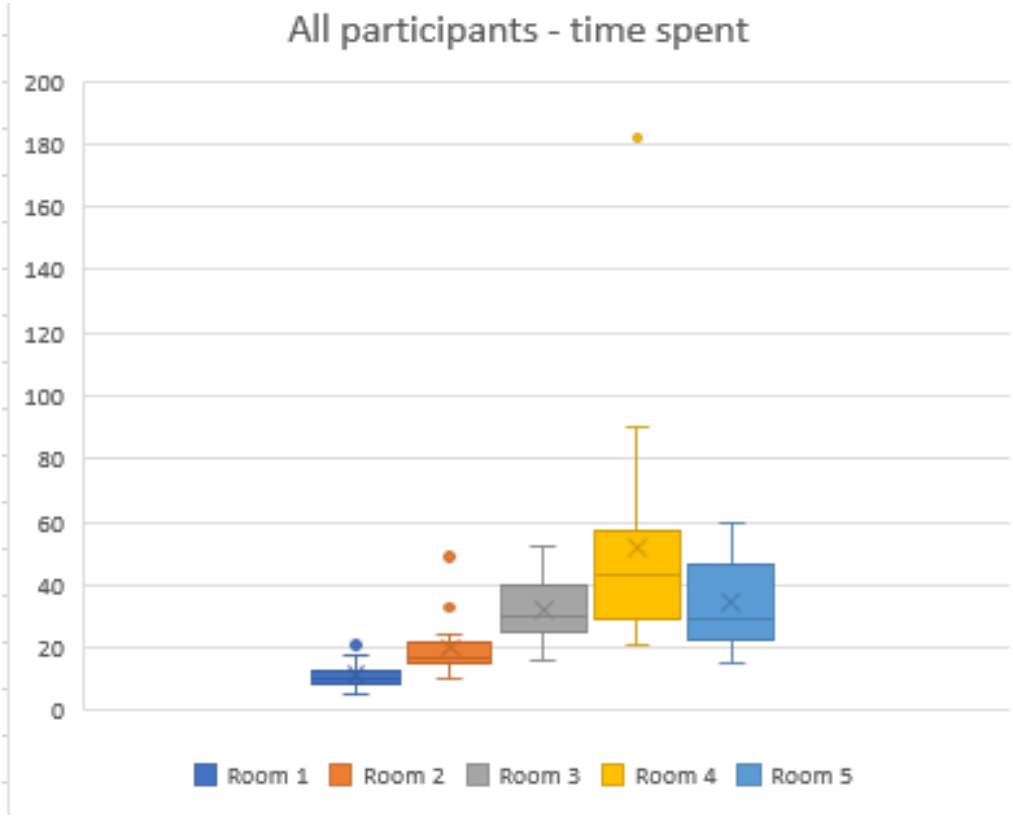


Figure 17: Time spent, all results

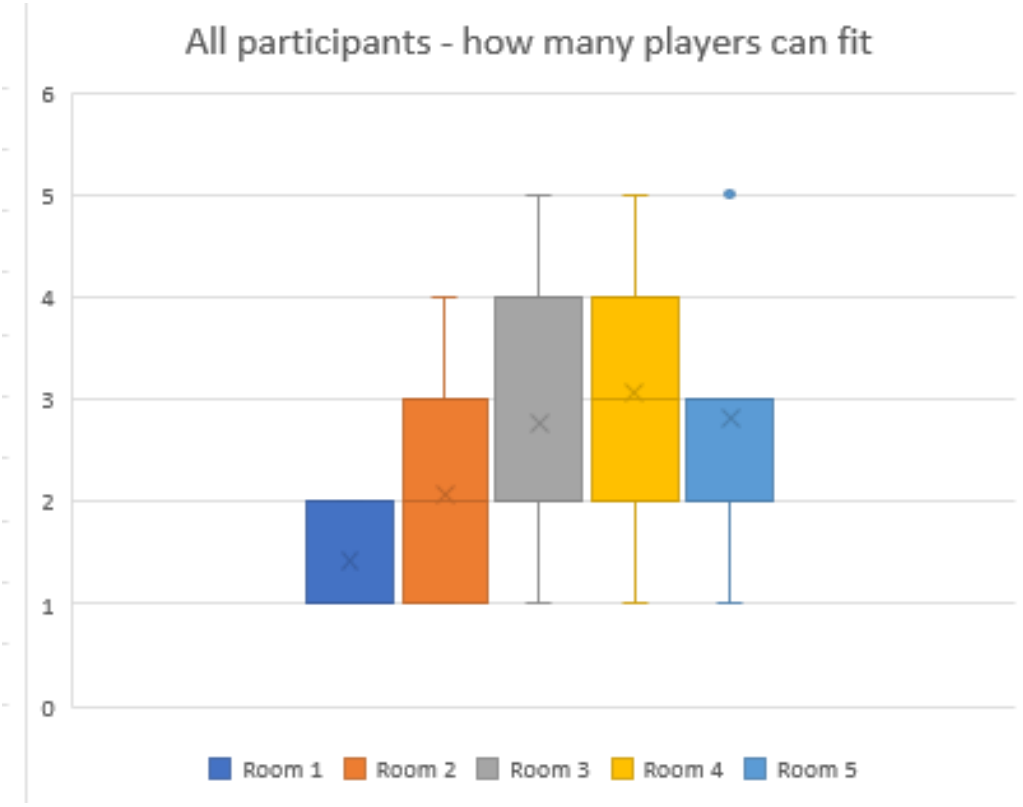


Figure 18: Player capacity, all results

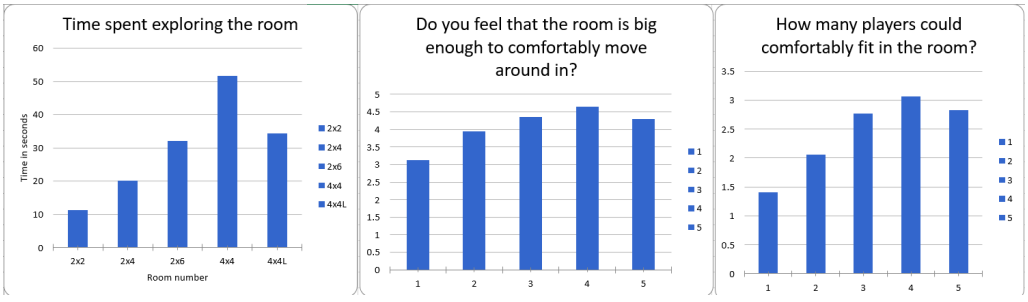


