

An Augmented Reality Game to Support the Ski-Learning Process

BACHELOR THESIS

Luciënne de With
s1596349
Creative Technology
University of Twente

Supervisors
Dr. J. Zwiers
Dr. D. Reidsma

June 28th, 2017

Abstract

Within this research an augmented reality game that supports the ski-learning process has been designed. The game that was designed throughout this research can serve as a support tool during skiing lessons on a revolving ski slope and improves the user experience of the skiers. By making the choice for augmented reality instead of virtual reality, the user's safety while playing the game is guaranteed and cyber sickness is prevented.

Summary

With this thesis it is investigated how a game in augmented reality that supports the ski-learning process can be designed. A state of the art literature research on games that use augmented reality to teach or train people in sports was done. Six augmented reality games that teach or train people in sports were found. Furthermore, several reliable methods to provide feedback, to motivate the players, and to cause a learning effect for players were found, which served as a basis for the further steps in this research. Effective ways to provide feedback include summary feedback, intermediate feedback, multimodal feedback, and the use of assessment games. People get motivated to play and learn in games by the entertaining factor of games, multiplayer games, classic game elements such as rankings and a clear goal in the game, personalization of the game, a human-like character as trainer or coach, and rewards. Furthermore, a learning effect can be achieved within games by providing the player with clear tasks and explanations, by building upon the player's prior knowledge, and by decreasing the guidance offered in the game. During the ideation phase of this project, a total of twenty-four ideas were found for the possible implementation of a game in augmented reality that supports the ski-learning process. In the specification phase, these ideas were brought back to one final idea for the implementation, which was the following: *a multiplayer game that can be played on a revolving ski slope while wearing a head-mounted display, in which obstacles need to be avoided to prevent losing points, and gates need to be skied through in order to gain points*. This specified product idea was implemented into an application that runs on the Microsoft HoloLens. User tests were executed in order to investigate how the users perceived the game. Based on the user tests, it can be concluded that the product that resulted from this project is seen as enjoyable, interesting, something to put effort in, important, and suitable for ski-learning purposes.

Acknowledgements

This project would not have been what it is today without the help of others. That is why there are some people who I would like to thank for the help and dedication that they have put into this project.

First of all, I would like to thank my supervisor from the University of Twente, Job Zwiers, for his guidance, supervision, feedback, and his relevant suggestions for possible improvements that could be made to this project. I would also like to thank my critical observer from the University of Twente, Dennis Reidsma, for providing me with new insights regarding the possible directions of this project.

Furthermore, I want to thank the client of this project, Michiel Groot-Koerkamp, for initiating this project, for offering me the opportunity to work on it, and for providing a portable revolving ski slope that could be used during the user tests. Along with Michiel Groot-Koerkamp, I would like to offer particular thanks to André de Brouwer, Richard Bults, and Job Zwiers for the dedication and effort they put into facilitating the option to place the revolving ski slope at the terrain of the University of Twente, and for offering me the opportunity to use the revolving ski slope during the user tests of this project.

Moreover, I want to thank all the people who participated in the user tests and who shared their opinions about the realised prototype with me. My special thanks go to Alfred de Vries, Henk Waaijer, and Sander Baks for supervising my experiments and for assisting me alongside the revolving ski slope when needed.

Finally, I would like to thank my family and friends, for their support during this project and for searching their attics and garages while it was 30°C outside to be able to provide me with skiing boots.

Table of Contents

- Abstract 2
- Summary 3
- Acknowledgements 4
- Table of Contents 5
- List of Figures and Tables 7

- Chapter 1 – Introduction 9
 - 1.1 Problem statement 9
 - 1.2 Research questions 9
 - 1.3 Outline 9

- Chapter 2 – Methods and Techniques 10
 - 2.1 Creative Technology Design Process 10
 - 2.2 Methods 12

- Chapter 3 – State of the Art on Augmented Reality Games that Teach People in Sports 17
 - 3.1 Related work 17
 - 3.2 Motivational aspects 18
 - 3.3 Feedback 20
 - 3.4 Learning effect 20
 - 3.5 Cyber sickness 21
 - 3.6 Conclusion 23

- Chapter 4 – Ideation and Exploration 24
 - 4.1 First Ideation 24
 - 4.2 Stakeholder Analysis 27
 - 4.3 Early Design Choices 27
 - 4.4 Use Cases 28
 - 4.5 First Brainstorm 30
 - 4.6 Brainstorm and Evaluation with Client 36
 - 4.7 Product Idea 37

- Chapter 5 – Product Specification 39
 - 5.1 Early Prototypes 39

5.2 Requirements analysis	44
5.3 Game Design	46
Chapter 6 – Product Realisation	49
6.1 System Architecture	49
6.2 Implementation	50
Chapter 7 – Evaluation	63
7.1 Functional evaluation	63
7.2 User evaluation	65
7.2.1 First Round of User Tests	65
7.2.2 Second Round of User Tests	72
7.3 Conclusion and Discussion of Test Results	76
Chapter 8 – Discussion and Conclusion	79
8.1 Conclusions	79
8.2 Discussion	80
Chapter 9 – Future Work	83
Appendix	85
Appendix A. Code Markerless Augmented Reality Prototype	85
Appendix B. Realisation Timeline	87
Appendix C. Spawner Script	89
Appendix D. Camera Collider Script	93
Appendix E. Player Manager Script	95
Appendix F. Canvas Position Script	97
Appendix G. Floating Script	98
Appendix H. Game Manager Script	99
Appendix I. Network Manager Script	101
Appendix J. Obstacle Script	102
Appendix K. Ski Rotation Script	103
Appendix L. Questionnaire First Round of User Tests	104
Appendix M. Questionnaire Second Round of User Tests	108
References	111

List of Figures and Tables

Figures		
Figure	Description	Page
2.1	Creative Technology Design Process.	10
4.1	A revolving ski slope.	24
4.2	The first ideation phase.	26
4.3	The ideas that were found for skiing and earning points in the game.	31
4.4	The ideas that were found for the presence of a fellow player in the game.	33
4.5	The ideas that were found for providing feedback in the game.	34
4.6	The ideas that were found for implementing levels in the game.	35
5.1	An example of a Vuforia application that is very similar to the Vuforia prototype that was made.	43
5.2	An application that is very similar to the markerless augmented reality prototype that was made.	43
5.3	The multiplayer prototype, showing the game being played with only the host and with the host and client.	44
5.4	The elemental tetrad, containing the four basic elements that form a game.	47
6.1	Simple system architecture of the skiing game.	49
6.2	An augmented reality application on a smartphone.	51
6.3	The three-dimensional objects created in the Maya software and used in the game as gates and obstacles.	52
6.4	The situation on the revolving ski slope with the corresponding axes.	52
6.5	Placement of the game objects along the ski slope.	54
6.6	Placement of the game objects along the slope and the associated calculations.	55
6.7	The client-server model.	55
6.8	The parent-child relationship for the parent of the Unity camera and the Unity camera.	58
6.9	A schematic overview of the communication between the scripts of the project.	60
6.10	The game from the perspective of the player.	61
6.11A	Placement of the player and the game objects while the game is being played in the Unity editor.	61
6.11B	Placement of the player and the game objects while the game is being played in the Unity editor.	62
7.1	Pie charts containing the characteristics of the test participants of the first round of user tests.	65
7.2	Means and standard deviation of participants' perceived cyber sickness symptoms.	67
7.3	Overall scores for the questions about intrinsic motivation, divided in the categories effort/importance, perceived competence, and interest/enjoyment.	68
7.4	Average IMI scores per category (interest/enjoyment, perceived competence, effort/importance) over the whole population.	69
7.5	Average IMI scores per category (interest/enjoyment, perceived competence, effort/importance) per level of experience.	70
7.6	Mean scores of the multiplayer statements in the test.	71
7.7	Mean scores of the multiplayer statements in the test for people who noticed the fellow player.	71
7.8	Participants' appreciation of the skiing game as a learning tool.	72

7.9	The updated appearance of the obstacle game object.	73
7.10	Pie charts containing the characteristics of the test participants of the first round of user tests.	74
7.11	Means and standard deviation of participants' perceived cyber sickness symptoms.	75
7.12	Mean scores for statements related to feedback, given during the second round of user tests.	76

Tables

Table	Description	Page
4.1	The possible sub-directions for direction number one and direction number three.	25
4.2	Overview of the stakeholders per category for the skiing game.	27
4.3	The ideas for skiing and earning points in the game and their explanations.	31
4.4	The ideas for the presence of a fellow player in the game and their explanations.	33
4.5	The ideas for providing feedback in the game and their explanations.	34
4.6	The ideas for implementing levels in the game and their explanations.	35
4.7	The results of the brainstorm session with the client.	36
4.8	Evaluation of the ideas that were generated through the brainstorm sessions.	37
5.1	A use scenario for the skiing game with the paper prototype that was made.	39
5.2	Product requirements and prioritization for the skiing game.	45
6.1	Overview of the scripts and their functionality.	59
7.1	Evaluation of the product requirements.	63

Chapter 1 – Introduction

1.1 Problem statement

Taking skiing classes on a ski-slope in The Netherlands does not necessarily provide the user with a feeling that is related to the experience of being in the snowy mountains. Yet people go skiing to get the “*real*” skiing experience, which is lacking at the moment. Therefore the idea arose of combining skiing on an artificial ski-slope with augmented reality. To enable the user to learn something from the experience and to add meaning to it, it was decided to make the envisioned product a serious game that can be used to support the ski-learning process. Overall, games are seen as entertaining and therefore motivational. However, serious games do not have a primary focus on entertainment but on education instead [1], [2], [3], which makes them a perfect means for training people in the field of sports. Since games are seen as entertaining, people experience learning through serious games as entertaining as well, which makes them more excited to learn in serious games [4], [5]. Augmented reality, which presents the user to an environment where normally present surroundings are overlaid with virtual three-dimensional objects [6], [7], is seen as a suitable technology for serious games, as it allows for natural interactions between the player and the game.

1.2 Research questions

In this thesis it is investigated how a game that uses augmented reality technology to support the ski-learning process can be designed. An important aspect when designing such a game is the question what the added value of a game in augmented reality for skiing classes on a revolving ski slope is. Another important aspect that was investigated is how people perceive a game in augmented reality that is meant to support the ski-learning process. Additionally, it is researched if the augmented reality skiing game that was designed causes cyber sickness symptoms for its players. These three aspects were investigated in order to answer the main research question of this thesis.

1.3 Outline

In Chapter 2 a description of the used methods and techniques for this thesis will be given. This will be followed by the results of a state of the art research on serious games that use augmented reality to teach or train people in sports in Chapter 3. Subsequently, Chapter 4 describes the ideation and exploration of the possible design choices for the skiing game. Chapter 5 gives an overview of the product specifications of the skiing game, followed by a description of the product realisation in Chapter 6. This will be followed by an evaluation of the user tests that were carried out in Chapter 7. In Chapter 8, there will be a discussion of this research and conclusions will be drawn upon how a game that uses augmented reality technology to support the ski-learning process can be designed. Finally, in Chapter 9 recommendations for further research and future work will be given.

Chapter 2 – Methods and Techniques

In this chapter the methods and techniques that were used in this research to eventually answer the research questions that were mentioned in Chapter 1 will be explained. The chapter starts with an explanation of the design process that was followed throughout the project, which is the Creative Technology Design Process. In order to properly execute the phases of the Creative Technology Design Process, a number of methods were used. The used methods are listed and explained in this chapter.

2.1 Creative Technology Design Process

The overall process of this graduation project was carried out following the Creative Technology Design Process by Mader and Eggink [8]. The Design Process of Creative Technology consists of four phases: Ideation, Specification, Realisation, and Evaluation. Figure 2.1 gives a schematic representation of this design process.

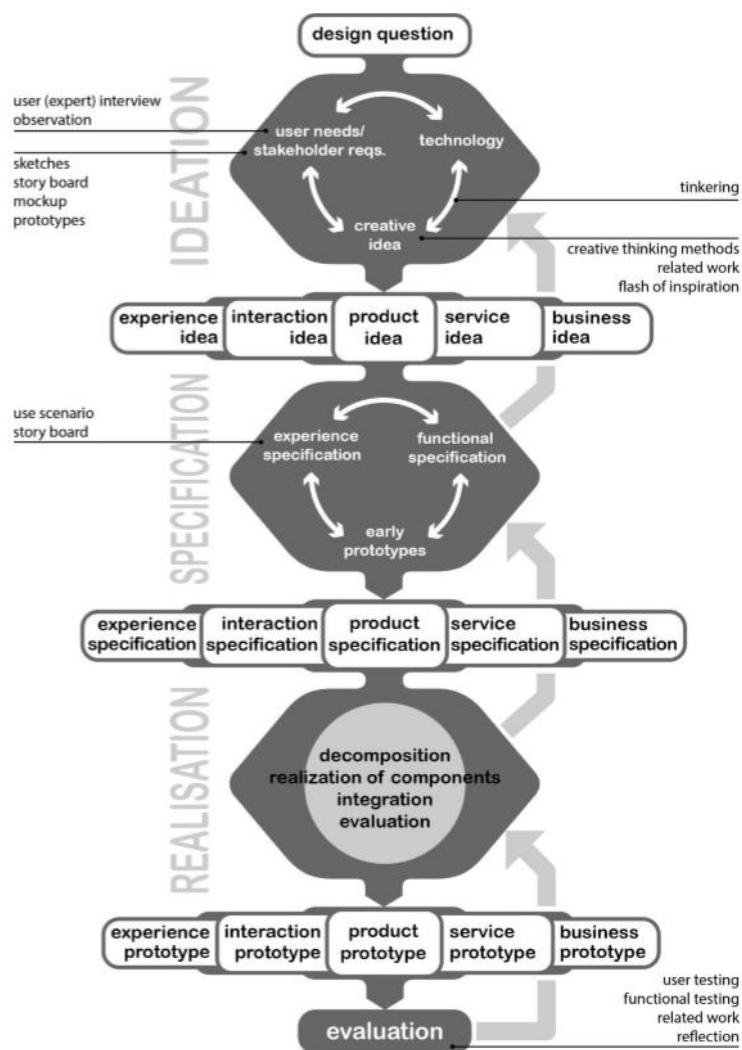


Figure 2.1: Creative Technology Design Process

Every phase in the Creative Technology Design Process has two separate phases of its own, a divergence phase and a convergence phase. Every phase (Ideation, Specification, Realisation, Evaluation) starts with a divergence phase, where the design space is opened and different possible solutions can be explored. The divergence phase is followed by the convergence phase, where the design space is reduced again to one or few solutions.

2.1.1 Ideation

The ideation phase usually starts with a design question, an assignment from a client, or a creative idea. In this case, the ideation phase started with the assignment that was given by the client of this project, which was to make a combination between a revolving ski slope and augmented reality or virtual reality. In the Creative Technology Design Process, technology can be the starting point for the ideation phase. That is why tinkering, finding new functions or utilizations for existing technologies, is an important part of the ideation phase.

Related work is often used as an inspiration during the ideation phase. Therefore Chapter 3, section 3.1, gives an overview of related work. This overview contains games that use augmented reality to teach or train people in sports, since the search for games that use augmented reality to teach or train people in skiing did not deliver results. The games found under related work were reviewed, and useful techniques that were used in these games were taken into consideration in the ideation phase.

It is important to know whom the final design of the skiing game is targeted at. That is why in the ideation phase a stakeholder analysis was executed, to find out who the possible end users of the skiing game are. Once it was clear who the end users were, several brainstorm sessions were held and multiple use cases were made that entailed a variety of design possibilities. Out of the different generated solutions, one product idea was chosen that was taken to the specification phase.

2.1.2 Specification

The product idea that resulted from the ideation phase served as a starting point for the specification phase. The goal of the specification phase was to decide on the functional specifications of the skiing game. Specification was done by making several prototypes and evaluating their functionalities. For this purpose, a paper prototype was made of what the researcher envisioned the final game to be like. Besides the paper prototype, several prototypes were made to test the possibilities and functionalities of the software that was chosen to use for making the skiing game. After creating and evaluating the prototypes, a list with functional requirements of the final prototype of the skiing game was made. The MoSCoW method, which is explained in section 2.2.5, was used to prioritize between the requirements.

2.1.3 Realisation

The functional requirements that were defined in the specification phase are the basis for the realisation phase. The goal of the realisation phase is to make a working prototype that satisfies the functional requirements as were set.

2.1.4 Evaluation

The final prototype that resulted from the realisation phase was evaluated in the evaluation phase. The evaluation consisted of two parts, functional evaluation and user evaluation. Functional evaluation was used to determine if the resulting prototype fulfils the functional requirements that were set in the specification phase. This part of the evaluation was carried out by the researcher. User evaluation consisted of test sessions with end users, where users got the chance to share their opinions about the final prototype. User evaluation was done to determine if the final prototype satisfies the needs and desires of the end user.

2.2 Methods

A number of methods were used throughout the process of this project. The used methods are listed and explained in this section.

2.2.1 Research

Literature research was done in the field of serious games that use augmented reality to teach or train people in sports. The results of this research can be found in chapter 3 and serve as background knowledge to this project. First of all, a state of the art research was done about existing games that use augmented reality to teach or train people in sports. The research was elaborated by the techniques that were used in these games to teach people, give them feedback and motivate them. Also, research was done about cyber sickness, especially about its causes and ways to decrease or prevent its symptoms, since cyber sickness is a well-known problem in applications that present the user to virtual environments. The result of the literature research served as a starting point and as background knowledge for the ideation phase of this project.

2.2.2 Stakeholder analysis

An analysis of stakeholders was executed to identify the users that the skiing game was designed for. Stakeholders are the people who will or can be affected by the product. The stakeholders of the skiing game were identified using the methodology of Sharp et al. [9]. Sharp et al. identified four groups of baseline stakeholders, the stakeholders who are most directly influenced by the product and have the most influence on the product. The categories of baseline stakeholders are users, developers, legislators, and decision-makers.

- **Users**

According to Sharp et al. [9], users are the people who interact with a product and control it directly. Eason [10] argues that users can be divided into three different groups, which are primary users, secondary users, and tertiary users. Primary users use the product the most directly and often. Secondary users are the users who use the product occasionally. Tertiary users are influenced by the product's launch on the market and can have an influence on its sales.

- **Developers**

The developers are the stakeholders who are responsible for the development of the product. Stated differently, the developers are the people who design and build the product. They have a great influence on the requirements engineering process of the product.

- **Legislators**

The legislators are the stakeholders that are capable of influencing the product by rules and

regulations. Legislators can be both people and institutions, on a local, national, or international level.

- **Decision-makers**

As the name says, the decision-makers are the people who make decisions about the product. According to Sharp et al. [9], decision-makers are present in both the developer organisation and the user organisation.