Figure 19: Average values

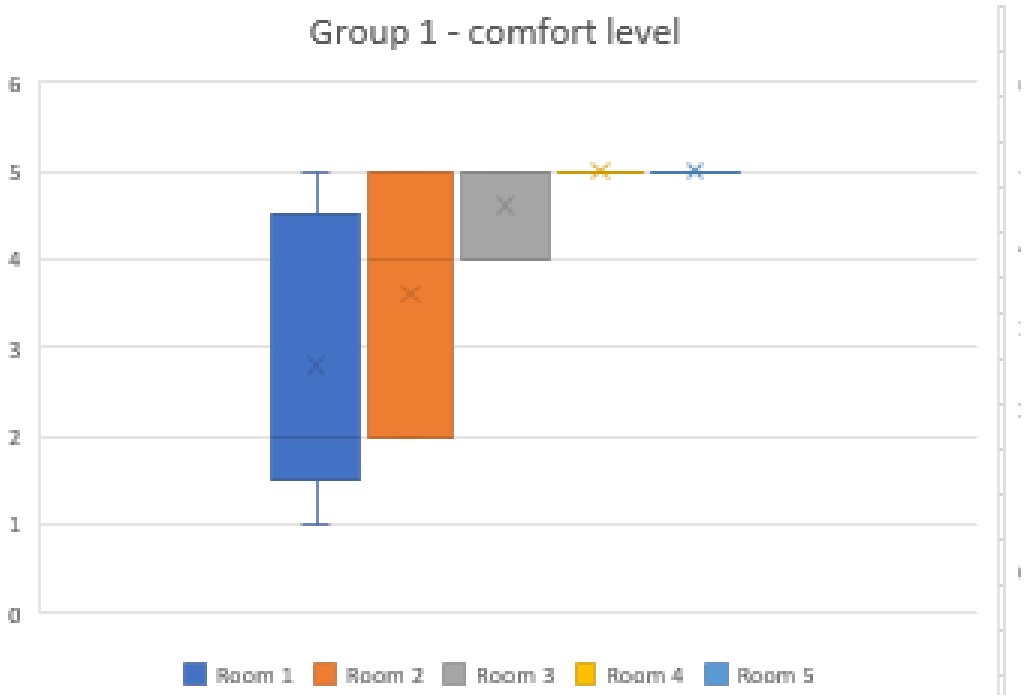


Figure 20: Comfort levels, no prior experience

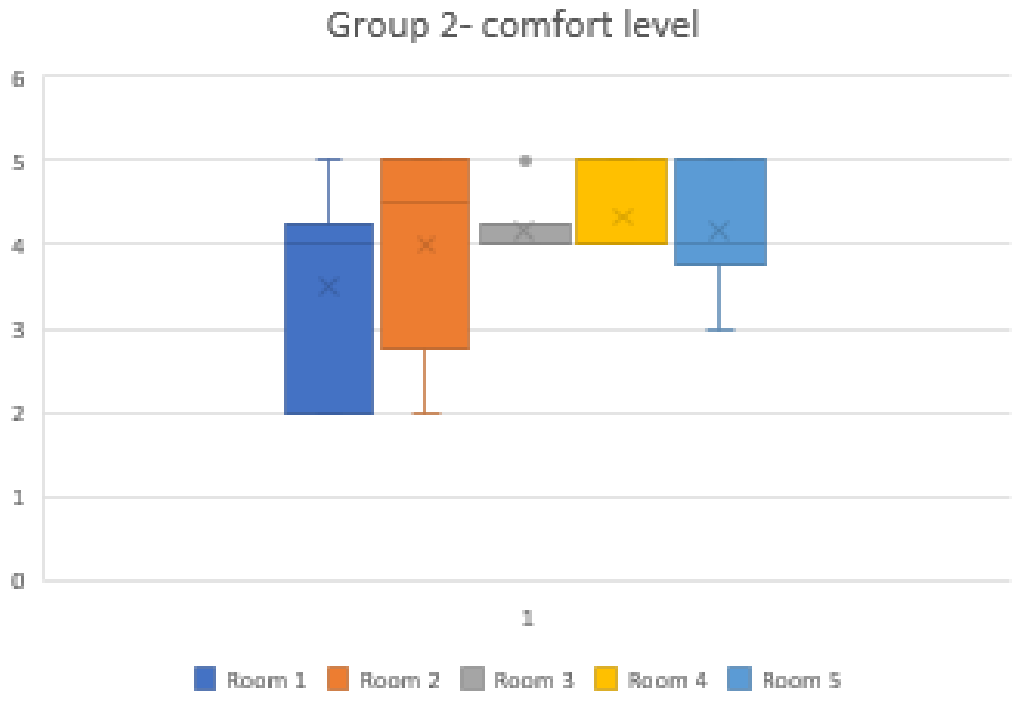


Figure 21: Comfort levels, 1-4 hours prior experience