2.2.3 Brainstorm

In the ideation phase, several brainstorm sessions were held in order to generate ideas on how to design a game that supports people in the ski-learning process. There is a great number of brainstorming techniques in existence that could be used to do this. Wilson [11] distinguishes between individual brainstorming and group brainstorming. Wilson also mentions three fundamental principles that should be taken into account when having brainstorm sessions. The first fundamental principle is to aim for quantity, not quality. The goal of brainstorming is to get as many ideas as possible, which means that the successfulness of a brainstorm session can be measured by the number of ideas that was generated. The second fundamental principle of brainstorming is that the ideas of others cannot be criticized, positively or negatively, implicitly or explicitly, during the brainstorm session. The third principle is that new and wild ideas should be stimulated. New ideas can arise from already existing ideas, by combining them, stretching them, improving them, or by finding a metaphor for them. Wild ideas, which are ideas that are not directly feasible or applicable, can serve as a trigger to find suitable ideas. These three fundamental principles for brainstorming essentially mean that every idea is welcome in a brainstorm session.

Wilson [11] also mentions several brainstorming techniques for group brainstorming, which are the following:

- **Buzz Sessions**

Buzz Sessions are an effective technique for brainstorm sessions in large groups. The group is divided into smaller groups, which all get a topic to brainstorm about for a set period of time. After the set period of time, all the small groups come back to the big group and present their ideas.

- **Free Listing**

When Free Listing, all individual participants of a brainstorm session are asked to make a list of their ideas or solutions to the topic of the brainstorm in a short and predefined period of time.

- **Reverse Brainstorming**

In Reverse Brainstorming, also called Negative Brainstorming, the goal is to first find negative ideas or faults and then focus on positive ideas and solutions. The idea behind this approach is that it is often easier to find faults than it is to find solutions. The faults are used as an input to find solutions.

- **Delphi Method**

The Delphi Method is a brainstorming technique that only involves experts in the field of the topic of the brainstorm. A coordinator asks the experts for ideas on how to solve a specific problem. All the experts give their opinion, and all their opinions are criticized by the other experts. At the end, a summary of the given solutions is made and sent to all the experts. In a second round, more specific questions are asked, based on earlier results. Again, the results

are summarized and sent to all the experts. This process continues until there is a final idea that will serve as the solution.

- **Remote Brainstorming**

Remote Brainstorming is rather a communication technique than a brainstorm technique. Using this method, brainstorm sessions can be held over distance, using communication technologies. The options for brainstorming are dependent on the options that are offered by the chosen communication technology.

Within this project, two brainstorm sessions were held. The first was an individual brainstorm session by the researcher, using the Free Listing technique. The first brainstorm session served as an input to the second session, which was held by the researcher and the client of this project. Also for that brainstorm session the Free Listing technique was used, since no other techniques seemed applicable for a small group like that.

2.2.4 Use Cases

In the ideation phase, use cases were used to identify requirements that were needed for the skiing game from a user's point of view. Use cases describe expected interactions between the user and the product. A use case can be defined as a *"set of scenarios tied together by a common user goal"* [12]. In software engineering, the use case template of Cockburn is often used to construct valid use cases. Cockburn [13] describes two structures for use cases, "fully dressed" and "casual". For this project the casual structure defined by Cockburn was used, which entails the following details:

- Title of the use case, stating the goal.
- Primary actor.
- Scope, answering what problem is being solved, how the problem will be solved, and why this is an appropriate solution.
- Level, which can be "system", "internal", or "context". The system level is applicable to goals that can be reached in a single session with the system. The internal level applies to goals that are not complete. The context level is used for goals that involve other systems next to the system that the use case is about.
- Story, consisting of success scenario and extension conditions. Extension conditions are steps that could go wrong in the scenario.

2.2.4 Game Design

Because the final prototype that resulted from this project is a game, game design principles were studied. This was done by using the method of lenses by Schell [14]. Schell defines one hundred lenses, which all provide different ways of seeing and thinking about a game. Every lens requires to see the game from another perspective and possibly change thoughts about it. A selection of the lenses of Schell were used to consider the possible solutions and design choices that arose from the ideation and specification phase and to add or refine some of these ideas.

2.2.5 Requirements Analysis and Prioritization

In the specification phase the functional requirements that the skiing game should fulfil were set, based on earlier results from the ideation phase and early prototypes. Since there is a time limit to

this project, the feasibility of the requirements had to be taken into account. Also, the different requirements were prioritized, defining which requirements should be met first. The prioritization of the requirements was done following the MoSCoW method [15]. MoSCoW stands for *Must have*, *Should have*, *Could have*, and *Won't have*. All four of them have their own level of prioritization, where *Must have* is the most important and *Won't have* the least important. The requirements on the *Must have*-level are the minimal requirements that the product should fulfil. The *Should have*-requirements are not as critical to the launch of the product as the *Must have*-requirements, however they are of a high value to the user and are therefore seen as important. The *Could have*-requirements are features that are nice to include in the design of the product, but only if time and costs allow. These requirements are the first to be removed in case there is not enough time to fulfil all requirements. Finally, the *Won't have*-requirements are the requirements that were taken into consideration for the final product, but were taken out of the design plan because of limited duration of the project. The *Won't have*-requirements typically include features that could be added to a future version of the product.

2.2.6 Evaluation

After a functioning prototype of the skiing game was realised the prototype was be evaluated, which was the last phase of this research. In the evaluation phase, a functional evaluation as well as a user evaluation were executed. The functional evaluation assessed if the prototype functions as intended and if all the requirements are met. The user evaluation allowed end users to interact with the prototype and share their opinions about it. The functional evaluation had to be executed before the user evaluation could be executed, to prevent the prototype from malfunctioning during user tests.

2.2.6.1 Functional Evaluation

The functional evaluation of the final prototype of the skiing game was done by comparing the functionality of the skiing game to the requirements that were set in the specification phase, using the MoSCoW analysis. A table containing the requirements and their MoSCoW value was made. For every requirement it was assessed if the requirement was met in the prototype or not.

2.2.6.2 User Evaluation

The user evaluation was executed by organised test sessions with potential end users of the skiing game. A number of test participants had to ski on the revolving ski slope while wearing a head-mounted display that presented them to the skiing game. They could play the game for approximately three to four minutes. After testing the game, they were presented to a questionnaire that they had to fill out. The questionnaire was composed based on two validated tests. In addition to the questions from the validated tests, some extra questions were added.

The first part of the questionnaire consists of the questions from the Simulator Sickness Questionnaire by Kennedy et al. [16], also referred to as SSQ. In the SSQ a person can indicate how much certain symptoms, which are related to cyber sickness, are affecting him/her at that moment. To indicate how much a symptom is affecting the person who is answering the SSQ, the person can choose from the options *none*, *slight*, *moderate*, and *severe*. Kennedy et al. divided the different symptoms listed in the SSQ in three symptom clusters, which are Oculomotor, Disorientation, and Nausea. The symptoms that belong to the Oculomotor cluster are general discomfort, fatigue,

headache, eyestrain, difficulty focussing, difficulty concentrating, and blurred vision. The Disorientation cluster contains difficulty focussing, nausea, fullness of head, blurred vision, dizzy (eyes open), dizzy (eyes closed), and vertigo as symptoms. The symptoms that belong to the Nausea cluster are general discomfort, increased salivation, sweating, nausea, difficulty concentrating, stomach awareness, and burping. It can be noted that some of the symptoms belong to two of the symptom clusters. Based on how the test participants indicated being affected by the different symptoms it was decided if they were suffering from cyber sickness after the experiment, or not. Based on the symptoms that were rated highest it was decided what cluster of symptoms were affecting the test participants the most.

The second part of the questionnaire consists of part of the questions from the Intrinsic Motivation Inventory, also referred to as IMI, mentioned by Van Delden [17]. IMI is a validated test to measure a participants' subjective experience with an experiment. Stated differently, the results of IMI indicate what the test participants' opinions are about the prototype that is presented to them during the experiment. However, only part of the questions of IMI were chosen to be included in the questionnaire that was used in the user evaluation of the skiing game. Because of this, it can be doubted how validated the test questions are, since they are taken out of their context. The questions of IMI are divided into seven categories, which are interest/enjoyment, perceived competence, effort/importance, pressure/tension, perceived choice, value/usefulness, and relatedness. Participants could answer the questions by choosing points on a scale from one to seven, where one means "not true at all" and seven means "very true". A scale from one to seven was chosen because the standard IMI questions also use a scale that ranges from one to seven and the standard IMI test provides a way to calculate test scores based on these scales. Also, a scale from one to seven allows test participants to give their answers very detailed, as they can choose from not true at all, not true, slightly not true, neutral, slightly true, true, and very true. This way, participants are allowed to show their doubts or their certainty when they are saying a statements is true or not true, because they can also say it is slightly true/not true or very true/not true. Therefore, it is expected that a scale from one to seven delivers more reliable results than when a smaller scale would be used.

The categories which were part of the questionnaire that was presented to test participants after the experiments are interest/enjoyment, perceived competence, and effort/importance. The other categories are not related to the experience of the game and are therefore excluded, except for the questions under value/usefulness. These were taken as a starting point to formulate new questions about the use and usefulness of the skiing game in particular. A total of ten questions were added to the questionnaire, consisting of two open questions and eight questions that should be answered on a scale from one to seven. It was chosen to use a scale from one to seven again to keep the questionnaire consistent, as all other questions also used a scale from one to seven.

The scores obtained from the IMI part of the questionnaire were processed by calculating the mean score for every IMI category (interest/enjoyment, perceived competence, effort/importance) by averaging the scores obtained for the statements in the category. Every IMI category consists of one or more questions which are said to be "reversed statements", as they state something negative. According to the IMI test [18], the scores of the reversed statements can be calculated by subtracting its value from eight, and using the resulting number as the item score.

Chapter 3 – State of the Art on Augmented Reality Games that Teach People in Sports

This chapter contains the result of a state of the art research on serious games that use augmented reality to train people in sports. Section 3.1 presents related work, concerning related exergames, movement games and rehabilitation games. In section 3.2, motivational aspects of serious games that teach people in sports or movements are described. Section 3.3 gives an overview of effective ways of providing feedback in serious games that are aimed at teaching people in sports or movements. In section 3.4, the effects that cause a learning effect in serious games that train people in sports or movements are described. This is followed by an explanation of cyber sickness and its causes in section 3.5. Finally, section 3.6 provides a conclusion, describing aspects that will be taken from this background research to the design process.

3.1 Related work

It is difficult to find a system that uses a serious game in augmented reality to train people for the particular case of skiing. However, serious games that use augmented reality to train people in sports or movements were found. The results include exergames, movement games, and rehabilitation games. Exergames are games that encourage people to exercise [2], [19]. The category of movement games entails the games in which people do not learn sports, but focus on learning certain movements instead. Rehabilitation games use game technology to help and motivate people in their physical rehabilitation. Per category a number of relevant games will be mentioned and explained.

3.1.1 Exergames

Various exergames were found, although only the games Calory Battle AR and GeoBoids actually make use of augmented reality. Calory Battle AR is a mobile augmented reality exergame platform that uses sensors to connect between the real world and the game [7]. In Calory Battle AR the player has to help the Dewes to fight the Caloroids by finding and deactivating calory bombs that were placed around a geographical area. In order to do so, the player has to go outside and perform physical activities. GeoBoids is a game that is rather similar to Calory Battle AR. It uses augmented reality to display virtual creatures, the GeoBoids, on a map on the player's smartphone. The player has to find and catch them within a set time limit [7], [20]. While playing the game, the player can see the GeoBoids moving around in the real world. Both exergames use augmented reality to make a connection between the game and the real world, making the user feel more context aware and motivated in the games.

3.1.2 Movement Games

YouMove is an augmented reality game that teaches the trainee how to perform bodily movements. The trainee is presented to an augmented reality mirror that is overlaid with a simplified representation of the human skeleton. The human skeleton on the mirror makes certain movements

that should be mimicked by the user. A Kinect is used to track the position and pose of the user, which is directly compared to the pose presented on the mirror to determine if the user is correctly imitating the human skeleton. The game consists of five stages which should be executed in the following order: “demonstration”, “posture guide”, “movement guide”, “mirror”, “on your own”. In every stage the trainee becomes less dependent on the system, which requires more skills from the trainee [21]. To conclude, YouMove is a game that teaches the user movements by making the user less reliant on the game system.

Reidsma et al. [22] designed a very different system that motivates its user to perform movements, which entails a virtual trainer that looks and behaves like a human. This system is not a game because it does not contain any game elements, such as competition, the chance of winning or the risk of losing. However, it is seen as relevant to this research since it stimulates users to perform physical activity. The movements to be made and the pace are determined and shown by the virtual trainer based on the user’s heart-rate [22]. The system uses anthropomorphic behaviours and representations to motivate and activate its user.

3.1.3 Rehabilitation Games

Tannous et al. [4] and Hossain et al. [23] both designed two comparable games that adopt game technology and augmented reality for rehabilitation purposes. The serious game concept by Tannous et al. shows similarities to the work of Anderson et al. [21] with the game YouMove. Just as in YouMove, this game uses a Kinect to make a three-dimensional visualisation of the player’s body which is directly compared to a model of the right position or movement to be made by the player. The model of the position or movement is brought into the system by the expert, the person who helps the patient in the rehabilitation process [4]. The second game, SIERRA from Hossain et al. [23], uses augmented reality to display virtual objects on a real table, which should be reached for or picked up by the player, the rehabilitation patient. For both games it is the case that the more movements the patient is able to carry out correctly, the more complicated the next movements in the game will be.

3.2 Motivational aspects

In a serious game motivational aspects are needed to allow the game to have an educational effect on its players. That is because people need motivation to continue playing the game. Only when people play the serious game long enough they get the chance to learn something from it. Therefore, motivating the players of a serious game to play the game contributes significantly to the effectiveness of the serious game.

Games in itself are seen as entertaining and therefore motivate people to play them. Serious games make use of the concepts that make games entertaining, such as rules and a clear goal that should be reached [3]. Hossain et al. [23] argue that by using the entertaining aspects of regular games, serious games become more entertaining and motivate their players. However, Iten and Petko [24] disagree and state that entertainment in games has also proven to distract the player. Distraction prevents the player from having a focus on the learning goal of the serious game, which undermines the purpose of the serious game. Therefore it remains doubtful to what extend a serious game should be entertaining.

Besides entertainment, multiplayer games are more motivational for their players than single player games. There are several reasons why people are more motivated to play multiplayer games. First of

all, Göbel et al. [2] state that multiplayer games offer competition. When there is competition, people want to prove that they are the best which makes them likely to continue playing the game until they get near that goal. Secondly, playing against a human opponent is less predictable than playing against the computer, which enhances the replayability of the game [2]. Thirdly, Whitehead et al. [19] add that there is a social aspect in multiplayer games. They argue that peer pressure plays an important role in motivating the fellow player. Because of these three factors, people will enjoy the game for a longer time and therefore feel motivated to play the game.

Also, classic game elements are known for contributing to a player's motivation to play the game. Göbel et al. [2] explain that when players are presented to a ranking of the best players, they tend to compare their personal results to those of others, which makes them want to do better in the game and continue playing. Cheng and Liu [25] add to this that a clear goal in the game makes the player aware of the gap between what should be reached and the current situation that the player is in. When the goal is clear and the game offers means that can be used to reach it, the player feels motivated to keep on playing until the goal of the game has been reached. Also, Whitehead et al. [19] state that games that do not allow for cheating are more motivational than games where cheating is possible. Therefore, when cheating is impossible, people will feel motivated to play the game because that is the only way they can reach their desired result. All in all, rankings, a clear goal and fair play are classic game elements that motivate the players of a game.

Furthermore, personalization of a game increases the motivation of its players as well. This works especially well for sports games. Hardy et al. indicate that serious sports games can be personalized by giving trainers and trainees the option to add personal content to the game, such as training schedules and exercises. Using this input, the training will be at an appropriate level of challenge for the trainee [5]. It is clear that a challenging but manageable game adds to the motivation of the player.

Another motivational factor in games, but also in general, is the use of a virtual human-like character as a trainer. People have the tendency to follow the behaviour of the virtual trainer as long as it looks and behaves like a human. Reidsma et al. [22] claim that a human-like virtual coach can be able to make people do certain fitness exercises without verbally giving them the command to do so. People tend to copy the behaviour of the human-like trainer. Therefore, a human-like representation in a game that verbally or non-verbally transmits what should be done adds to a player's motivation to continue playing the game.

A final motivational factor in games are rewards, which are usually given for specific performances or actions in the game. According to Swartz and Lyons [26] rewards offer the player a favour or advantage in return for his/her performance in the game, which has a positive effect on the player's motivation to continue playing the game. Goh et al. [27] suggest that rewards reinforce a player's enjoyment and make a player feel self-determined and competent, which adds to the motivation of the player. However, Cruz et al. [28] contradict these statements by claiming that rewards are not stimulating every form of motivation. According to the Self-Determination Theory from Deci and Ryan [29] there are two types of motivation: intrinsic motivation, and extrinsic motivation. Intrinsic motivation is defined as an internal and inherent desire to do an activity for one's own pleasure and satisfaction, while extrinsic motivation is a feeling of motivation that comes from external sources instead of from an internal drive. Cruz et al. [28] claim that rewards increase extrinsic motivation and decrease intrinsic motivation. Rewards stimulate a player to play for the goal of getting more rewards, instead of playing out of an internal drive to do so. This means that when the rewards are removed from the game, the player will not feel motivated to play the game anymore. The two most

commonly used types of rewards are points and badges [26], [27]. Points and scores are the result of change in behaviour [26] and supply the player with insights in personal performances [27], while badges are seen as an indication of a player's status in the game [27].

3.3 Feedback

Another important aspect in serious games is the feedback provided, since people need feedback to know what they did right and what they did wrong. Based on feedback, improvements can be made which will enable the player of the serious game to make progress. In the examined literature three effective ways were found to provide feedback.

First of all, summary feedback and intermediate feedback play significant roles. Summary feedback can be defined as feedback at the end of a series of trials in the game, whereas intermediate feedback is provided at every single trial. Anderson et al. [21] state that providing a player with intermediate feedback can possibly cause an overload of information presented to the player, which will hardly benefit their performance. They add to this that supporting a player by summary feedback instead will allow the player to think about the feedback for a longer time and will enable the player to improve his/her skills throughout a series of trials, without being interrupted. However, contradictory findings from Hossain et al. [23] show that real-time feedback is most effective, since it allows the user to know what to do in every stage of the game. Real-time feedback is rather related to intermediate feedback than to summary feedback, as it is provided continuously. Therefore, it remains unclear if summary feedback is actually preferred over intermediate feedback. However, it is expected that a restricted amount of intermediate feedback is desired from the trainee's side, since it allows the trainee to adapt behaviours according to the feedback at the right moments. The use of a limited amount of intermediate feedback can be accompanied by summary feedback at the end of every game to make the feedback more effective.

Second, multimodal feedback has positive effects on the player. Multimodal feedback is feedback that is provided to multiple senses [23]. Usually, multimodal feedback entails audio feedback, visual feedback, and possibly even haptic feedback. According to Hossein et al. [23], accessing multiple senses by the feedback makes the player more aware of the feedback and more likely to change his behaviour accordingly. Anderson et al. [21] add that audio, for example, has positive effects on the learning timing of the player. All in all, multimodal feedback makes the player more attentive and thoughtful towards his/her own behaviour in the game.

Third, the use of assessment games, intermediate games that assess whether the player actually learnt something from the normal game, are seen as a powerful way to address the player and provide feedback. Hossain et al. [23] state that by playing an assessment game, the player will discover what skills were learnt throughout the gameplay and what skills are still inadequately developed. This will help the player to clearly see what his/her competences are and what should still be developed through gameplay. Assessment games also provide feedback to the game itself as to what level the player is currently at. Using assessment games, the game can determine what exercises or challenges the player needs in the regular game.

3.4 Learning effect

Since the aim of serious games is to educate players, it is important to look into the factors that cause a learning effect in them. A learning effect can be defined as the case where the player learns

something new from the game. A number of causes that make a serious game more effective in teaching were found.

First, clarity of the game enables the player to learn something from it. Ke [3] and Iten and Petko [24] agree that clear tasks, explanation and cues in a serious game cause a learning effect for the player. According to Iten and Petko [24] that is because a player's expectations of an easy and instructive game makes his/her approach towards the game more positive. Ke [3] adds that this becomes even more effective when the game builds upon the player's prior knowledge, as this makes it easier for the player to connect the gameplay experience to the educational content of the game. Ke [3] also states that in this case, rules play a significant role as well, to restrict the player in what can or cannot be done during gameplay. Rules empower the player's learning efforts even more as they force the player to find alternative ways to reach the goal of the game. All in all, clear tasks and explanations, the use of prior knowledge, and the use of rules add to the clarity of the serious game.

Second, a decrease in guidance during gameplay has positive effects on the player's learning as well. Anderson et al. [21] concluded that gradually reducing the guidance in the game forces the player to fill up the gap of missing guidance with increased skill. Therefore, lowering the amount of guidance in the game over time makes the player work harder on his/her skills and causes a learning effect. Cheng and Liu [25] add that the decrease in guidance also helps the player to get in "flow". Flow is the situation where a challenging goal is set in the game and the supplies and techniques that are offered enable the player to reach the goal. Learning in flow is most likely to happen when players experience a balance between their own skills and the challenges offered by the game, which is an important aspect to take into account when decreasing the guidance in the game.

3.5 Cyber sickness

An important aspect to consider when designing for virtual reality or augmented reality is cyber sickness. Bruck and Watters [30] define cyber sickness, also referred to as simulator sickness or virtual environment sickness [31], as a feeling of illness that is similar to motion sickness, while there is no physical motion present. Rebenitsch and Owen [32] confirm this definition and describe cyber sickness as an illness that is very similar to motion sickness, without the presence of actual physical motion. The symptoms of cyber sickness are comparable to the symptoms of motion sickness and include nausea, disorientation, headaches, and dizziness [31], [32].

3.5.1 Possible causes of cyber sickness

In the examined literature, six theories were found about the cause of cyber sickness. The first theory is the sensory conflict theory, which confirms the definitions mentioned above. According to Duh et al. [31] and Rebenitsch and Owen [32] the sensory conflict theory claims that cyber sickness results from contrasting information from a person's visual perception and inertial perception. An example of this is a system where the person is stationary in the physical world and gets visual signals of movement in the virtual world.

Second, lag is considered as a cause for cyber sickness. Milgram [33] argues that in augmented reality a lag of the graphics compared to the physically present world causes symptoms that are strongly related to cyber sickness. Rebenitsch and Owen [32] also consider lag as a possible cause for cyber sickness.

A third theory that explains a possible cause for cyber sickness is the postural instability theory. Riccio and Stoffregen [34] posed the postural instability theory as an alternative to the sensory conflict theory. The theory suggests that cyber sickness is caused by the fact that virtual environments are different from people's natural environment. They imply that virtual environments force people to find new ways of controlling their postural stabilities, since it is different from their natural environment. Similar to animals that get sick in environments where they cannot get control over their balance, humans get sick from losing their postural stability. Rebenitsch and Owen [32] also mention the postural instability theory as a possible cause for cyber sickness.

Fourth, the duration of exposure in augmented reality or virtual reality plays an important role. Rebenitsch and Owen [32] claim that the intensity of cyber sickness increases as the duration of exposure increases. Bruck and Waters [30] add to this that cyber sickness symptoms can already be increased after only six to ten minutes of exposure.

The fifth theory that possibly explains the cause of cyber sickness is the rest frame theory. Rebenitsch and Owen [32] describe the rest frame theory as a theory that is rather similar to the postural stability theory. The rest frame theory claims that cyber sickness is caused by disagreements between the direction a user thinks is upwards based on what he or she sees in the virtual environment, and the actual upwards direction in the physical world. In other words, this theory posits that if the virtual environment is tilted compared to the physical world, a user may experience cyber sickness.

A sixth theory that was found on the cause of cyber sickness is the Eye Movement Theory of motion sickness. Bruck and Watters [30] reported that fatigue is one of the components of cyber sickness. They claim that rapid movement of the eyes results into tiredness of the eye muscles. In their research they linked this to the Eye Movement Theory posed by Ebenholtz [35], which suggests that overstimulating the muscles in the eye as a result from exposure to a virtual environment can cause tiredness of the eye muscles and headache, which are symptoms of cyber sickness [31], [32].

3.5.2 Decreasing symptoms

Contrary to the possible causes of cyber sickness, there is not much information on ways to decrease or prevent its symptoms. In the examined literature, only few methods were found to decrease cyber sickness symptoms. Repeated exposures to the virtual environment can decrease symptoms of cyber sickness. Duh et al. [31] mentioned a research from Kennedy and Fowlkes [36] in their work, that showed that symptoms of cyber sickness decreased with repeated exposures to the virtual environment. Besides the number of exposures, Rebenitsch and Owen [32] claim that limiting the horizontal field of view and including the physical world in the virtual environment reduces cyber sickness symptoms. The latter suggests that cyber sickness symptoms will be less severe in augmented reality environments than in virtual reality environments, since augmented reality environments include the physical world [6], [7].

Often times it is expected that cyber sickness symptoms can be decreased by improvements on the technology. Duh et al. [31] report that it is often expected that improvements in computer hardware can reduce cyber sickness. However, they oppose to this that, considering the sensory conflict theory is true, improvements to the hardware will potentially even stimulate cyber sickness. Findings by Rebenitsch and Owen [32] confirm this expectation, as they found that symptoms of cyber sickness increased with improved technology. Based on these findings, it is expected that cyber sickness symptoms will continue to increase with further improvements to technology.

3.6 Conclusion

Overall it can be concluded that several reliable methods exist that teach people in sports through a serious game in augmented reality. In the examined literature, methods were found that are related to motivational factors, feedback and learning effects in serious games. Also, it must be taken into consideration that cyber sickness symptoms can appear when people play such games.

A number of exergames, movement games and rehabilitation games were found in a state of the art research. Studies conducted on these games showed that, although players get motivated by the entertaining aspect of serious games, this can also be a distraction for them. Furthermore, multiplayer games, classic game elements such as rankings and a clear goal in the game, personalization of the game, a human-like representation of the trainer or coach, and including rewards in the game are successful motivational factors in serious games. With regard to the feedback provided in serious games it remains unclear if summary feedback or intermediate feedback is most adequate. However, based on the findings it can be concluded that multimodal feedback and the use of assessment games are reliable and effective. Finally, clarity in games such as clear tasks, explanation and cues enforce the learning effect that the game has on the player. Besides that, a decrease in the guidance that the game offers also shows positive learning effects.

Cyber sickness symptoms, causes, and ways to prevent it were examined as well. Cyber sickness symptoms are very similar to the symptoms of motion sickness, while there is no physical motion present when one suffers from cyber sickness. Possible causes of cyber sickness are the sensory conflict theory, lag of the graphics, the postural instability theory, long exposures to the virtual environment, the rest frame theory, and the Eye Movement Theory of motion sickness. Cyber sickness symptoms can be prevented or decreased by having repeated exposures to the virtual environment, limiting the horizontal field of view, and inclusion of the physical world in the virtual environment. Based on these findings it is expected that augmented reality, which presents the user to an environment where the physical world is overlaid with three-dimensional virtual objects, will not cause severe cyber sickness symptoms.

3.6.1 Design Recommendations

Based on the findings of this chapter several recommendations can be made about the design of the serious skiing game. First of all, the envisioned game should be a multiplayer game, since that adds to the motivational impact that the game has. Second, players should be able to earn points in the game. Rankings should show how many points every single player has earned so that individual players can compare their achievements to those of their opponents or other players. The rankings will provide the players with more insight into their own performance in the game. Third, intermediate feedback should be used. However, this should be done to a limited extent, to prevent the player from being overloaded with information. The intermediate feedback should be in the form of multimodal feedback, providing the player with visuals and audio that show areas where improvement is possible and indicate how the improvements should be made. The actual look of the multimodal intermediate feedback will be further explored in Chapter 4 Ideation and Exploration. Fourth, the option to accompany the intermediate feedback with summary feedback will be explored. At the end of every skiing attempt the player could get an overview of the things that went well and the things that did not go well and how they could be improved. The summary feedback could be given by a human-like virtual representation of a trainer or by multimodal feedback as well. The latter is preferred since it is also used in other parts of the game. The final look of this form of feedback will also be further explored in Chapter 4.

Chapter 4 – Ideation and Exploration

This chapter describes the results of the ideation phase, which was executed to gather possible product ideas for the prototype that will be the result of this project. The ideation phase consisted of diverging phases and converging phases. In the diverging phase, possible ideas were explored, which were brought back to one final idea in the converging phase. The result of this chapter is a final product idea, which will be further specified in the specification phase of this project.

4.1 First Ideation

The starting point for the ideation phase was the assignment that was given by the client of the project. The client gave the assignment to make a combination between a revolving ski slope and augmented reality or virtual reality. A revolving ski slope, also called infinite slope, is a ski slope with a revolving surface [37]. This causes the same effect as running on a running treadmill, but then for skiing. Figure 4.1 gives a visual representation of a revolving ski slope.

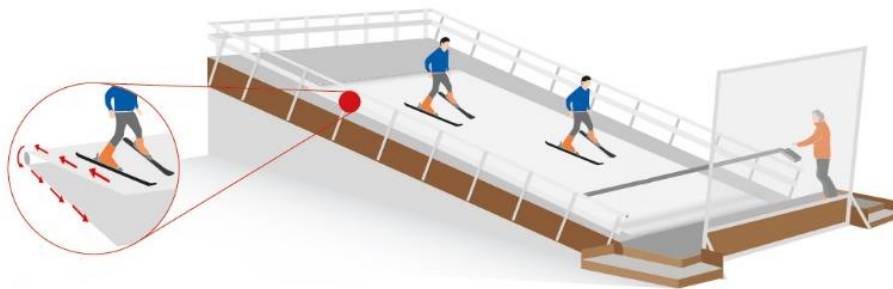


Figure 4.1: A revolving ski slope.

Based on the assignment, the first ideation phase was started. The goal of the first ideation phase was to determine what the exact assignment for this project was going to be, since a combination between a revolving ski slope and augmented reality or virtual reality can be made in many ways. Possible directions for this project that were thought of were the following:

1. The combination between a revolving ski slope and augmented reality or virtual reality to enhance the skiing experience. This could for example be done by stimulating the user's senses in such a way that an experience is being evoked that the user will strongly relate to the experience of real skiing. The user could be shown to a visual representation of a real ski area in the snowy mountains, accompanied with the sounds and smells that are present there. Cold air or the feeling of snow could even be included to add to the skiing experience.
2. The combination between a revolving ski slope and augmented reality to offer people help and instructions when they are skiing. This could for example be done by visualizing a track that indicates to the user where he/she should ski.
3. The creation of a skiing teacher or teaching system in augmented reality or virtual reality, which can be used to teach people how to ski.

4. The creation of a virtual coach that gives a skier instructions and exercises that can be done before the actual skiing activity, as preparation.
5. The combination between sensors that measure physical data, such as heart rate and blood pressure, and sensors that measure ski-related data, such as the pressure put on the skis. This can be combined with augmented reality or virtual reality to inform a skier about his/her data. Additionally, this could be taken even further by using the obtained data to control the user's behaviour based on it. For example, if the skier's heart rate is too high, a representation in augmented reality or virtual reality could be used to try to slow him/her down, which is likely to eventually also slow the skier's heart rate down.
6. The creation of a tracking system that can track a user's positions and movements in the physical world, on the actual ski slope that is, and uses that information to enable the user to move himself/herself through a virtual environment based on his/her movements in the real world.
7. The combination between skiing on a revolving ski slope and augmented reality or virtual reality to make a social platform for skiers, which allows them to stay in contact while skiing. The social platform would be presented to the users in augmented reality or virtual reality.

Out of the seven possible directions that are described above, two were considered most interesting and feasible within the limited time that is available for this project. The two directions that were further explored were direction number one, the combination between a revolving ski slope and augmented reality or virtual reality to enhance the skiing experience, and direction number three, the creation of a skiing teacher or teaching system in augmented reality or virtual reality. Directions number two and four were considered feasible, but less interesting than directions number one and three. Directions number five, six, and seven were not considered to be feasible within the limited time scope of this project. With respect to direction number five, a system that combines sensor input from multiple sensors, transfers the input to a form that can be used to make a representation of it in virtual reality or augmented reality, and combines the data in such a way that meaningful conclusions can be drawn upon it did not seem realistic and achievable within ten weeks. Regarding direction number six, tracking is a subject that has undergone extensive research and yet there are no easily implementable or reliable solutions found. Therefore, it is expected that this direction poses problems that are unrealistic to solve within this project. Concerning direction number seven, social platforms contain extensive internal architectures which are not seen as realistic to be developed within the limited timeline of this project. Apart from the feasibility of direction number seven, also its desirability was questioned. Distracting people from their skiing activities with a social platform seems to be an objectionable thing to do. For both of the chosen directions, two possible sub-directions were found. Table 4.1 lists the possible sub-directions per chosen direction.

Table 4.1: The possible sub-directions for direction number one and direction number three.

	Direction	Sub-directions
1	The combination between a revolving ski slope and augmented reality or virtual reality to enhance the skiing experience	Research about the influence that the virtual environment has on the skiing experience, investigating if the environment has to look realistic, or whether more abstract environments can also enhance the skiing experience.

		Research about which senses should be stimulated and with how much intensity to give users the feeling of skiing in the Alpes, instead of on the revolving ski slope.
3	The creation of a skiing teacher or teaching system in augmented reality or virtual reality	The creation of a system that uses simple icons to give skiers instruction and to teach them new skiing techniques, which will be displayed to them in augmented reality or virtual reality. The creation of a (serious) game that teaches people how to ski, using augmented reality or virtual reality techniques.

The last sub-direction, the creation of a (serious) game that teaches people how to ski, was finally chosen to be the direction for this project. This direction was chosen because it combines game design with augmented reality or virtual reality, which is seen as an interesting direction. Therefore, the rest of this chapter will focus on the direction of the (serious) teaching game for skiing. Figure 4.2 gives an overview of the converging and diverging phases of the first ideation phase that was described in this section.

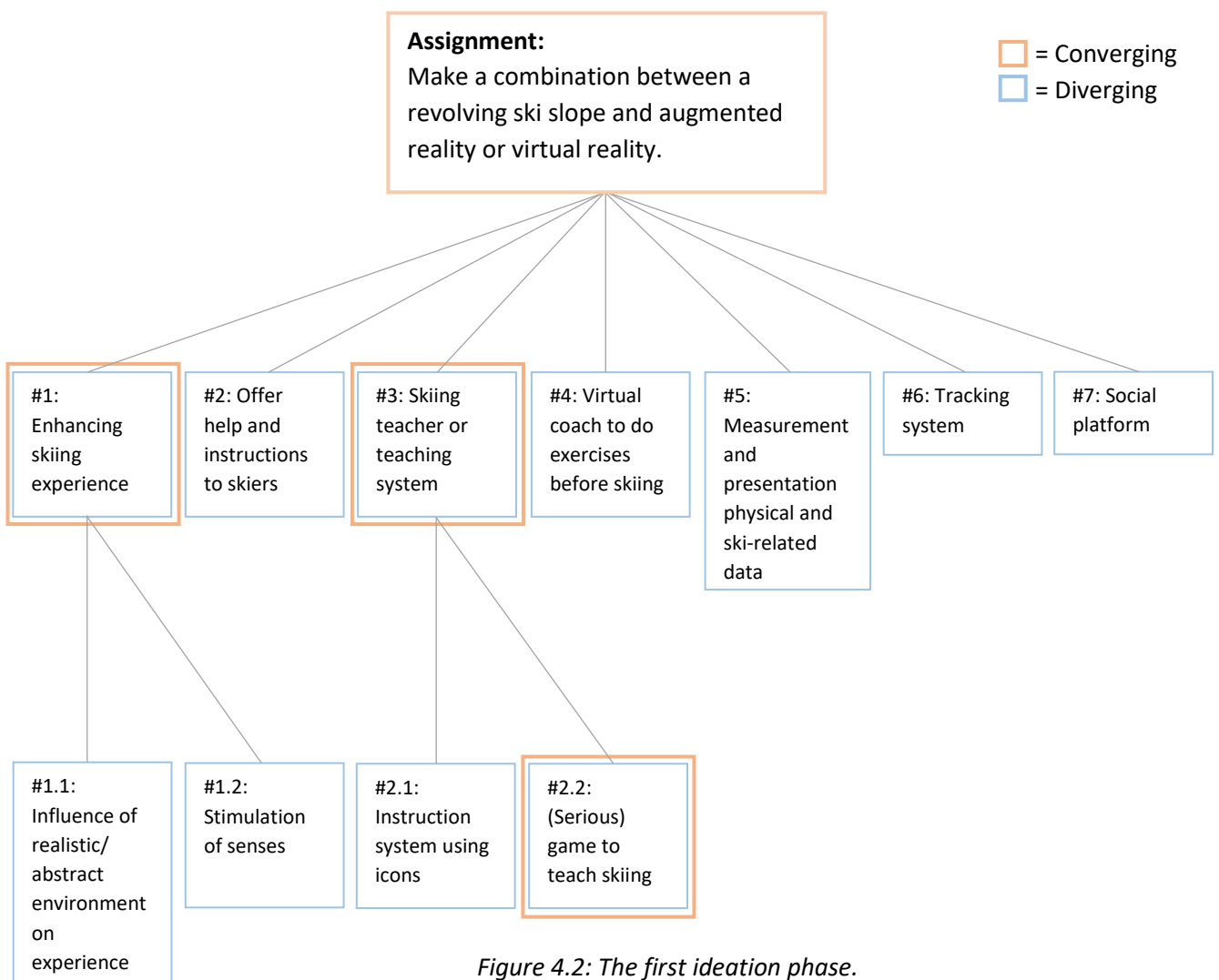


Figure 4.2: The first ideation phase.