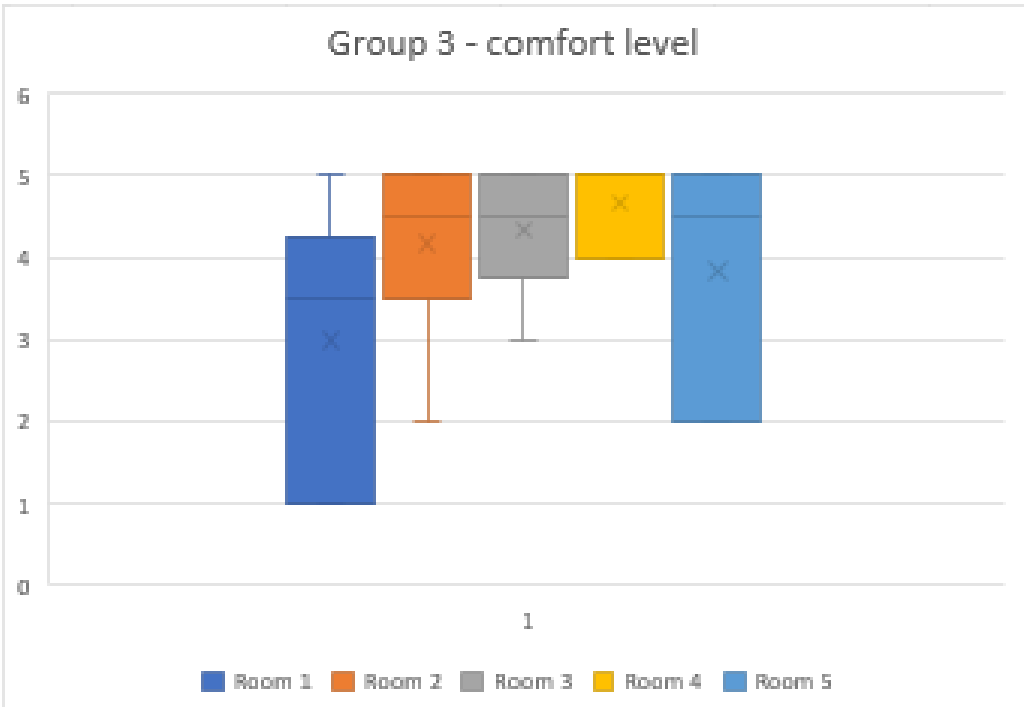


Figure 22: Comfort levels, 5+ hours prior experience

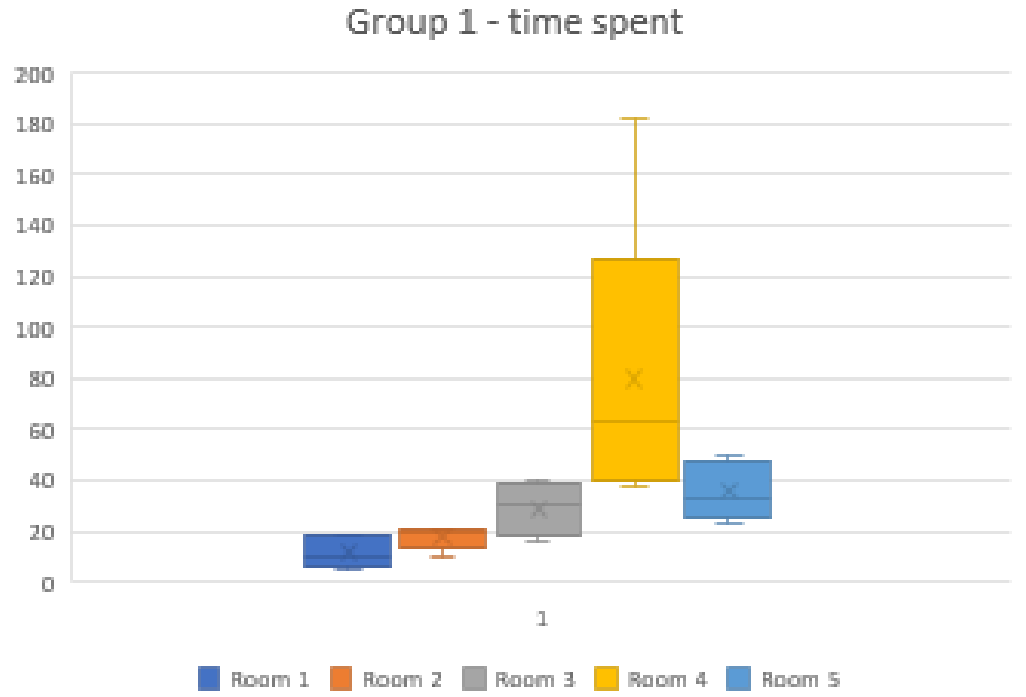


Figure 23: Time spent, no prior experience

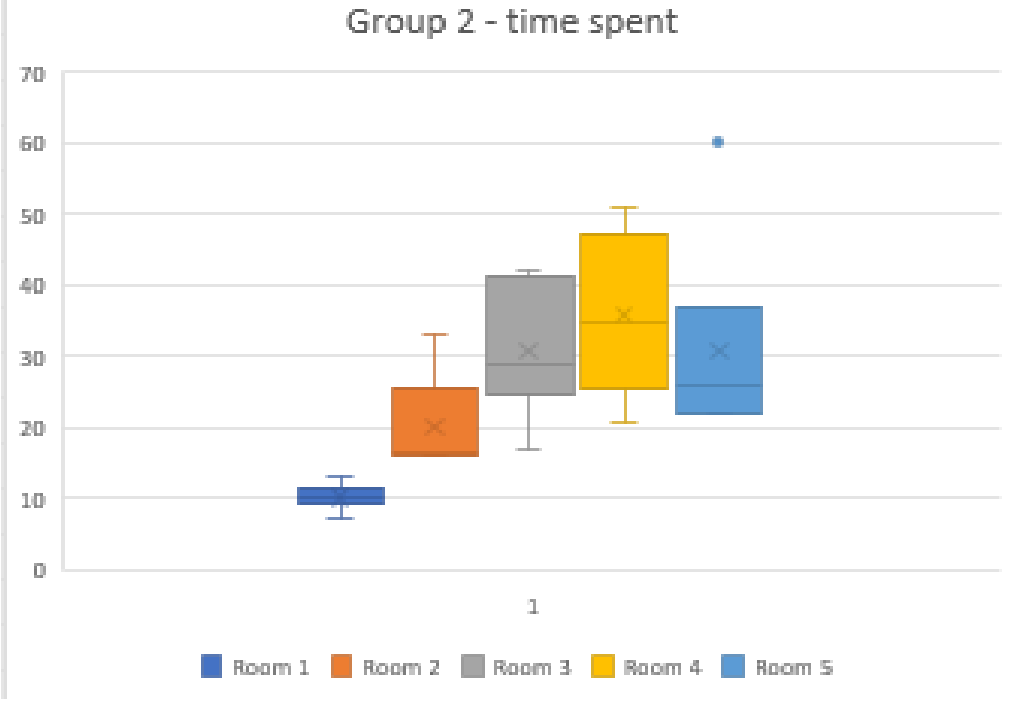