4.2 Stakeholder Analysis

A stakeholder analysis was executed to identify who the product will be designed for. As was described in Chapter 2 Methods and Techniques, stakeholders are identified according to the methodology of Sharp et al. [9]. Sharp et al. identify four groups of baseline stakeholders, which are users, developers, legislators, and decision-makers. The last three groups are not important for the project in this early stage, as the focus is now on creating a working prototype that is aimed at the primary users. There is no product in existence yet, which means that developers, legislators, and decision-makers will not get confronted with the product and will not have an influence on it. This might happen at a later stage, when there is a product being readied for the market.

The identified stakeholders per category are listed in Table 4.2. Four groups of user-stakeholders were found, which include skiing pupils, skiing teachers, leisure skiers, and skiing centres. Skiing pupils are people who take ski classes to learn how to ski, they can be seen as the primary target group of the product. Skiing teachers are the people who teach the skiing pupils how to ski, they could be interested in using the product that is being developed throughout this project as a support tool during their teaching activities. Leisure skiers are people who practice skiing as a leisure activity, for entertainment. The product could potentially be interesting for them as well to add more fun to their skiing activities. Skiing centres are the places that employ the skiing teachers and where skiing pupils can take lessons before they actually go to the real skiing areas. Skiing centres could be interested in the product to offer it to their customers and their employees during skiing classes.

Table 4.2: Overview of the stakeholders per category for the skiing game.

Category	Stakeholders
Users	Skiing pupils
	Skiing teachers
	Leisure skiers
	Skiing centres
Developers	Software engineers and programmers
	Developers of revolving ski slope
Legislators	Government
	Insurance companies
Decision-makers	Management or CEO of the company that distributes the product

4.3 Early Design Choices

After the first ideation phase and the stakeholder identification, some early design choices could be made. The first and very essential choice was the choice between augmented reality and virtual reality. Augmented reality can be defined as the technology that presents the user to an environment where normally present surroundings are overlaid with virtual three-dimensional objects [6], [7], while virtual reality is the technology that presents the user to an environment that is entirely virtual and simply replaces the real world entirely [38], [39]. As people will be presented to the augmented reality environment or virtual reality environment while skiing on the revolving ski slope, it is advisable to include the real world in the application. When a person cannot see his/her physical, normally present surroundings during skiing, it is expected that the person might easily lose his/her balance on the ski slope, which can cause dangerous situations. One of the problems with virtual reality would be that people cannot see where they are located on the slope, so they can also

not correct their movements when they are likely to fall down the slope. Therefore, augmented reality was chosen as the most suitable technology for this project.

A second early design choice that was made was the choice to make a system that *supports* the ski-learning process, rather than a system that would *replace* the normal skiing teacher and become a teacher on its own. Since skiing pupils need intensive and detailed instruction when they ski on the slope for the first time, it is expected that this cannot be reached yet with a learning game in augmented reality. However, it is expected that the teacher can use the game as an extra tool during the teaching activities. Therefore, the skiing game will be designed as a tool that supports the ski-learning process, instead of actually teaching a pupil how to ski.

4.4 Use Cases

Based on the stakeholders that were identified in the stakeholder analysis of section 4.2, two use cases were made. The use cases were made based on the “casual” use case template by Cockburn [13], which was explained in Chapter 2 Methods and Techniques. The goal of the use cases is to identify requirements that are needed for the skiing game from a user’s point of view. Therefore, the use cases are made for the users that are listed in Table 4.2. Use case number one has skiing pupils and skiing teachers as the primary actors, use case number two has leisure skiers as primary actors and the employees of skiing centres as secondary actors. Besides the actors, every use case contains a scope, level, success scenario, and extension conditions. The latter consists of the steps that could go wrong in the success scenario. The use cases described in this section served as a starting point for the brainstorm sessions that were executed afterwards, as is described in the sections that follow.

Use case #1	
Title	Using the skiing game during ski classes
Primary actors	Skiing pupil Skiing teacher
Scope	For the pupil: The problem of boring ski classes on a revolving ski slope is being solved by introducing an augmented reality skiing game to the ski classes. This is an appropriate solution because games are a good means for entertaining people [3], [23] . For the teacher: The problem of only being capable of teaching one pupil at a time is being solved by using the augmented reality skiing game. This is an appropriate solution because the skiing game allows the teacher to let one pupil practice on his/her own with the skiing game while giving personal attention to another pupil, which can be reversed when the first pupil has finished the game.
Level	Internal
Success scenario	1. The skiing pupil comes to the skiing class.

Extension conditions	<ol style="list-style-type: none"> 2. The skiing pupil gathers the right materials (ski boots, skis) and enters the slope. 3. The teacher gives the skiing pupil a head-mounted display with the skiing game on it. 4. The pupil puts on the head-mounted display. 5. The teacher turns the revolving ski slope on. 6. When the pupil is stable enough, he/she can turn the game on and the game will start. 7. The skiing game challenges the pupil to make certain movements in order to perform well in the game. The pupil plays the game and the teacher can give his/her attention to another pupil in the meantime. 8. After some time, the game is finished. At that moment, the slope stops. The pupil is presented to summary feedback, which indicates how well he/she did in the game. 9. The pupil takes the head-mounted display off. 10. The teacher returns from the other pupil (who can now start playing the game) and decides what should be worked on during the lessons, based on how well the pupil performed in the game. <ol style="list-style-type: none"> 5. If the pupil has never used the head-mounted display before while skiing on the revolving slope, he/she might fall because distraction or disorientation. 6. Due to technological problems, the game might not start. 7. The challenges of the skiing game might be too easy or too difficult for the pupil, causing boredom or a feeling of incompetence. Challenges that are too difficult can also cause dangerous behaviour, because the pupil is demanded to perform actions that he/she is not capable of.
----------------------	---

Use case #2

Title	Using the skiing game for skiing for fun
Primary actors	Leisure skiers
Secondary actors	Employee of skiing centre
Scope	The problem of the lack of fun while skiing on revolving ski slopes is solved by using the augmented reality skiing game while skiing on the revolving ski slope. This is an appropriate solution because games are a good means for entertaining people [3], [23]. Besides that, the inclusion of the multiplayer aspect in the game allows multiple leisure skiers to ski together, even over distance.
Level	System
Success scenario	<ol style="list-style-type: none"> 1. The leisure skier goes to a skiing centre with revolving ski slopes. 2. The leisure skier gathers the right materials (ski boots, skis) and enters the slope.

Extension conditions

3. An employee of the skiing centre gives the leisure skier a head-mounted display with the skiing game on it.
 4. The leisure skier puts on the head-mounted display.
 5. The leisure skier can start the game application.
 6. The leisure skier has the option to connect his/her game to other skiers, by virtually inviting skiers that are using the application at the same time (which will be shown in the application). These other skiers might be acquaintances of the leisure skier. However, the leisure skier can also connect to skiers that he/she does not know.
 7. When the connection is made and all players are satisfied with the amount of fellow players, the game starts.
 8. The employee of the skiing centre turns the revolving ski slope on.
 9. The leisure skier starts playing the game. The goal is to perform well enough to earn the most points and reach the finish earlier than the opponents.
 10. When all players have reached the finish, the game ends.
 11. The employee stops the revolving ski slope.
 12. Summary feedback provides all players that were present in the game with an overview that compares their personal achievements to those of the other players.
 13. The leisure skier can decide to play the game again (repeat from step 5 onward), or to stop.
 14. The leisure skier takes the head-mounted display off.
 15. The leisure skier leaves.
-
5. Due to technological problems, the game might not start.
 6. Due to connection problems, it might not be possible to connect to other players at all times.
 7. Due to technological problems, the game might not start.
 9. The challenges of the skiing game might be too easy or too difficult for the leisure skier, causing boredom or a feeling of incompetence. Challenges that are too difficult can also cause dangerous behaviour, because the leisure skier is demanded to perform actions that he/she is not capable of.
 12. Due to technological problems, summary feedback might not appear.

4.5 First Brainstorm

Based on the choices made in the previous sections and the insights from the use cases, a first individual brainstorm about the design of the skiing game and its interactions was executed. Chapter 3 State of the Art contains the background research that served as a basis for the decisions that were made for the design of the skiing game. In the concluding section of that chapter, some design recommendations were already made for the skiing game. These recommendations can be summarized as follows: the game should be a multiplayer game, players should be able to earn points in the game, players should be presented to a ranking with points, limited intermediate feedback should be used in the game, the feedback should be in the form of multimodal feedback, and summary feedback could be included if possible. The first brainstorm was held based on these

recommendations. Therefore, the brainstorm was conducted on the following four topics that need to be included in the game: skiing and earning points, the presence of a fellow player, feedback, and levels.

4.5.1 Skiing and Earning Points

This part of the brainstorm was about what tasks the user should complete while skiing in the game and how points can be earned with it. A total of six ideas were found regarding this topic. First the ideas numbered one to five were found. Idea six arose by combining ideas four and five. Figure 4.3 contains the ideas that were found. All the six ideas are summarized in Table 4.3 below the figure.

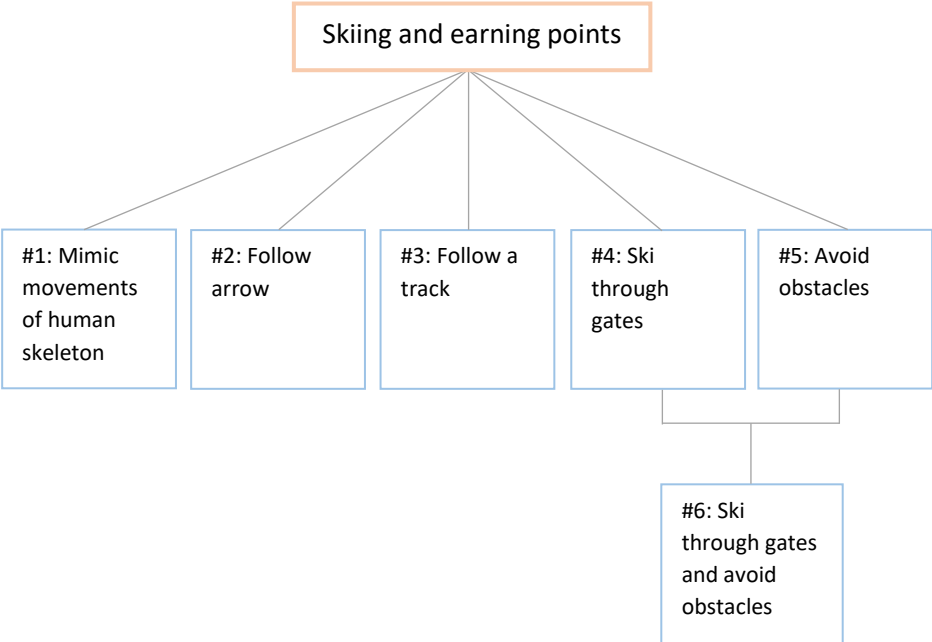


Figure 4.3: The ideas that were found for skiing and earning points in the game.

Table 4.3: The ideas for skiing and earning points in the game and their explanations.

Skiing and earning points	
Idea	Explanation
#1	The user sees a representation of a human skeleton in front of him/her. The human skeleton performs movements that should be mimicked by the user. If the user is mimicking the movements of the skeleton correctly, points will be earned. When the user does not copy a movement or when the user is too late, points will be lost. The more points, the more difficult the movements to be made become and the faster they need to be executed.
#2	Instead of a representation of a human skeleton, an arrow pointing in a certain direction will guide the user in the movements to be made. However, this is very limited version of the game as it can only steer the user in a direction. The arrow has to stop pointing in the direction when the user is at the right place, because otherwise the user will reach the side of the ski-slope which might lead to dangerous situations. Again, points can be earned by following the instructions. When the instruction is missed or carried out wrong, the user loses points. The more points, the more difficult the game becomes.

#3	A track will be displayed in the game on the surface of the ski slope. The track can contain curved parts and straight parts. The player has to follow the track. On the track are certain marks that are placed at a predefined distance from each other. When the player manages to ski on the track from one mark to another, points will be earned. The options could be added that the player loses points if he/she does not manage to ski on the track between two marks.
#4	The user has to ski through gates that are randomly being placed at the ski slope. The user can see the gates coming from a distance. If the user successfully skies through the gates, points will be earned. If the user misses a gate, no points will be earned or points might even be lost. A feature that could be added to this idea is to make the placement of the gates more difficult when the user earns more points. That means that the more points the user earns, the more skiing skills he/she has to show.
#5	This idea is based on the actual situations during the teaching sessions at a revolving ski slope in Deurningen. During these sessions the teacher places cones on the revolving ski slope which should be avoided by the pupil. In a similar manner, obstacles could be placed in the serious game. To make the situation more realistic, a person falling down could be an obstacle that should be avoided by the player. A person falling down is harder to avoid and less predictable than a cone that is being placed by a teacher, as the pupil can already see the cone coming long before he/she is there. When one of the obstacles is hit by the player, points will be lost. On the contrary, points can be earned by successful in avoiding the obstacles. Points can be allocated based on the time that the user did not hit an obstacle. The longer the player avoids obstacles, the more points he/she will get. The more points, the more difficult the placement of the obstacles becomes.
#6	This idea is a combination between the idea of the gates and the idea of the obstacles. Both the gates and the obstacles will be present in the game. The player can earn points by successfully skiing through the gates and he/she can lose points by colliding with an obstacle.

4.5.2 Presence of the Fellow Player

This part of the brainstorm was about the presence of the fellow player in the game, in case this would be implemented. A total of five ideas for presenting a fellow player arose, which are shown in Figure 4.4. Table 4.4 contains the explanations of the five ideas.

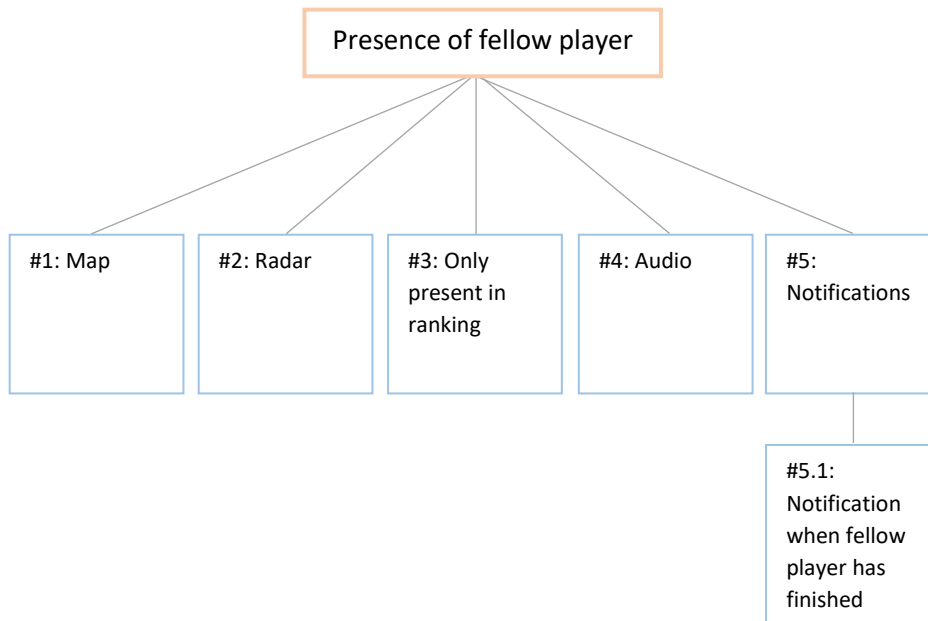


Figure 4.4: The ideas that were found for the presence of a fellow player in the game.

Table 4.4: The ideas for the presence of a fellow player in the game and their explanations.

Presence of fellow player	
Idea	Explanation
#1	A map that shows where you, the player, are and where your fellow players are. This could be placed in a corner in a small format.
#2	A radar that shows where you, the player, are and where your fellow players are. This could be placed in a corner in a small format.
#3	Fellow players do not have a visual presence during gameplay at all. They are only present in the overall ranking that will be shown to the player once the game or a trail in the game is over.
#4	The fellow players can be represented through audio. The player will hear skiing sounds, such as the sound of ski blades moving over snow, that indicate that someone is present in front of or behind him/her. If the opponent is in front, the player will hear the audio coming from the front. If the opponent is behind the player, the player will hear the audio coming from behind. If this is done, it could be made more realistic by using a visual representation of the opponent as well when he/she is in front of the player.
#5	Over time certain notifications can be given that tell the player where the opponent is. For example, every thirty seconds the game could tell the player “ <i>opponent is 200 meters in front of you</i> ”, or something similar. This message could subtly be placed in a corner of the visual part of the game. It should be taken into account that this option could lead to distraction of the player.
#5.1	A very subtle version of presence of a fellow player, which is rather similar to the idea described above, is that a player gets a notification once the fellow player has reached the end of the game. However, then the player can only see where the opponent is for once, only during gameplay, and only in case his/her opponent has reached the end of the game earlier. In other words, the player that finishes first

		never gets the notification of the other player finishing, which could result in him/her not being aware of the fellow player during gameplay.
--	--	--

4.5.3 Feedback

This part of the brainstorm session was about how feedback could be provided to the players of the skiing game. As can be seen in Figure 4.5, a total of six ideas arose on this topic. First, the ideas one to five were found, after which the combination of ideas one, two, and three led to idea six. All the six ideas regarding feedback in the game are explained in Table 4.5.

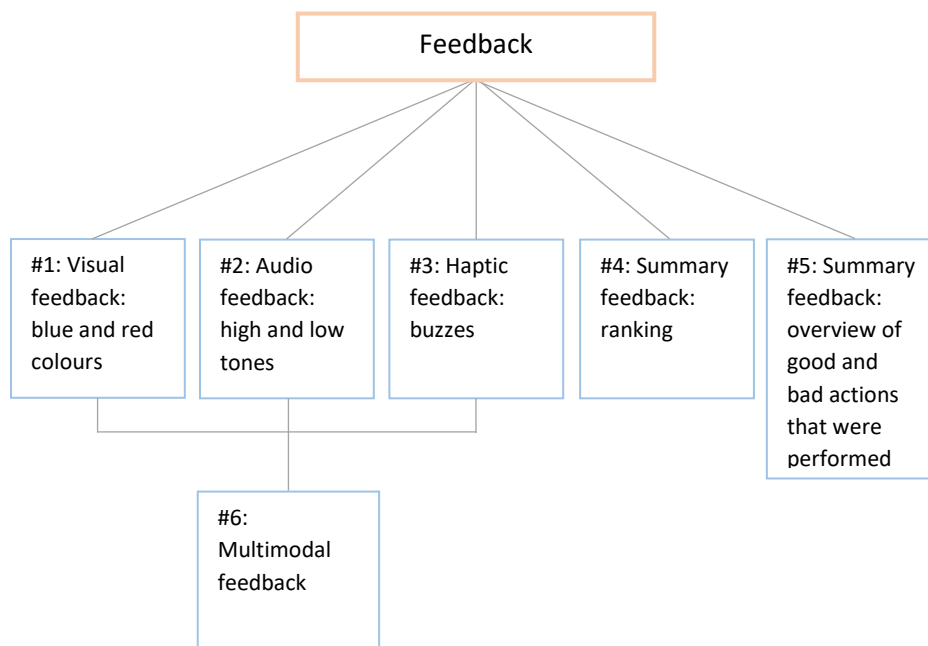


Figure 4.5: The ideas that were found for providing feedback in the game.