Figure 24: Time spent, 1-4 hours prior experience

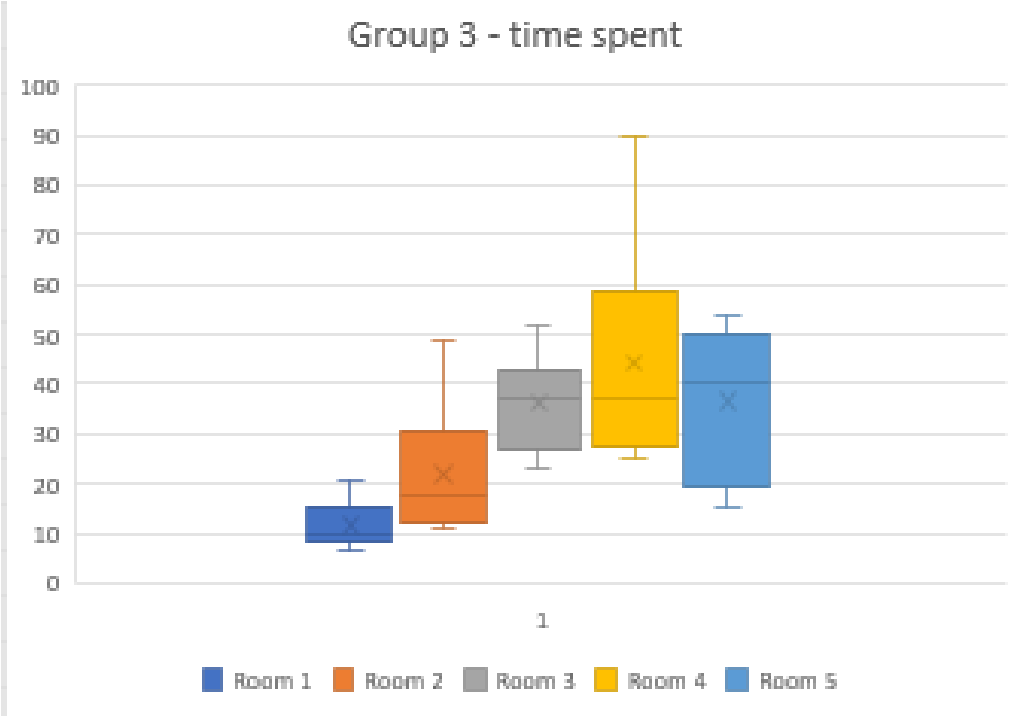


Figure 25: Time spent, 5+ hours prior experience

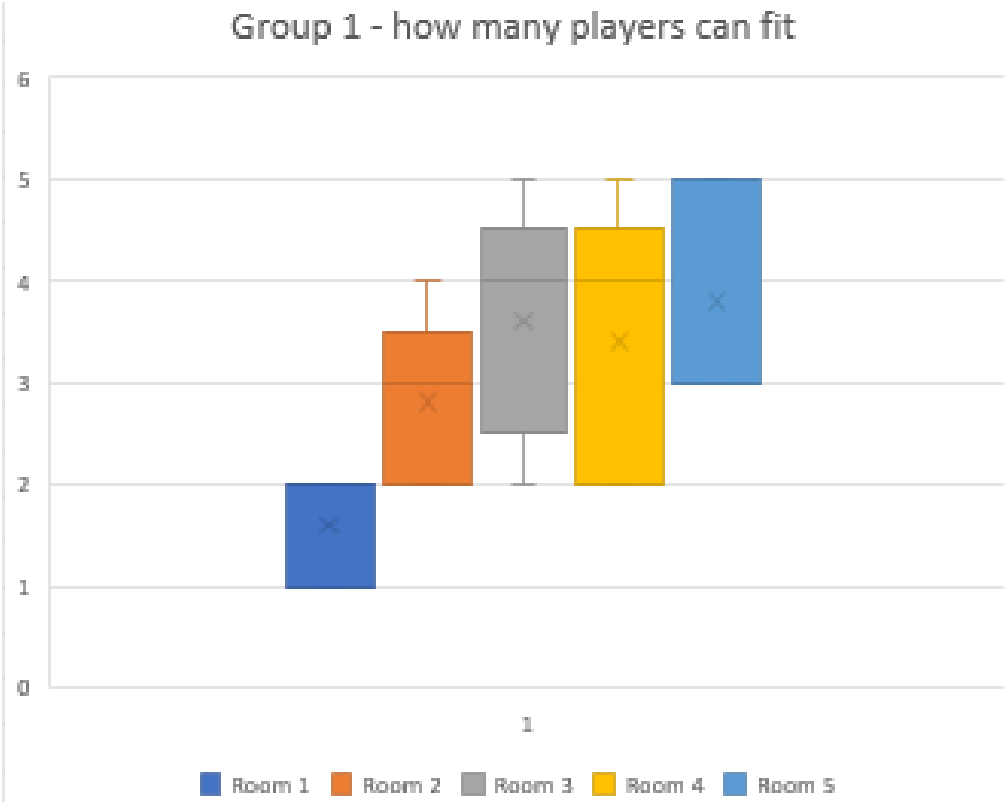


Figure 26: How many players could fit in the room, no prior experience

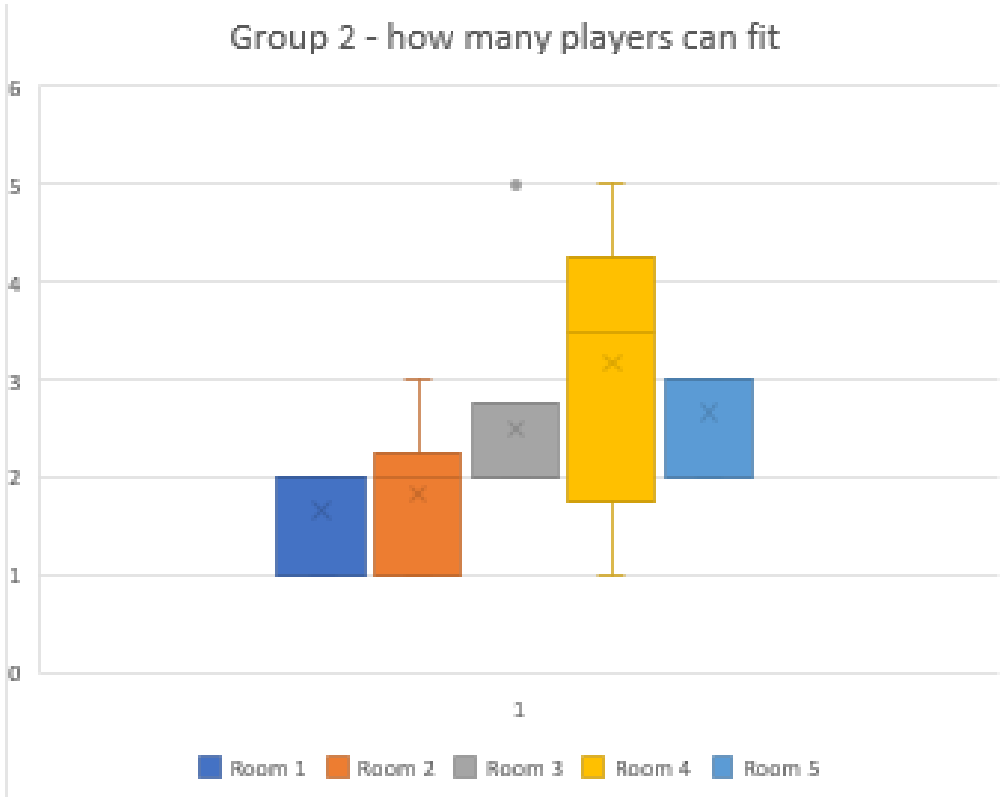


Figure 27: How many players could fit in the room, 1-4 hours prior experience

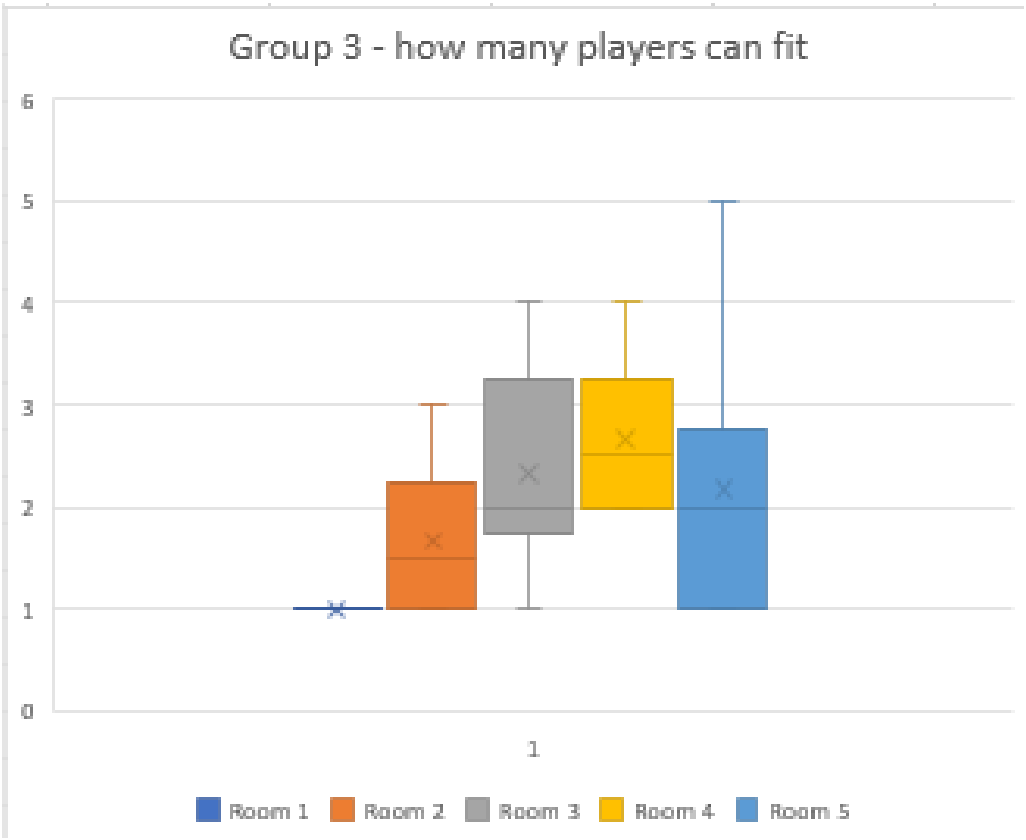