Table 4.5: The ideas for providing feedback in the game and their explanations.

Feedback	
Idea	Explanation
#1	Visual feedback. When the player is doing something right, the thing that was done right will light up in a blue colour and the points that were earned with it will be displayed in a blue colour as well, indicating that they add up to the total number of earned points. Blue seems to be a suitable colour for giving positive visual feedback because blue is seen as calm, serene, and healing [40]. On the other hand, when the player is doing something wrong, the thing that was done wrong will light up in a red colour and the points that were lost with it will be displayed in a red colour as well, indicating that they are subtracted from the total number of earned points. Red seems to be a suitable colour for giving negative feedback because red is seen as an intense and sometimes even angry colour [40].
#2	Audio feedback. A high tone (positive sound) will be played when the player is doing something right, a low tone (negative sound) will be played when the player is doing something wrong. The audio also indicates if points are earned, or lost.

#3	Haptic feedback. A light buzz will be given on the augmented reality device when the player is doing something right, and a heavy buzz will be given when the player is about to lose the game or when the player is doing something wrong.
#4	Summary feedback. A ranking at the end of a level that shows how well the player did compared to his/her previous achievements and the achievements of others. The player's achievements and placement in the ranking will be represented by the amount of points earned by the player and, if present, the opponent(s). This way, the player can see how well he/she is doing compared to others or to his/her own previous results.
#5	Summary feedback. At the end of a level or at the end of the game an overview will be provided to the player that shows the moments that went very well (where maximum points were earned) and the moments that did not go so well (where no points were earned or points were lost). For the moments that did not go so well, suggestions will be given on how the player can improve his/her performances. The moments will be shown to the player by screen captures of the game at that moment.
#6	Multimodal feedback. This is the combination of idea one, two, and three, which results into a feedback system where visual feedback, audio feedback, and haptic feedback are present.

4.5.4 Levels

The final part of the individual brainstorm session was about the way in which levels should be implemented in the skiing game. This part of the brainstorm session resulted into four ideas, which are all included in Figure 4.6 and explained in Table 4.6.

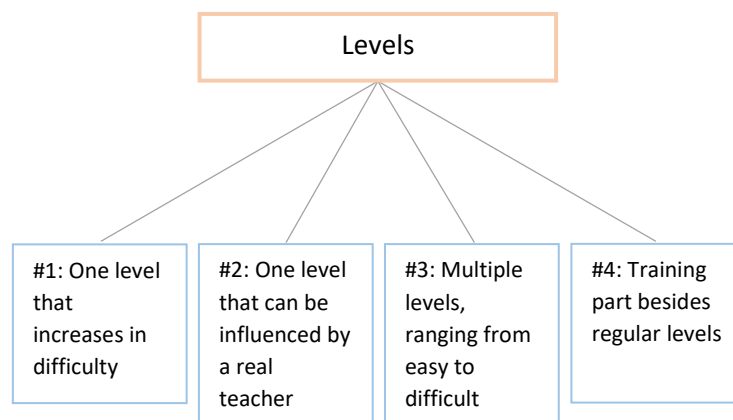


Figure 4.6: The ideas that were found for implementing levels in the game.

Table 4.6: The ideas for implementing levels in the game and their explanations.

Levels	
Idea	Explanation
#1	A one-level game where the level will be increased in difficulty, variation and unpredictability over time to make sure that players that are longer in the game are or become more skilled at skiing. Points can be earned in similar ways as described in the first section of this ideation section. The more points, the better a player did in the level.
#2	A one-level game where a human teacher has influence on the game and decides on its difficulty, duration and unpredictability. The teacher can do this through his/her own

	interface on a separate device, for example on a tablet, phone, laptop. This way, there is still human judgement in the game, determining what level of difficulty the player can handle.
#3	A game with multiple levels. The first level starts simple and does not last very long (one or two minutes). The second level is slightly more difficult and lasts a little longer. Slowly but steadily the difficulty and amount of time will be increased in the levels. A player can only play a certain level once he/she has completed all the levels that were before it. Hence, a player can only go to level five once levels one, two, three, and four have been successfully completed. A level will be successfully completed when the player has played the whole level and earned a sufficient amount of points.
#4	A training part besides the levels, where the player can practice before entering the game. In the training part no points can be earned, it is mostly focussed at showing the right movements to the player and at letting the player follow the movements. The player will get visual and audio feedback. For example, when the player does something right in practice, he/she will see a green representation of what went right and hear a high tone (positive sound). The player's achievements in the training part of the game do not give any points, achievements, head start or other kinds of benefits in the game.

4.6 Brainstorm and Evaluation with Client

After collecting ideas about different parts of the skiing game through an individual brainstorm session, the ideas were presented to the client for a second brainstorm session and an overall evaluation of the ideas. The brainstorm session with the client was based on the ideas that were already found during the individual brainstorm session that was executed beforehand. Also, since there was already a number of ideas in existence, the brainstorm session with the client was smaller than the individual brainstorm session. The results of the brainstorm session with the client included three extra ideas for the representation of the fellow player in the game. These ideas are included and explained in Table 4.7, where the numbering of the ideas is following the numbering of the ideas that were generated in the first brainstorm session for this part of the game.

Table 4.7: The results of the brainstorm session with the client.

Presence of fellow player	
Idea	Explanation
#6	Giving opponents the possibility to stay in contact through a head-set, which enables them to talk to one another. It must be noted that this can also be very distractive for the players.
#7	The fellow player can be present in the game through a shadow or three-dimensional model that represents a person who is skiing. The shadow or model should be placed at the location of the fellow player in the game.
#8	The fellow player can be present in the game through an arrow that is pointing at his/her location in the game, with the fellow player's avatar printed on the arrow so that it can be seen who the fellow player is.

After the second brainstorm session, idea generation ended and idea evaluation started. Together with the client, all the generated ideas were rated on feasibility within the limited timeline of this project and desirability from the side of the client to be included in the skiing game. Table 4.8 provides an overview of the evaluation of the generated ideas. The last column of the table provides a total score that was calculated by adding the score for feasibility to the score of desirability. The

amount of plusses and minuses in the column with the total score indicates how likely it is that the idea will be implemented in the skiing game.

Table 4.8: Evaluation of the ideas that were generated through the brainstorm sessions.

Evaluation of generated ideas			
Skiing and earning points			
Idea	Feasibility ++ = Very feasible + = Feasible - = Not very feasible -- = Not feasible at all	Desirability ++ = Highly desirable + = Desirable - = Not very desirable -- = Not desirable at all	Total
#1: Mimic movements of human skeleton	-	+	0
#2: Follow arrow	++	-	+
#3: Follow a track	+	-	0
#4: Ski through gates	++	+	+++
#5: Avoid obstacles	++	+	+++
#6: Ski through gates and avoid obstacles	++	++	++++
Presence of fellow player			
Idea	Feasibility	Desirability	Total
#1: Present on map	+	+	++
#2: Present on radar	+	+	++
#3: Only present in ranking	++	-	+
#4: Present through audio	+	+	++
#5: Present through notifications	++	-	+
#5.1: Notification when fellow player has finished	++	-	+
#6: Present through contact via head-set	-	-	--
#7: Present through shadow or model of a skier	++	+	+++
#8: Present through arrow with avatar on it	+	++	+++
Feedback			
Idea	Feasibility	Desirability	Total
#1: Visual feedback: blue and red colours	++	+	+++
#2: Audio feedback: high and low tones	++	+	+++
#3: Haptic feedback: buzzes	-	+	0
#4: Summary feedback: ranking	+	+	++
#5: Summary feedback: overview of good and bad actions that were performed	--	++	0
#6: Multimodal feedback	+	++	+++
Levels			
Idea	Feasibility	Desirability	Total
#1: One level that increases in difficulty	+	+	++
#2: One level that can be influenced by a real teacher	--	++	0
#3: Multiple levels, ranging from easy to difficult	-	++	+
#4: Training part besides regular levels	--	++	0

4.7 Product Idea

As a result of all the ideas that were generated throughout the ideation phase described in this chapter, a final product idea was created. The decisions made for the final product idea were based

on the evaluation of the ideas with the client, where the ideas that obtained the highest score were chosen to be included.

In the early stages of the ideation phase it was decided that the final product of this project will be a (serious) skiing game. With regard to skiing and earning points in the game, it was decided that the game will present the player to a series of gates and obstacles. The player can earn points by skiing through the gates, and points are lost if the player collides with an obstacle. As was decided earlier on, an attempt will be made to make the skiing game a multiplayer game. The presence of the fellow player in the multiplayer version of the game will either be through a shadow or model of a skier at the location of the fellow player, or through an arrow that is pointing at the location of the fellow player with an avatar chosen by the fellow player on it. The choice between one of these options will be made when it becomes clear which option is implementable within the limited scope of this project. With regard to the feedback that will be provided to the players of the game, it is decided that the players will receive multimodal feedback while playing the game and that they are presented to a ranking of players and their scores at the end of the game. The latter serves as a form of summary feedback. In the most ideal case, the multimodal feedback will consist of visual feedback, audio feedback, and haptic feedback. Finally, the game will consist of one level that increases in difficulty over time.

Chapter 5 – Product Specification

This chapter contains the results of the specification phase, which was executed to explore the design space and to decide on the functional specifications of the skiing game. Requirements elicitation was done by creating early prototypes, which include a paper prototype of the product idea that resulted from the ideation phase and several technology prototypes that were made to explore the possibilities that are offered by existing technologies. Based on these results, a requirements analysis was executed in which the requirements of the skiing game were determined. Finally, the game design of the skiing game is specified.

5.1 Early Prototypes





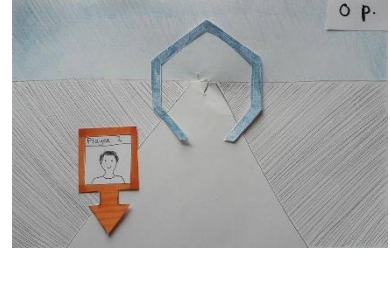
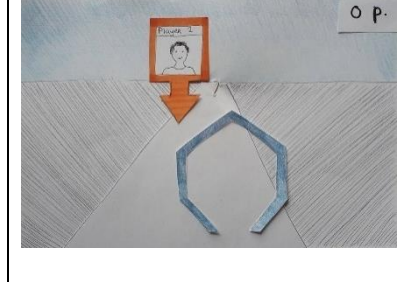
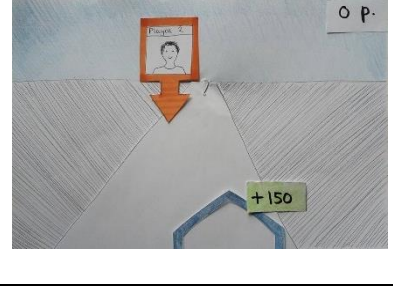
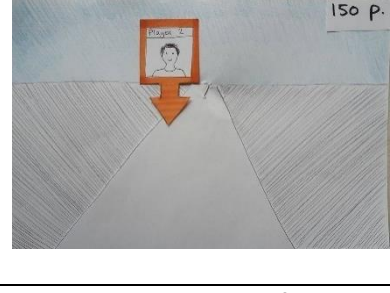
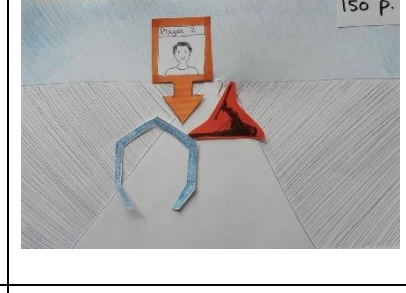
At the start of the requirements elicitation phase some early prototypes were generated to investigate the product idea that resulted from the ideation phase and to explore the possibilities that are being offered by the technology that is at hand. To do so, prototypes were made in two different ways. First, a paper prototype of the skiing game was made to specify and visualize the product idea that the ideation phase ended with. Paper prototyping was chosen for this purpose since it allows for rapid prototyping and offers the possibility of quickly adjusting the prototype. Second, three technology prototypes were made to explore the possibilities of augmented reality technology and multiplayer games. Out of these three technology prototypes, two were made as augmented reality prototypes and one was made as a multiplayer prototype.

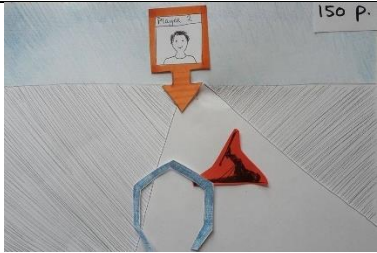
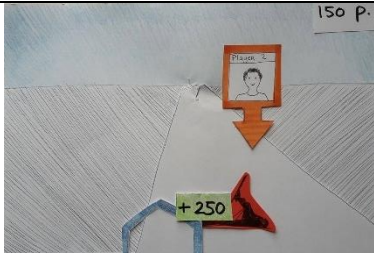







5.1.1 Paper Prototyping

A paper prototype was made based on the product idea that resulted from the ideation phase. The purpose of the paper prototype was to visualize and concretize the final product idea that resulted from the ideation phase, so that it is clarified what must be made in the final prototype that will result from the realisation phase. The results of the paper prototyping activity and a scenario for the skiing game are included in Table 5.1.

Table 5.1: A use scenario for the skiing game with the paper prototype that was made.

Use scenario with paper prototype		
1. The user has to tap or make another gesture that is recognized by the head-mounted display to start the game.	2. The player can enter his name and start the game.	3. The game counts down to one.

		
<p>4. Then the game starts.</p> 	<p>5. The opponent is visible via an arrow pointing at his location, with his name and picture above it. The player has zero points now. The player can earn points by skiing through the blue gates.</p> 	<p>6. The opponent is ahead of the player. The player is getting close to the first blue gate.</p> 
<p>7. The opponent is now even further ahead. The player earns 150 points because he successfully skis through the big blue gate.</p> 	<p>8. After skiing through the gate, the gate disappeared and the points are added to the total amount of points the player has earned in the game.</p> 	<p>9. Now the player is approaching a small blue gate and an obstacle. If the player skis through the small blue gate he gets points, if he hits the obstacle he loses points.</p> 
<p>10. The player moved to the left part of the slope to be able to ski through the small blue gate.</p>	<p>11. The player successfully skis through the small blue gate and earns 250 points with it. However, he is in danger right now because the obstacle is also very close to him.</p>	<p>12. The 250 points are added to the total amount of points the player earned so far.</p>

		
<p>13. The player does not manage to avoid the obstacle and loses 200 points. Meanwhile, the distance between the player and the opponent has decreased.</p> 	<p>14. The 200 points are subtracted from the total amount of points the player had earned so far. The player continues to play the game and meets a few more gates and obstacles that should either be hit or avoided.</p> 	<p>15. Then the finish line appears. Meanwhile player 2 is falling behind.</p> 
<p>16. The player approaches the finish and manages to overtake his opponent.</p> 	<p>17. The player finishes.</p> 	<p>18. All the players are presented to a ranking that shows their placement and the points they earned. There is the option to try again and the option to quit.</p> 

The paper prototype shown in Table 5.1 was informally tested with two people. For these two informal tests, the scenario described in Table 5.1 was followed. The goal of the tests was to investigate if the game and its interactions were clear to people or not, not if people can ski with it. Therefore, the people who tested the game were not obliged to have skiing experience in order to participate in the test, which was the case for the people who eventually tested the final prototype that resulted from this project. Before the scenario started, the participants were told that it is the goal of the game to gain as many points as possible, which will help you as a player to reach the

finish before your fellow player does. The participants were also told that points can be earned by skiing through the blue gates and that points are lost by colliding with the red obstacles.

After the test, both participants reported that the game was very clear to them. It was clear for them what they had to do in order to reach the goal of the game. One of the players noted that the difference between the gates was not very clear. This player noticed that some gates were bigger than other gates, and that there was a difference in points that could be earned by skiing through them. This participant gave the advice to either use different representations for different gates, for example a difference in colour, or to not use different gates and keep the game simple. Also, both participants asked what the arrow with the picture was while they were testing the paper prototype. Only after some time they realized that this was a representation of their fellow player. One of the test participants noted that this could be the case because the arrow does not show someone who is skiing, which does not indicate that it is a fellow player who is also playing the skiing game.

Based on the results obtained from the informal tests of the paper prototype, it is decided that only one version of the gates will be present in the game, to keep it simple which game objects should be avoided and which should be aimed for. Also, it is decided that it might be better to use a three-dimensional model of a person who is skiing as a representation for the fellow player, because that indicates that there is another skier present in the game whom the player is competing with.

5.1.2 Technology Prototypes

After the product idea was concretized with a paper prototype, technology prototypes were made to explore the possibilities of the existing technologies. First, two augmented reality prototypes were made to gather insights in this specific technology and how it can be applied to this project. Second, a multiplayer prototype was made to investigate how multiplayer games are made and if this is a realistic option for this project. All technology prototypes are made in Unity3D, a game engine that allows for augmented reality development and the creation of multiplayer games.

5.1.2.1 Augmented Reality Prototypes

The first augmented reality prototype was made with Vuforia, a plugin for Unity3D that is meant for augmented reality development [41]. Vuforia can be described as marker-based augmented reality. In a Vuforia application, certain markers can be identified, which are referred to as targets. Targets can be a single image, a cuboid, a cylinder, or a three-dimensional image. Once a target is defined, it can be recognized by the physical camera of the device that the application runs on. A three-dimensional object can be added to the Unity scene, which causes the three-dimensional object to be displayed over the target once the physical camera of the device detects it. The prototype that was made with Vuforia was very simple and had minimal functionality, because it served as a way to try the technology. The prototype consisted of an image of stones as a target and a cube as the three-dimensional object that is displayed on the target image when the physical camera detects it. Figure 5.1 shows a Vuforia application that is very similar to the prototype that was made with Vuforia, except for the fact that in Figure 5.1 a teapot is used for the three-dimensional object that is displayed over the target.



Figure 5.1: An example of a Vuforia application that is very similar to the Vuforia prototype that was made.

Although Vuforia is easily implementable and functions very well, it was decided that this technology is not suitable for this project. If Vuforia would be used, markers would be needed on the revolving ski slope in order to present the user to three-dimensional objects in augmented reality. This is not desirable, as the revolving ski slope does not have markers on it and placing markers on the revolving ski slope means that the markers, and therefore the placement of the three-dimensional objects in augmented reality, would be static. This would result in a very predictable skiing game.

The second augmented reality prototype that was made can be referred to as “markerless augmented reality”, as it does not need any markers to provide augmented reality. This prototype did not use an augmented reality plugin in Unity. For the prototype, a first-person shooter game was developed based on [this](#) tutorial [42]. Figure 5.2 shows an application that is very similar to the prototype that was made based on the tutorial. The first-person shooter game used the physical camera of the device that it would run on to see the physical world. Three-dimensional objects were simply overlaying the physical world by placing them in front of the physical camera object in Unity. The application made it possible to look around and see all the three-dimensional objects by using the device’s gyroscope. The code that was written for this prototype can be found in Appendix A: Code Markerless Augmented Reality Prototype. Since this prototype has proven that markerless augmented reality is implementable and does not require any markers to be placed on the revolving ski slope, it is seen as a suitable approach for augmented reality development within this project.

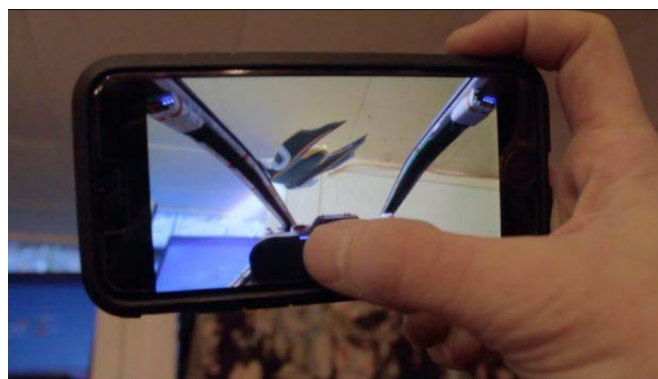


Figure 5.2: An application that is very similar to the markerless augmented reality prototype that was made.

5.1.2.2 Multiplayer Prototype

A third technology prototype was made to investigate the possibilities with regard to the development of a multiplayer game. For the multiplayer prototype, the Multiplayer Networking tutorial provided by Unity [43] was followed. This resulted in a networked game. Every player that entered the game got its own player prefab assigned. Players could shoot each other or the enemies. When a player or an enemy is shot, its health, which is shown in a health bar near its head, decreases. When the health of a player or enemy reaches zero, it dies. Figure 5.3 gives a visual representation of the multiplayer prototype.

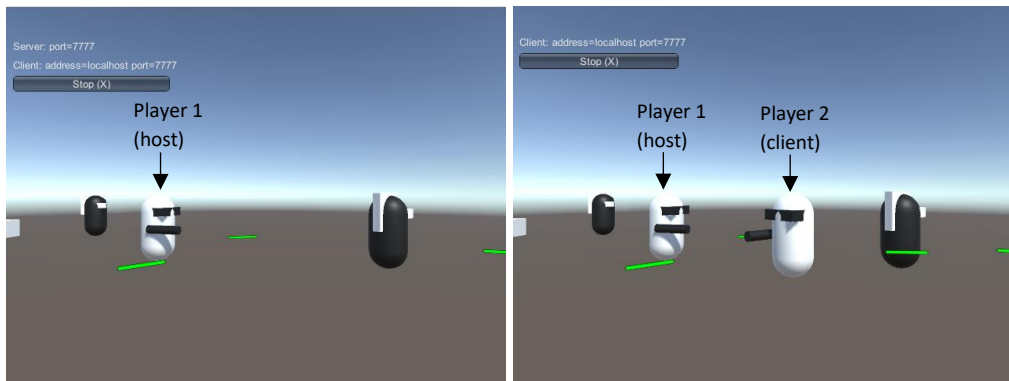


Figure 5.3: The multiplayer prototype, showing the game being played with only the host (left) and with the host and client (right).

With this prototype it was discovered how multiplayer networking can be set up in a Unity project and how multiple players can play a game with or against each other. Based on the new acquired knowledge, it was expected that making the skiing game a multiplayer game is a feasible option.

5.2 Requirements analysis

Based on the product idea that resulted from the ideation phase and the early prototypes that were made, a requirements analysis was executed. The requirements that the skiing game has to fulfil have been composed based on the MoSCoW analysis [15], which was already explained in Chapter 2. The MoSCoW analysis describes a product's *Must haves*, *Should haves*, *Could haves*, and *Won't haves*. The *Must have* criteria are the minimal criteria to be reached in the final prototype. The *Should have* criteria are important for the users to be included, but not necessary criteria for the final prototype. The *Could have* criteria are desirable, but also not necessary for the final prototype. These are the first criteria to be eliminated from the product when time does not allow to include all requirements. Finally, the *Won't have* criteria are the least critical criteria and are the criteria that cannot be included in the end product due to the limited timeline of this project. However, they are still listed in the requirements overview since they are desirable for later versions of the skiing game. Table 5.2 provides an overview of the product requirements for the skiing game and their prioritization based on the MoSCoW principle.

Table 5.2: Product requirements and prioritization for the skiing game.

Requirement		Prioritization level
The application must contain an augmented reality environment that can be displayed on a pair of augmented reality glasses.		Must
The application must offer the ability to play a game in the augmented reality environment.		Must
The application must track a user's motion in the physical world and determine the user's placement in the augmented reality environment based on it.		Must
The application must move the user forward in the augmented reality environment, without the user having to move forward in the physical world (since that will not occur on the revolving ski slope).		Must
The application must present three-dimensional objects that serve as gates in the augmented reality environment.		Must
The application must allocate the user points when he/she skies through a gate.		Must
The application should present three-dimensional objects that serve as obstacles in the augmented reality environment.		Should
The application should subtract points from the user's total number of points when he/she collides with an obstacle.		Should
The application should provide multimodal feedback to the user, which includes audio feedback, visual feedback, and if possible haptic feedback.		Should
Sub-requirements	The application should play a low tone when the user hits an obstacle.	
	The application should play a high tone when the user passes through a gate.	
	The application should assign red colours to the obstacles.	
	The application should assign blue colours to the gates.	
	The application should give a heavy buzz when the user hits an obstacle.	
	The application should give a small buzz when the user passes through a gate.	
The application should show a ranking based on the number of points per user, to show the user how well he/she is doing compared to others.		Should
The application could offer the option to play the game with multiple people (multiplayer).		Could
The application could display three-dimensional objects that are related to physical skiing environments, such as trees and snow.		Could
The application could be an asymmetrical game, in which not every player gets the same resources and powers assigned. This could solve the issue of one player being abundantly better at the game than another.		Could
The application could be a game with triangularity, in which the user is presented to high risk/high reward and low risk/low reward options.		Could
The application won't present a visual representation of a human skeleton that is skiing in front of the user and demonstrates movements that should be mimicked.		Won't
The application won't consist of several levels.		Won't
The application won't offer a training part, where people can practice their skiing skills before they participate in the game.		Won't

The application won't offer the option for the teacher to give input to the game (such as determining its difficulty, speed, slope, or duration) through a second device that has its own interface.	Won't
The application won't contain summary feedback that gives an outline of the parts of the game where the player performed well, and the parts where the player did not perform well.	Won't

5.3 Game Design

The final step of the specification phase was the specification of the game design of the skiing game. The game design of the skiing game was done based on the lenses of Schell [14]. Schell defines one hundred lenses, which all provide a different perspective through which a game can be viewed. Some of the lenses defined by Schell were accessed to make decisions for the final game design of the skiing game.

Two very essential lenses for this game are the Lens of Reward and the Lens of Punishment. Rewards are given for good behaviours, while punishments are given for bad behaviours. Essentially, this means that rewards and punishments can be seen as a form of feedback. In the skiing game, players get rewards for skiing through the gates and punishments for colliding with the obstacles. The rewards and punishments are given in the form of points, by respectively adding or subtracting points from the total number of earned points. It is decided that skiing through a gate will deliver the player as many points as colliding with an obstacle costs. An amount of one hundred points was chosen for this. This is done to make players of the skiing game aware that the mistakes they make in the game, by colliding with an obstacle, could lead to serious problems in case it would happen during actual skiing. However, it was decided that collisions with obstacles should not cost more points than the amount of points that can be earned by skiing through a gate, to prevent players from becoming demotivated.

Another important lens for the skiing game is the Lens of Essential Experience, which indicates what the essential experience of the game should be for the player. For the skiing game, the essential experience that the player should have is two-sided. On the one hand the skiing game is designed to support the ski-learning process, while on the other hand the skiing game should be fun to play and should make skiing on a revolving slope more fun. Essential to those experiences are a game that has a clear but challenging goal [3] and offers means or techniques to reach that goal [25]. That is why the goal of the game, earning as many points as possible, is so simple and requires the simple action of skiing at the right place on the slope to reach it. By having such a simple goal in the game, the game also complies with the Lens of Flow. The Lens of Flow states that the game should hold the player's focus, which is most likely to happen for games with a simple goal. Within this context, "simple" means easy to understand and clear, it does not mean that it is easy to perform. Also, playing against a human opponent is seen as fun by most people, while it also helps them in their learning process since they want to increase their skills in order to be better than their opponent. That is why it will be attempted to make the skiing game a multiplayer game.

The next lens that is important for the skiing game is the Lens of Surprise. In the serious skiing game, the aim is to surprise players by the varying placement of the gates and obstacles. That is why the locations of gates and obstacles will be generated randomly in the game. This adds chance to the game, which makes the game also comply with the Lens of Chance. Additionally, this is also a reason why a multiplayer version of the game is desirable. If the game exists in a multiplayer version, players

will get the opportunity to surprise each other. This will be done by removing gates and obstacles that are hit by one player from the game, which prevents other players from colliding with these specific gates and obstacles as well. This way, a player can prevent his/her fellow players from, for example, skiing through a gate by skiing through the gate himself/herself just before the fellow players can perform this action.

Additionally, an interesting lens that has been included in the requirements analysis that was described in section 5.2 is the Lens of Fairness. It must be noted that a multiplayer game, which is an envisioned option for the skiing game, may not always be fair. One player could clearly be better skilled at skiing than another player, simply because they differ in skiing experience. The Lens of Fairness offers a possible solution for this problem, which is the implementation of an asymmetrical game. In symmetrical games every player gets equal resources and powers assigned, whereas in asymmetrical games players get different resources and powers [14]. In the skiing game this could be done by giving players that are far ahead of their opponents, which clearly indicates they are performing better in the game, more points deduction for colliding with obstacles, and less points added to their score for successfully skiing through a gate. This way, players with different skill levels can still compete against each other.

Another lens that could be interesting for the skiing game is the Lens of Triangularity, which was also included in the requirements analysis in section 5.2. Triangularity is the situation where the player has to choose between a low risk/low reward option and a high risk/high reward option. Overall, triangularity makes games and the decisions made in games more interesting for the players. If time allows, triangularity could be added to the game by rewarding more points for gates that were difficult to reach, for example because they are placed very close to an obstacle. Obviously this means that gates that are easy to reach, because they are not near any obstacles, deliver less points.

The final lens that is important for the skiing game is the Lens of the Elemental Tetrad. According to Schell [14], the elemental tetrad contains the four basic elements that form a game, which are mechanics, story, aesthetics, and technology. The game mechanics are the procedures and rules of the game, including the goal of the game and how it can be reached. The story of the game is the collection of events that happen in the game. Aesthetics are the look and feel of the game, including audio, feel, and taste. Technology are the materials and interactions that make the game possible to exist. According to Schell, all four elements of the elemental tetrad are equally important to the design of the game. Figure 5.4 gives a depiction of the elemental tetrad. As is shown in Figure 5.4, aesthetics are said to be the most visible to the players, technology the least visible, and mechanics and story are in between those two.

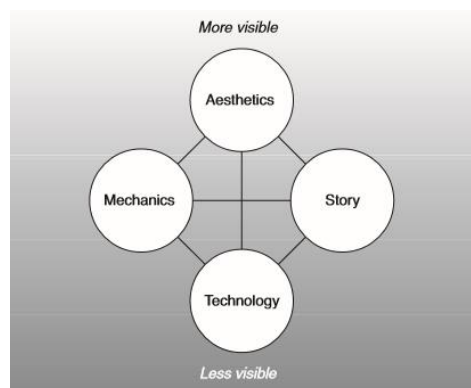


Figure 5.4: The elemental tetrad, containing the four basic elements that form a game.

The elemental tetrad of the skiing game is not in perfect harmony. However, this is not seen as problematic. Three elements are clearly present in the game, which are aesthetics, mechanics, and technology. Aesthetics are attempted to be achieved by the three-dimensional modeling and colouring of the gates and obstacles, and the accompanying sounds that are played as a form of feedback when the player collides with either one of them. The mechanics of the game are present in its clear procedures, rules, and goal, which can be summarized as follows: *try to gain as many points as possible by skiing through the blue gates and avoiding the red obstacles*. The technology of the game, enabling the game to exist, is reached through the use of the Unity 3D Game Engine, which allows for augmented reality development. The element that is missing, however, is the story. The game does not contain a story. Nonetheless, this is not seen as a problem, since the lack of the story adds to the simplicity of the skiing game, which is expected to be desired by people who want to play a game for the sole purpose of skiing.

Chapter 6 – Product Realisation

This chapter describes the realisation process of the skiing game that was specified in the previous chapter. Product realisation was done based on the requirements that were set and the game design that was explained in the previous chapter. First, the architecture of the system will be discussed in section 6.1. The system architecture described here served as a basis for the implementation process that was carried out in this project, which will be discussed in section 6.2.

6.1 System Architecture

This section describes the system architecture of the skiing game, which served as a basis for the implementation process described in section 6.2. The system basically consists of three physical components, which are the player, the augmented reality glasses (head-mounted display), and the game application. Figure 6.1 shows a schematic representation of the system architecture, describing its inputs and outputs and the communication between the different physical components. Since it is decided to make the game a multiplayer game, two players are shown in Figure 6.1 to indicate the principle.

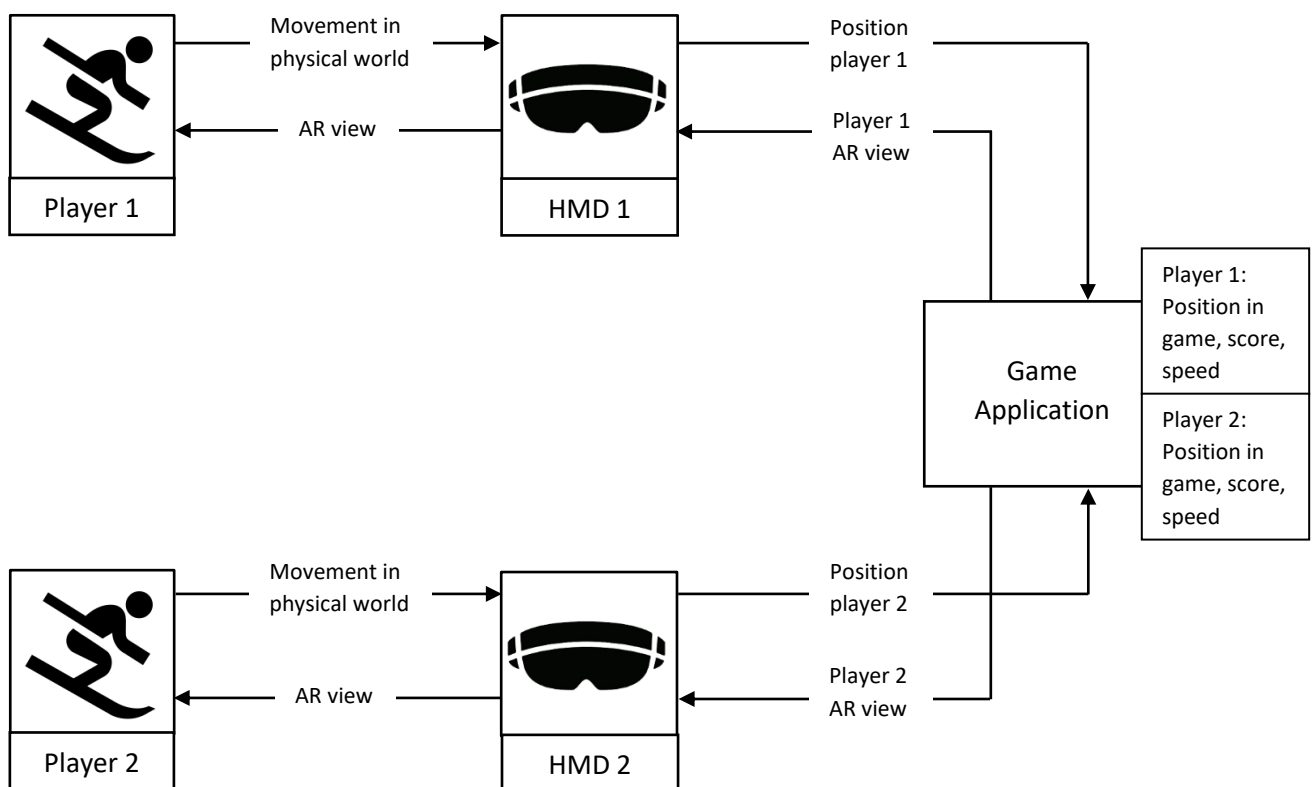


Figure 6.1: Simple system architecture of the skiing game.

Both players influence the game by their movements in the physical world. The head-mounted display (HMD) of each player communicates their movements in the physical world to the game

application. The game application determines for every player their position in the game, their scored points, and their speed. This information is used to build an augmented reality view for every individual player, which is communicated to their head-mounted displays. Through the head-mounted displays, the players are presented to the augmented reality view and they can try to influence it with their movements in the physical world. This process continues until the game has ended.

6.2 Implementation

This section describes the implementation process of the specified product. Implementation was done based on the system architecture that was described in section 6.1. The implementation process was a thorough process that took four to five weeks time in total. The goal of this section is to describe the implementation process per implementation activity. It must be noted that most of the implementation activities were spread out over the course of implementation and were not completed at once. A synchronous overview of which steps were taken in the process can be found in Appendix B: Realisation Timeline.

6.2.1 Software

The software that was chosen to make the final prototype of the skiing game with is Unity3D. Since augmented reality uses three-dimensional objects to overlay the physical world, Unity3D can be seen as a suitable engine for creating this. Also, there are several augmented reality plugins available for Unity, such as Vuforia [41], Tango [44], and ARToolKit [45], which suggests that the Unity engine is a proper tool for the development of augmented reality applications. The Unity game engine is often used by game designers and developers, which made it a community that has a great quantity of documentation. The community and documentation make it easier to learn something in the system and to get extra support from fellow game developers when needed. Besides that, Unity is a multiplatform engine [46], meaning that development in Unity can be targeted at several devices. Some of the most frequently used platforms that Unity can be used to build for are iOS, Android, and Windows. Multiplatform development is an important and beneficial aspect to this research project, as it provides the possibility to develop for almost all augmented reality devices that are in existence, which makes the outcome of this project more relevant and useful in further development. Apart from support for augmented reality development in Unity, multiplatform support is the main reason why Unity3D was the chosen software to build the skiing game prototype with.