Figure 28: How many players could fit in the room, 5+ hours prior experience

E Rooms from the experiment about room size

Figure 29: The one module room.

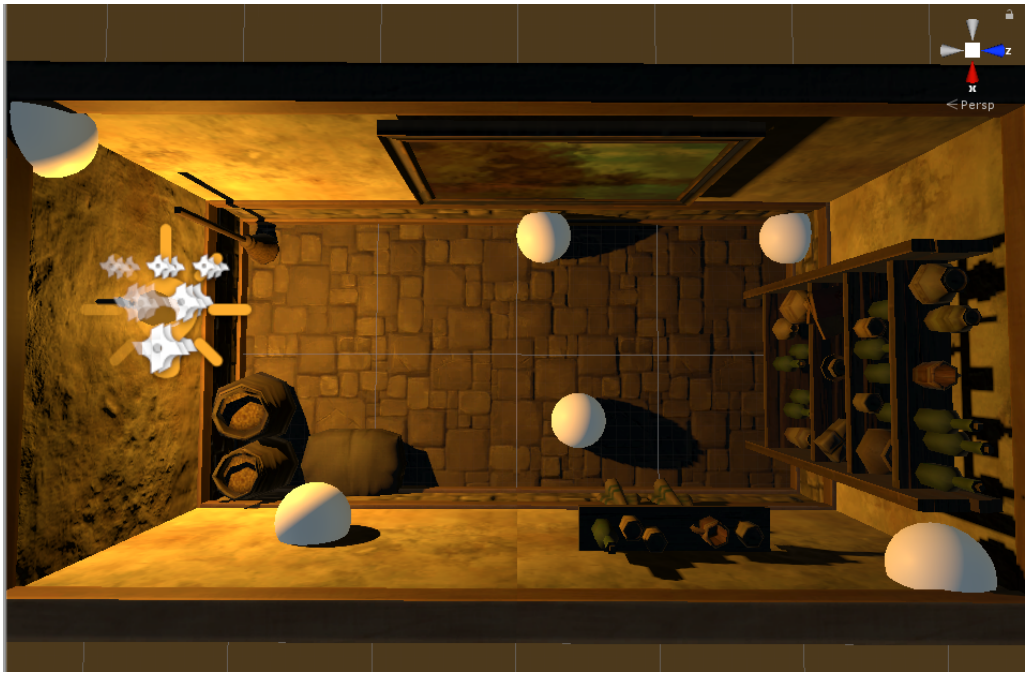


Figure 30: The two module room.