Unity3D provides the option to make scenes, which can be seen as a level in the game. Game objects can be added to the scenes and can be given properties and behaviours in the editor and through scripts [47]. Unity provides the option to write scripts in JavaScript or C#, the latter being the preferred choice nowadays.

6.2.2 Development for Augmented Reality

The settings that one should work with in Unity to develop for augmented reality differ per augmented reality device that the development is target at. When a new scene is created in Unity, there is by default a main camera object present in the scene. The main camera is the first enabled camera, which means that what is seen through the main camera in the Unity editor is what will also be presented to the player of the game once the game is exported to a certain device. This is important information for development in augmented reality, since this means that the main camera

object determines what the player can see. The main camera provides the view of the player, so the position of the main camera is also the position that the player of the augmented reality game will feel he/she is at while playing the game.

When developing for smartphones, the physical camera of the smartphone is used as a viewing window to see the physical world. In Unity, this is done by placing a plane game object in front of the Unity camera (in other words, placing a plane in front of what the player is seeing). The plane gets the texture of the physical camera assigned, which means that what can be seen through the camera of the device is displayed on the plane. Doing this on a laptop in the Unity editor shows what can be seen through the webcam of the laptop in the Unity scene. When this is exported to a smartphone, the physical camera of the smartphone will be used and the player can see the real world in the application. In an augmented reality application, the three-dimensional objects are placed over the view of the camera that can be seen when running the application. Figure 6.2 gives a visual representation of an augmented reality application on a smartphone. When using a smartphone as the augmented reality device, the smartphone can be placed in a Google Cardboard or a similar product, as long as the area in front of the camera is free.



Figure 6.2: An augmented reality application on a smartphone.

Development for augmented reality glasses differs from what is described in the section above. Some augmented reality glasses have a built-in camera, such as the Epson BT-300 [48]. The same application as the one described above can be used on these glasses. However, the representation of the physical world looks strange because the re-projected view of the physical camera is overlaying the direct see-through view of the augmented reality device. Besides that, it is not needed to use the physical camera since the glasses already provide a see-through to see the physical world. Because of this, and because not all augmented reality glasses have a built-in camera, it was decided to not use the physical camera of the device in the application. Instead, Unity provides the option to turn the background colour of the environment black, which means that every three-dimensional object in the application will be seen in front of a black background. The black Unity background looks transparent on most augmented reality glasses. When three-dimensional objects are present in the scene, the person wearing the augmented reality glasses sees the three-dimensional objects overlaying the physical world.

6.2.3 Placing Game Objects

A first crucial step in the development process of the skiing game was the creation and placement of

game objects in the scene. The most important game objects that had to be added to the game were gates and obstacles, which cause a gain or loss in points respectively when the player hits them. Simple three-dimensional objects, as can be seen in Figure 6.3, were created in the animation and modeling software program Maya and imported in the Unity project of the game. The gate is coloured blue, to indicate that it has a positive effect on the player. The obstacle got red assigned as colour, to indicate that it has a negative effect on the player. This is done because people often associate blue with health, while red is associated with intensity and anger [40].

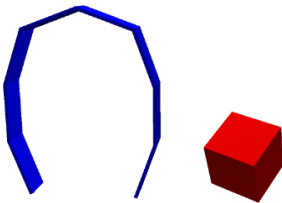


Figure 6.3: The three-dimensional objects created in the Maya software and used in the game as gates (left) and obstacles (right).

A second choice that had to be made here was about the movement in the game. The player of the game is not substantially moving forward in the physical world, as the revolving ski slope prevents the skier from actually going forward. When thinking of the situation on the revolving ski slope to be like a coordinate system, the x-axis, y-axis, and z-axis can be identified as is shown in Figure 6.4. The skier will mostly move along the x-axis. These physical movements along the x-axis influence the game, allowing the player to hit or avoid game objects. The skier has the possibility to slightly move along the z-axis. However, it is decided that the physical movement along the z-axis will be ignored inside the game, since it does not correspond to the player’s feeling of going forward while skiing on the revolving ski slope.

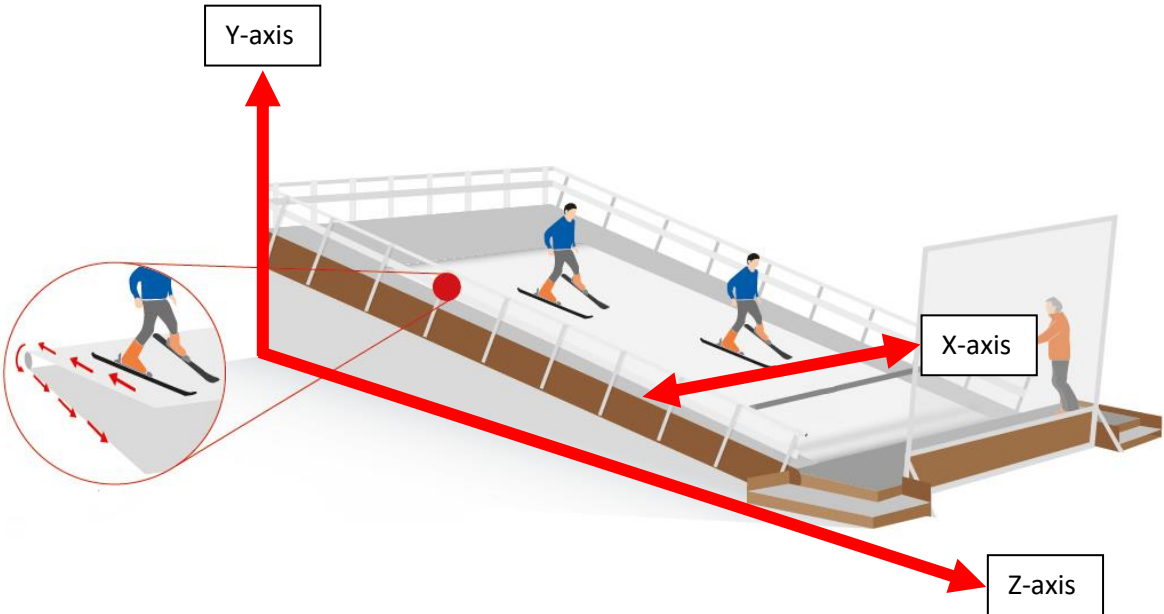


Figure 6.4: The situation on the revolving ski slope with the corresponding axes.

The feeling of going forward has to be created in the game, allowing the player to actually see three-dimensional objects coming his/her way while skiing on the revolving ski slope. In the game this can be done in two ways. One way is to give the player game object a certain speed along the z-axis in the game, making the player move forward in the direction of the placed gates and obstacles. Another way to do this is to give the game objects a certain speed along the z-axis in the direction of the player and keep the player game object at the same place along the z-axis, which means that the game objects move towards the player. The first option was chosen, as it allows different players to move through the game at different speeds, which is used in the multiplayer version of the game. Why this was needed in the multiplayer version of the game is explained under section 6.2.6 Networking.

To prevent the game causing a memory overload on the targeted device that the game will be played on, it was decided to not create all the game objects at once. Instead, it was decided to let the game create game objects during runtime. This means that the game creates a new game object every specified time period. For the same reason it was decided that the game also destroys an existing game object every specified time period. Game objects can only be destroyed when they are out of reach and sight of the player.

Based on the choices that are described above, it was decided how game objects should be placed in the game. The decisions made are as follows:

- Every specified time period a game object must be created.
- Every game object has to be destroyed when it is in existence for a specified time period.
- Game objects must be placed along the z-axis, in front of the player. The placement along the x-axis can be random, as long as it is within the bounds of the width of the physical ski-slope. The placement along the y-axis must correspond with the location of the slope, so that gates and obstacles are placed on the surface of the slope.

In the code this was implemented by taking the starting position of the player as a starting point for the game objects. A global variable was made to specify the distance from the player at which game objects can be created. Every frame it is checked if the z-position of the last created game object is smaller than the specified distance to the player plus the z-position of the player. This check ensures that game objects will only be created within a set distance from the player, preventing all game objects to be created at once. Creation of the game objects was done using Unity's instantiate function, which makes a clone of a specified game object every time it is called. The position that the game object should be created at is given as a parameter to the instantiate function. For the x-position, a random number between two specified numbers is chosen every time a game object is instantiated. The specified numbers can be manually adjusted to the width of the slope. For the z-position the z-position of the last instantiated game object plus a specified distance between game objects was taken. The choice for the y-position is made based on the slope that the game will be played at, which is explained under 6.2.5 Adding a Slope. Once a game object is created, it will be destroyed using Unity's destroy function, which takes the game object to be destroyed and the time after which it should be destroyed as parameters. The code explained above can be found in Appendix C: Spawner Script.

6.2.4 Detecting Collisions

Once the player hits one of the game objects, points must be added to or subtracted from the total number of gained points. Collisions between the player and the game objects are detected by giving

both the player and the game objects a collision box in Unity. A collision is detected when the collision boxes of two different game objects hit each other. Collision detection is done using Unity's OnTriggerEnter function on the player game object, which can detect collisions between itself and other game objects. The other game objects carry tags based on what they are. The gate game objects carry the tag "arc", and the obstacle game objects carry the tag "obstacle". When the OnTriggerEnter function detects a collision, the tag of the game object that collided with the player game object is checked. If the game object is tagged "arc", one hundred points are added to the player's total number of points. In the other case, if the game object is tagged "obstacle", one hundred points are subtracted from the player's total number of points and the player's movement is slowed down for five seconds. The code explained above can be found in Appendix D: Camera Collider Script.

6.2.5 Adding a Slope

Because the game will be played on a ski slope, a slope had to be added to the game to ensure that the game objects are placed on the slope in a naturally appearing way. Figure 6.5 gives a visual representation of how the game objects should be placed along the slope and approach the player in this case. It must be noted that the ski slope and the skier are present in the physical world, and that the gates and obstacles are only present in the virtual world which is presented to the player on the augmented reality glasses.

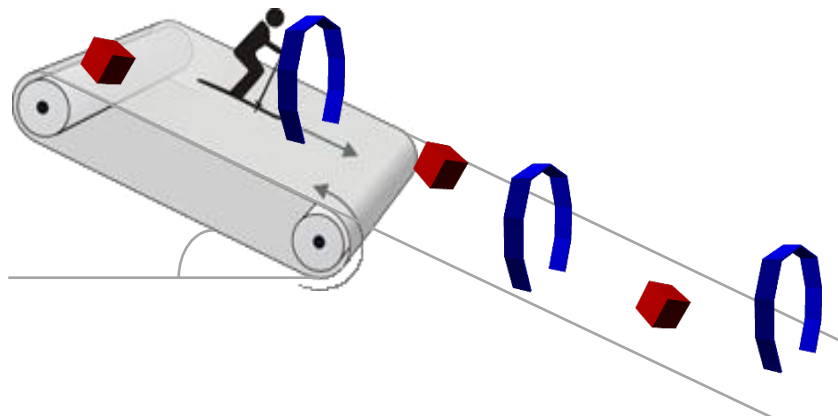


Figure 6.5: Placement of the game objects along the ski slope.

The most important part of adding a slope to the game is placing the game objects along the slope. To do so, the y-position of the game objects is calculated based on the angle of inclination and the z-position of the game object. A global variable was made to store the angle of inclination, which can manually be inserted in the game. The angle of inclination can be inserted in degrees, and is converted to radians. The z-position of the game objects that will be placed in the game is determined based on the z-position of the previous game object of the same kind plus a set distance between game objects, as was described in section 6.2.3 Placing Game Objects. The y-position of the next game object to be created is calculated by subtracting the tangent of the radians of inclination multiplied with the distance between two game objects (which is a value on the z-axis) from the y-

position of the last created game object. Figure 6.6 gives a representation of this for the obstacle game objects. The code explained above can be found in Appendix C: Spawner Script.

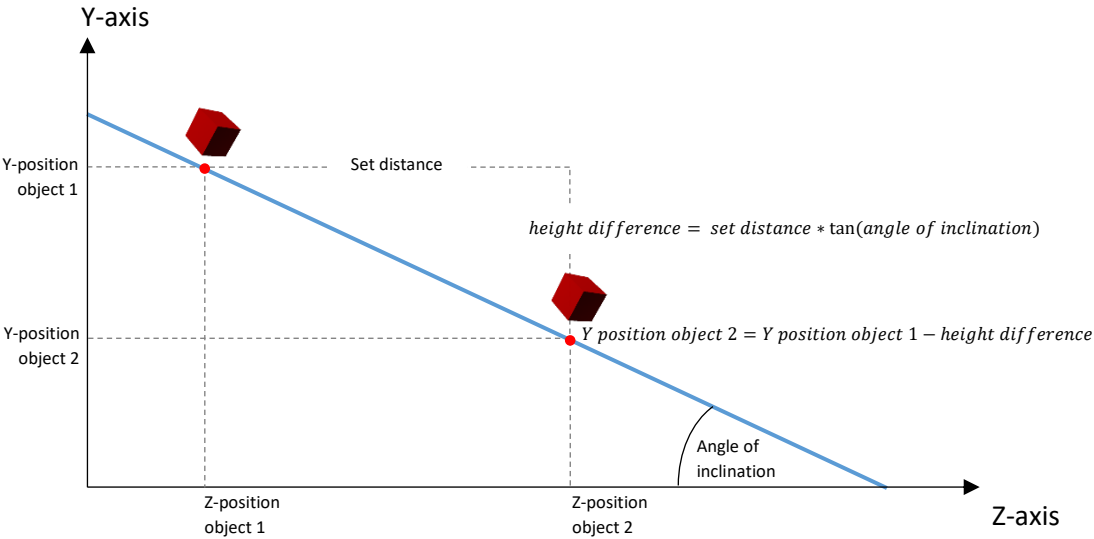


Figure 6.6: Placement of the game objects along the slope and the associated calculations.

6.2.6 Networking

To make the skiing game a multiplayer game, Unity Networking was used. Unity offers its own Network Manager as a tool to implement networking in a game. To use this tool, the Network Manager component must be added to an empty game object, preferably called “network manager” to indicate its function, in the scene. The Unity Network Manager manages the network state of a multiplayer game [49]. Using the Network Manager, a mode can be chosen for the game to run in. The modes that can be chosen are client, server, or host. In networking terms, the client requests a service and the server provides a service [50], which means that the client is dependent on the server and cannot do anything without the server. A typical model for the client-server relation is depicted in Figure 6.7. A host is a server and client at the same time. In a networked Unity project, the server and the clients are executing the same code at the same game objects at the same time [49].

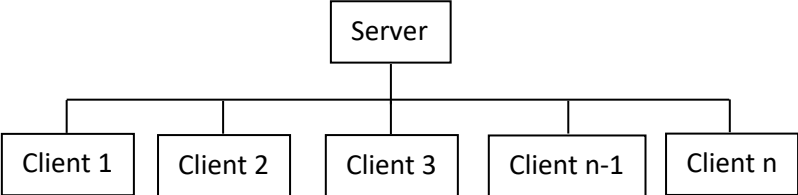


Figure 6.7: The client-server model.

Running a multiplayer game basically means that the player game object is created as many times as there are clients in the game, as every client needs a player game object to be able to control the game. The player game object must get a network identity assigned with local player authority. A network identity makes the networking system aware of the game object and the local player authority ensures that the game object can be controlled by the client that owns it [51]. Since a player comes with its own camera, which is needed to provide the player in augmented reality with its own view of the game, a multiplayer version of the game asks for identifying which camera belongs to which player. By default, Unity takes the most recently created camera to be the camera that the player is looking through. This means that if two or more players are present in the game, all of them will see the view of the player that entered the game most recently. This problem was solved by checking if a player is the local player, which is the player that the client has authority over. For every client, the camera of the players that are not the local player are disabled. This part of the code can be found in Appendix E: Player Manager Script. The same principle is applied to the score of each individual player, so that every player will only influence its own score. That part of the code can be found in Appendix D: Camera Collider Script.

Besides networking the player game object, a networked game also asks for networking other game objects. Networking the game objects ensures that all players see the same game objects at the same location in the game. It was decided that all players must be presented to the same set up of game objects to make the game more fair and to enhance the multiplayer aspect of the game. This principle makes the game more fair because one player cannot have an advantage over the other if they are presented to the exact same set up. Besides that, this principle is expected to enhance the multiplayer aspect of the game because a player becomes more aware of his/her opponent if the opponent is trying to reach or avoid the same game objects. Also, when one player hits or avoids a game object, the game object is destroyed, preventing another player to hit or avoid it as well. Creating the same game objects at the same locations on all clients of the game is done using Unity's spawn function. Every game object is given a network identity and added to the list of "Registered Spawnable Prefabs" in the Network Manager. Once this is done, game objects can be spawned on the server, meaning that the game objects will be created on all clients connected to the server [52]. Spawning game objects is done at the positions that were explained under 6.2.3 Placing Game Objects. The code explained above can be found in Appendix C: Spawner Script.

Having two or more players in the scene raised another problem concerning how players can be able to see each other. To make the game a fair game, players start at the same z-position and get the same speed assigned so that they have to travel the same distance to the end of the game. This causes a problem, because if every player stays at the same z-position as his/her opponents, they will not be able to see one another. If players cannot see each other, they will not notice that they are playing a multiplayer game. Two options were considered to solve this problem. The first option was to add another screen in the game, where a player can see a map that shows his/her own location and the location of the opponents. However, this options was not considered to be a good solution as it is expected that adding an extra screen to the game will distract the player from the goal of the game, which is to avoid the obstacles and ski through the gates, and cause an overload of information on the player. The second option was to apply changes in speed between different players, based on how well they are performing in the game. A difference in speed between players will cause one player to overtake the other, allowing the player that falls behind to see the opponent in front of him/her. The second option was considered fair and feasible, and it was implemented by slightly decreasing the speed of the player for five seconds when an obstacle is hit. The speed is only decreased for a small amount of time and then reset to the original speed to give a player that fell

behind the option to overtake the player in front of him. The code explained above can be found in Appendix D: Camera Collider Script.

6.2.7 Hardware

The hardware device that was chosen to be used for this project is the Microsoft HoloLens. The main reason why it was decided to use the HoloLens in this project is because the HoloLens is capable of spatial mapping. The HoloLens uses four environment understanding cameras, a depth camera, an ambient light sensor, and a 2MP photo/HD video camera to obtain information about its surroundings [53], which is used to build a three-dimensional model of it. As the HoloLens is capable of building a three-dimensional model of its surroundings, it is also capable of knowing its own position in the environment. This means that, when wearing the HoloLens, one can simply move through an augmented reality environment by moving in the physical world. Stated differently, no extra tracking of the skiers position is needed when the HoloLens is used on the ski slope, as it will know its own position on the ski slope and update the augmented reality environment that is displayed on it accordingly. If another augmented reality head-mounted display would be chosen to do a similar project, the problem of tracking needs to be solved.