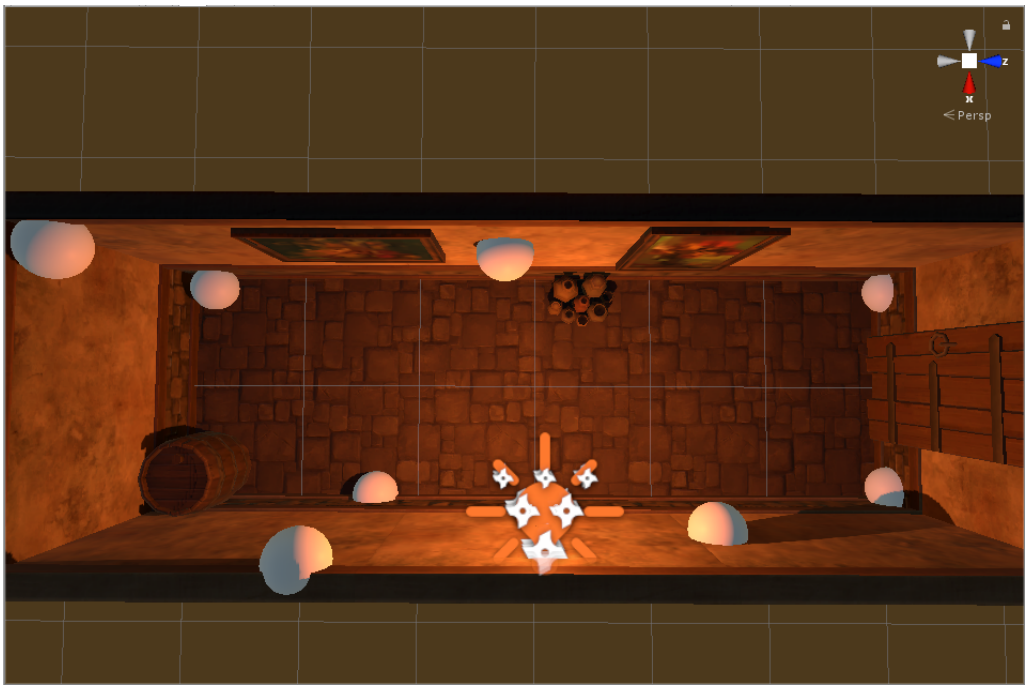


Figure 31: The three modules in a corridor room.

Figure 32: The four modules in a square room.

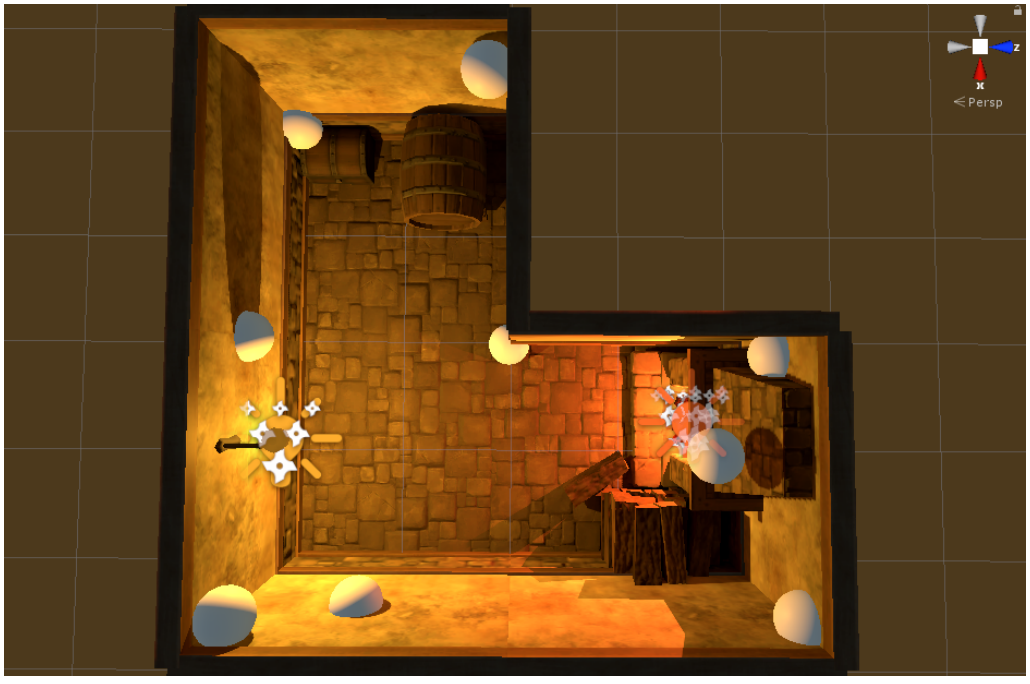


Figure 33: The three modules in an L shape room.

F Meeting Logs

F.1 Supervisor meetings

04.10.2018 - Bachelor Information Meeting

Discussion of the process and setup of the thesis. Deadlines for submission of documentation. Introduction to the process and the sessions to help with writing the thesis....

16.01.2019

Met with supervisor to discuss the project. Actions:

1. Roles in the group.
2. Language for thesis.
3. Discussing scope with Project Owner.
4. Risk management for the project.

06.03.2019

Met with supervisor to discuss the project. Actions:

1. Planned experiment on room sizes / Figured out what we needed to do for our planned experiment.
2. Figured out whether we needed to submit for GDPR approval.

27.03.2019

Met with supervisor to discuss the project. Actions:

1. Discussed how the experiment had gone and how to process the data we had collected.

07.05.2019

Met with supervisor to talk about the report. Actions:

1. Editing the bibliography.
2. Framing of the thesis.
3. How our experiment differs from testing in the usual sense.
4. The chapters of the thesis, what goes where and some in depth about these chapters specifically:
    - Technical design vs Implementation
    - Development process
    - Conclusion
    - Contents of the appendix.
5. Good habits for writing (e.g. write down notes of anything you realise while writing something else. That way you don't forget it and can flesh it out later).

F.2 Sprint meetings

18.02.2019

Actions:

- Continue using the old version of SteamVR for now. Isolate steamvr code as much as possible
- What can we do better:
    - Kind of hard to do stuff (room scaling)
    - Kind of hard to visualize how it is supposed to end up -> bigger picture before start working
- In regards to space:
    - Some of the activities require more space than a single room provide
    - Teleportation between multiple “rooms”
    - 2m still not very far to throw something
    - Able to see/throw over the “boundary” between rooms but not walk
    - Hard to stuff many people in a 2x2 room. Easier with more rooms
    - Social spaces vs rec-room
    - Tiles are not supposed to scale
    - Slots on walls where you can drop objects (i.e. shelf, cabinet)
    - When you reach the wall it has a shelf/box that’s like a bottomless bag
    - Boundaries themselves have the content
    - Kitchen can be white-boxed
- Next Sprint:
    - Nested prefab tiles
    - Cabinets with slots to interact with things
    - Attach images to the tasks

04.03.2019

Actions:

- How things have been / are going:
    - Play spaces:
        - Tool to generate play spaces
        - Generated the minimum and maximum play spaces
        - Started to fill in some of them.
        - They feel small. Rooms may feel smaller due to lack of peripheral vision
        - The smallest one is too small, not room for two people
    - Movement and networking
        - Teleport in play spaces relative to position. Teleport points (maintain offset)
        - Experiment on this? (movement)
        - Make a gif of this? (gyazo is a good tool for this. 8 second clips without paying)
    - Yet to finish:

- Interactable objects
- Different movement modes
- Nested prefabs (i.e. tiles with furniture in them)
- Doors

26.03.2019

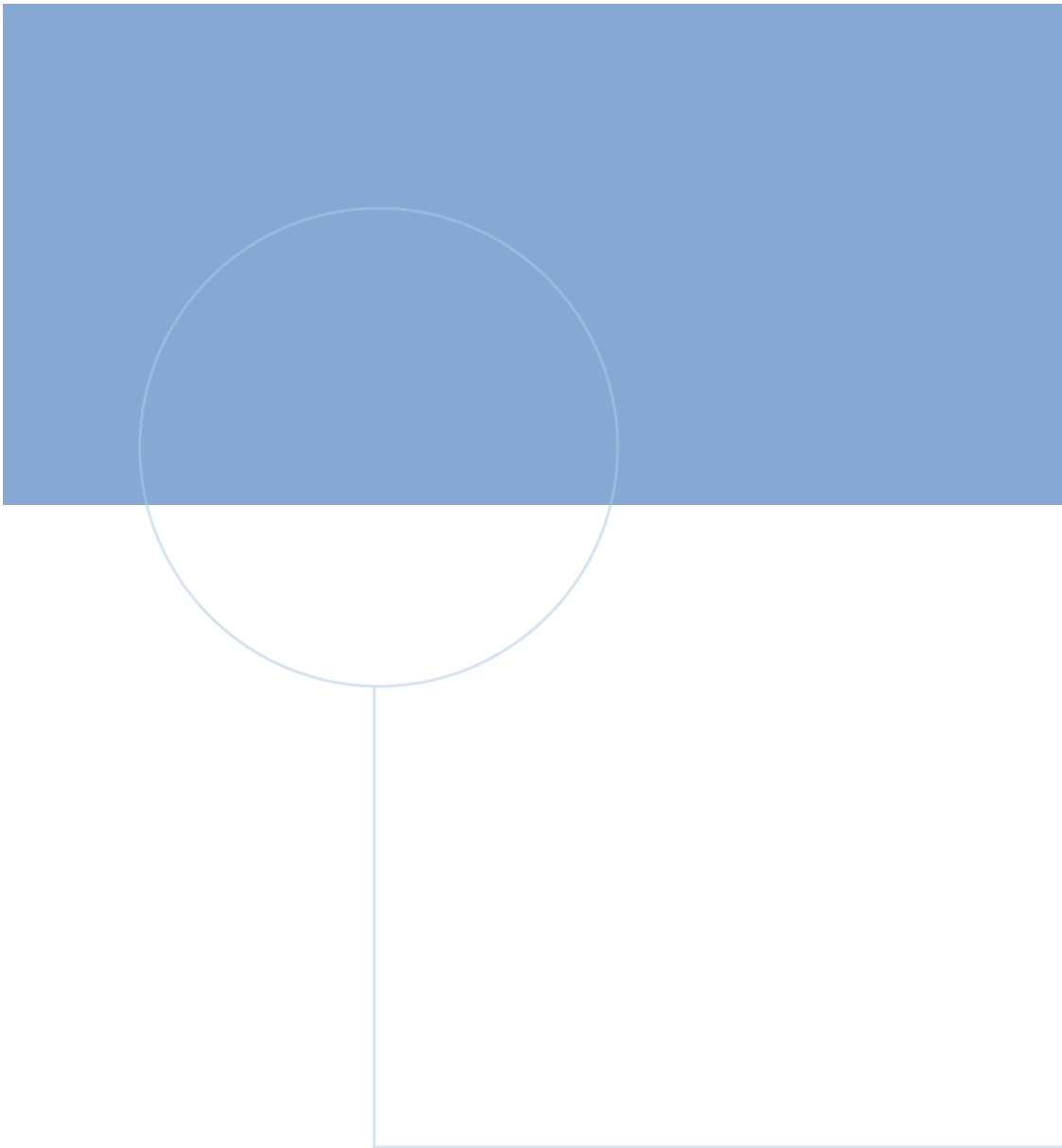
Actions:

- Rooms too small for 4 people
- 2x2m with interactables in the wall is still crazy tiny
- Following sprint:
  - Multiplayer not ready yet. After this friday.
  - Start board games
  - Do as much logic and games as possible and do the networking when we visit in Hamar.
  - Have a look at how other have done board games in VR
  - Card and darts

15.04.2019

Actions:

- Status update:
  - Darts - working, might make dart board keep track of points + synchronization
  - Checkers - working, the board does not enforce the rules on its own.
- Last week - what to do?
  - Research quick chat system in vr, add it to the report, from now till the 21st
  - Networking implementation from 22nd to 28th



NTNU

Kunnskap for en bedre verden