Using the HoloLens caused some problems in the development process of the skiing game. The first and most severe problem was the movement of the player through the augmented reality environment. As is explained above, the HoloLens is capable of keeping track of a user's position. It continuously updates the position of the user relative to the user's starting position in the mapped physical environment. As is explained in section 6.2.3 Placing Game Objects, it was decided to give the player of the game a certain speed along the z-axis so that he/she can move through the augmented reality environment without having to move forward on the ski slope. In the Unity project, the Unity camera provides the view of the player and, in case of the HoloLens, decides the movements of the player. However, giving the Unity camera a certain speed along the z-axis did not deliver the desired result, as the player did not move in the z-direction automatically. The Unity camera continuously updated its position relative to the starting position of the application in the physical world, which means that one could only move forward in the game by moving in the physical world. This problem was solved by giving the Unity camera a parent game object, which got the Player Manager script assigned that makes it move along the z-axis at a certain speed. However, collisions still had to be detected on the Unity camera object, which was done by assigning a collider script to the Unity camera object. Collisions must be detected on the Unity camera object because the parent of the camera is not capable of moving along the x-axis according to movements with the HoloLens in the physical world. This makes sense, because the parent of the Unity camera is not the Unity camera itself, and therefore does not get influenced by the movements that the HoloLens tracks in the physical world (which is also the reason why the parent is capable of moving forward along the z-axis, and the Unity camera object is not). The Unity camera object, however, is still assigning itself positions based on what the HoloLens tracks in the physical world, which means that movement along the x-axis is applied on the Unity camera and not on its parent game object. Therefore, the parent game object is not capable of colliding with obstacles or gates, while the camera is. The principle described above is visualised in Figure 6.8. As can be seen in Figure 6.8, the camera can move based on the movement that is detected by the HoloLens in the physical world. The parent moves based the code in the Player Manager script that is attached to it, and drags the camera game object along with it. In the limited time that was allocated for the realisation of the skiing game, no solution was found to assign the Unity camera's x-position to its parent.

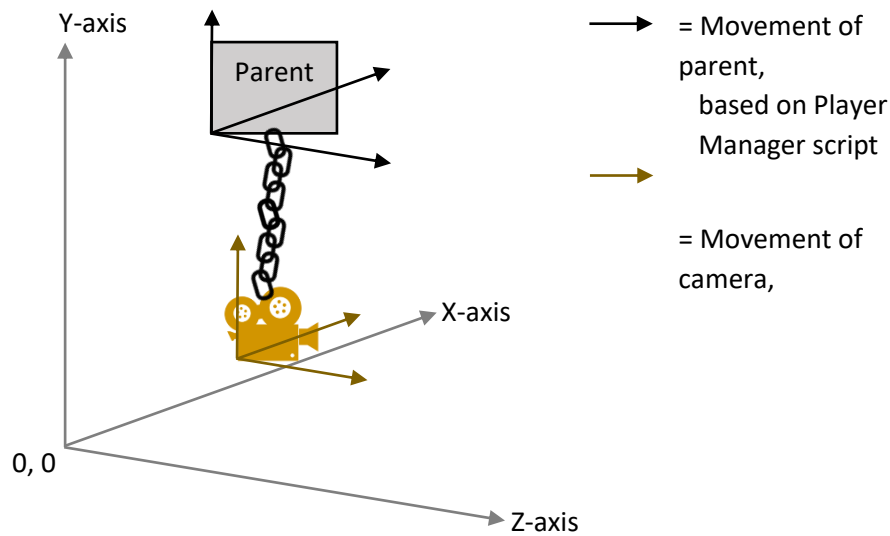


Figure 6.8: The parent-child relationship for the parent of the Unity camera and the Unity camera.

A second problem that was found during implementation on the Hololens is the positioning of the canvas in Unity. The canvas is a user interface (UI) element that is the parent of all other UI elements. It provides the space that the UI elements are drawn on. Unity offers two different modes for rendering the canvas, which are screen space and world space. With screen space rendering, the canvas will be rendered as an overlay on the screen. This means that wherever the Unity camera is located in the scene, the canvas will always be visible since it is overlaying the screen. World space rendering means that the canvas is present somewhere in the scene as a three-dimensional object, which means that it is only visible if the camera is placed right in front of it. The Hololens only offers the option to display the canvas if it is in world space. Because of this, one cannot simply set the canvas to screen space and expect to see all the UI elements. Instead, the canvas had to be set to world space and be placed in front of the Unity camera. Since the Unity camera is moving in the game, as is explained in the section above, the canvas has to move with it. This is done by assigning the Unity camera's position and rotation to the canvas, in the Canvas Position script which can be found in Appendix F: Canvas Position Script.

A third problem that was discovered while developing for the Hololens is the processing power of the Hololens. When the Hololens has to render a number of game objects at one time, it becomes slow and updates are not at the desired frame rate anymore. The Hololens comes with an Intel 32 bit processor and has 2GB RAM and 64GB Flash [53]. The problem was solved by limiting the number of game objects in the scene, which was done in the Spawner script that can be found in Appendix C: Spawner Script. The Spawner scripts handles this problem by only creating game objects within a certain distance of the player and destroying the game objects after a set time

6.2.8 Scripts and Structure

In the above sections most of the scripts that were used in this project are already mentioned. This section aims to provide an overview of the used scripts in the project and the communication

between the different scripts. Table 6.1 provides an overview of the scripts that are used in the project and their main functionality.

Table 6.1: Overview of the scripts and their functionality

Script	Functionality
Camera Collider script Appendix D	Detecting collisions between the player, who is wearing the camera, and other game objects. When a collision is detected, the score and speed of the player are adjusted accordingly.
Canvas Position script Appendix F	Positions the canvas in front of the player so that the player can see the UI elements while playing the game.
Floating script Appendix G	Makes the finish float.
Game Manager script Appendix H	Sets the UI elements of the game, which are the number of points during the game and the score overview at the end of the game.
Network Manager script Appendix I	This is not the Network Manager script by Unity, but a new script that was made for this project. This script calls the startHost function on the application on the Hololens, which starts the game as a host. This is done to prevent the player from having to navigate through a graphical user interface (GUI) in order to start the game.
Obstacle script Appendix J	Makes the obstacles spin to get the player's attention.
Player Manager script Appendix E	Decides which camera belongs to the player. Makes the player move at a certain speed along the z-axis on a slope with a certain angle of inclination.
Ski Rotation script Appendix K	Rotates the skies of the player game object according to the slope.
Spawner script Appendix C	Initiates and spawns all game objects, which are gates, obstacles, trees, and snow.

Some of the scripts communicate values to other scripts. An overview of the communication between scripts is presented in Figure 6.9. As can be seen, the Floating script, Network Manager script, and Obstacle script do not send or receive values from other scripts. All the other scripts do. When the game starts, the Player Manager script sends the initial speed of the player to the Camera Collider script. The Camera Collider script detects collisions and determines the new speed based on collisions with obstacles. The Camera Collider script sends the updated speed value to the Player Manager script, where the value is applied to the actual speed of the player. The Camera Collider script also communicates the score of the player to the Player Manager script, which makes sure that all players' scores are updated. Furthermore, the Camera Collider script sends the score, a boolean that determines the end of the game, and a boolean that determines that the score overview should be made visible to the Game Manager script, which handles the display of all the associated information in the UI elements. The Player Manager script sends the value of the angle of the slope to three other scripts, which are the Canvas Position script, the Ski Rotation script, and the Spawner script. All of these scripts use the value of the angle of the slope to determine the rotation and placement of, respectively, the canvas, the skies of the player, and the spawned game objects. To determine the position of the canvas, the Player Manager script also send the position values of the camera to the Canvas Position script, which uses the values to place the canvas in front of the camera. For the placement of the spawned game objects the Player Manager script also sends the z-

position of the player and the end position that the player is moving towards to the Spawner script. The Spawner script can use these values to determine when and where a game object should be spawned and when to stop spawning.

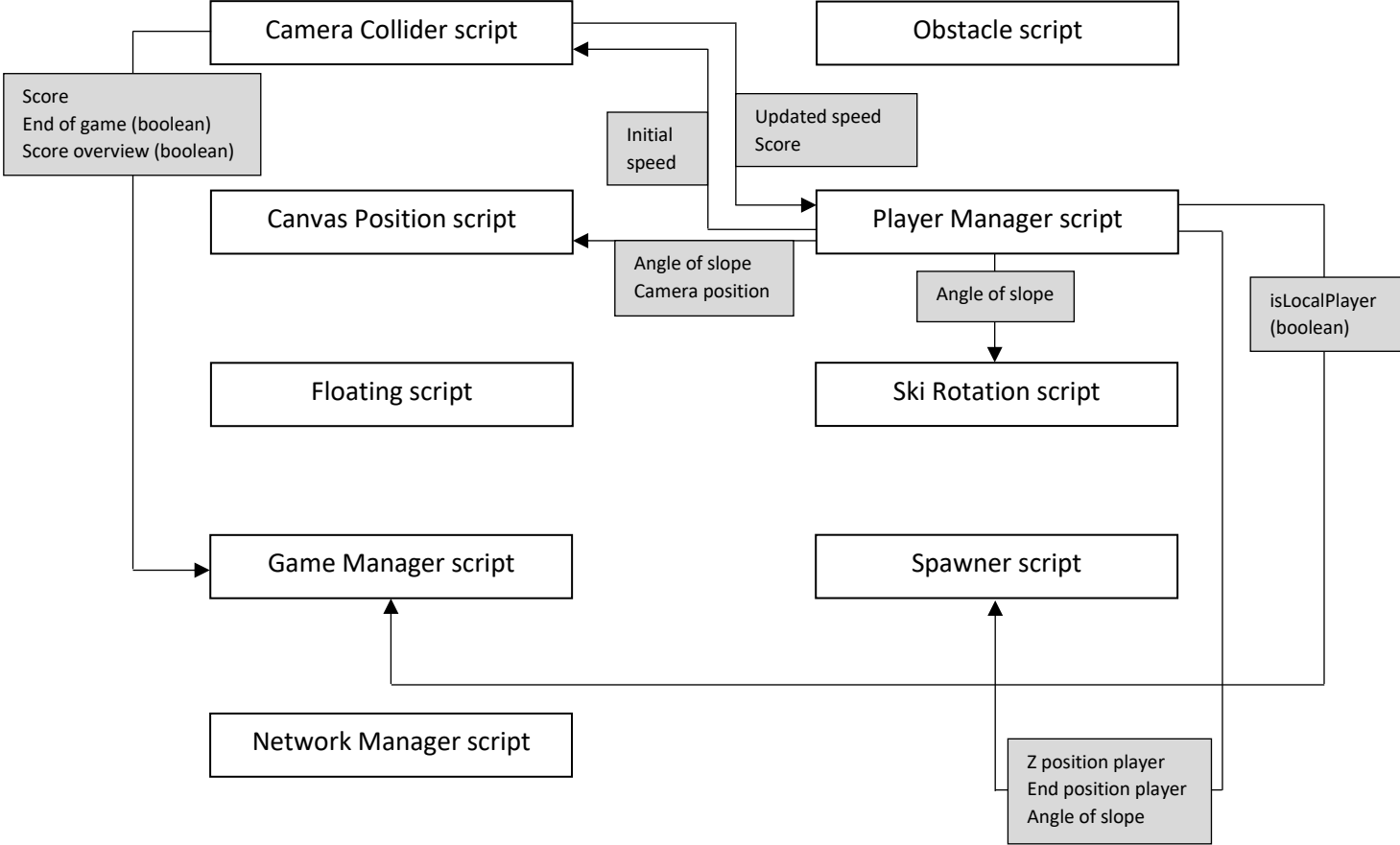


Figure 6.9: A schematic overview of the communication between the scripts of the project.

6.2.9 Result

All the sections that are described above explained the different steps that had to be taken to implement the skiing game. Figure 6.10 shows what the final result looks like from the perspective of the player. It must be noted that Figure 6.10 shows the game while it is running on a PC. On a Hololens or similar pair of augmented reality glasses the black background will be transparent, enabling the player to see his/her normal surroundings underneath the three-dimensional objects. Figure 6.11A and Figure 6.11B show the placement of the player and the other game objects while the game is being played in the Unity editor.

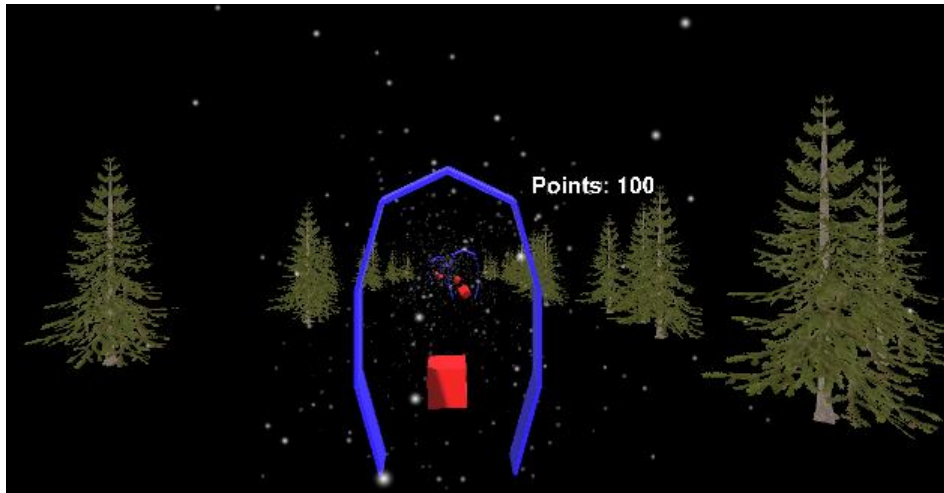


Figure 6.10: The game from the perspective of the player.



Figure 6.11A: Placement of the player and the game objects while the game is being played in the Unity editor.

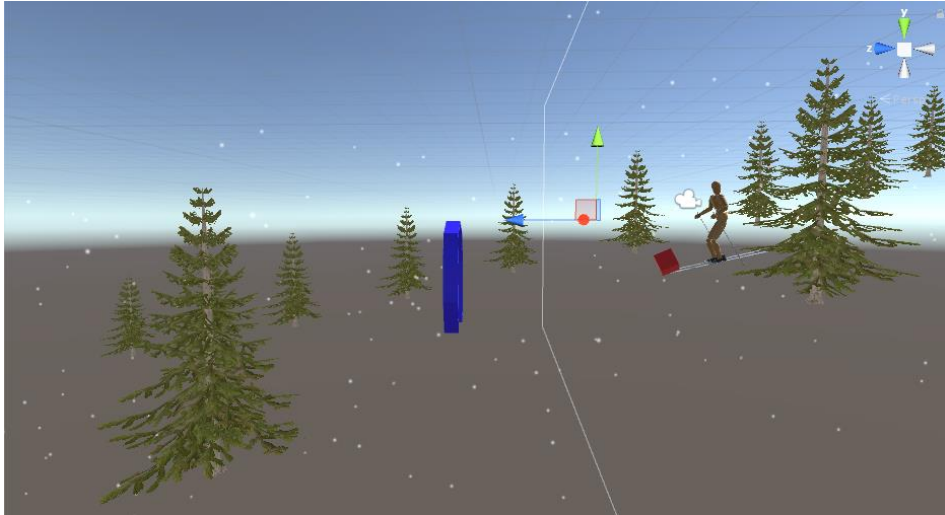


Figure 6.11B: Placement of the player and the game objects while the game is being played in the Unity editor.

Chapter 7 – Evaluation

This chapter contains the results of the evaluation phase of this project. The evaluation phase was two-fold, as it consisted of a functional evaluation and a user evaluation of the product that resulted from the realisation phase. Section 7.1 contains the functional evaluation of the skiing game, where it is determined if the final prototype meets its functional requirements. Section 7.2 contains the user evaluation of the skiing game, which was executed in two rounds. In the first round, users could interact with the prototype and give their opinions about it. Based on the results from the first round, an updated version of the prototype was made. The updated version was tested in a second round of user tests, which again allowed the participants to interact with the product and giving their opinions about it. Finally, in section 7.3 a conclusion will be draw upon the test results that are presented in this chapter.

7.1 Functional evaluation

This section contains the functional evaluation of the skiing game that resulted from the realisation phase that is described in Chapter 6. Functional evaluation was done based on the functional requirements that were determined in the specification phase that is described in Chapter 5. For every requirement, except for the *won't have* requirements, it is determined if the requirement is met in the final product that resulted from the realisation phase. Table 7.1 provides an overview of the product requirements and their assessment.

Table 7.1: Evaluation of the product requirements.

Requirements	Prioritization level	Fulfilled
The application must contain an augmented reality environment that can be displayed on a pair of augmented reality glasses.	Must	Yes
The application must offer the ability to play a game in the augmented reality environment.	Must	Yes
The application must track a user's motion in the physical world and determine the user's placement in the augmented reality environment based on it.	Must	Yes
The application must move the user forward in the augmented reality environment, without the user having to move forward in the physical world (since that will not occur on the revolving ski slope).	Must	Yes
The application must present three-dimensional objects that serve as gates in the augmented reality environment.	Must	Yes
The application must allocate the user points when he/she skies through a gate.	Must	Yes
The application should present three-dimensional objects that serve as obstacles in the augmented reality environment.	Should	Yes
The application should subtract points from the user's total number of points when he/she collides with an obstacle.	Should	Yes

The application should provide multimodal feedback to the user, which includes audio feedback, visual feedback, and if possible haptic feedback.		Should	Partially
Sub-requirements	The application should play a low tone when the user hits an obstacle.		Yes
	The application should play a high tone when the user passes through a gate.		Yes
	The application should assign red colours to the obstacles.		Yes
	The application should assign blue colours to the gates.		Yes
	The application should give a heavy buzz when the user hits an obstacle.		No
	The application should give a small buzz when the user passes through a gate.		No
The application should show a ranking based on the number of points per user, to show the user how well he/she is doing compared to others.		Should	Partially
The application could offer the option to play the game with multiple people (multiplayer).		Could	Yes
The application could display three-dimensional objects that are related to physical skiing environments, such as trees and snow.		Could	Yes
The application could be an asymmetrical game, in which not every player gets the same resources and powers assigned. This could solve the issue of one player being abundantly better at the game than another.		Could	No
The application could be a game with triangularity, in which the user is presented to high risk/high reward and low risk/low reward options.		Could	No
The application won't present a visual representation of a human skeleton that is skiing in front of the user and demonstrates movements that should be mimicked.		Won't	
The application won't have several levels.		Won't	
The application won't offer a training part, where people can practice their skiing skills before they participate in the game.		Won't	
The application won't offer the option for the teacher to give input to the game (such as determining its difficulty, speed, slope, or duration) through a second device that has its own interface.		Won't	
The application won't contain summary feedback that give an outline of the parts of the game where the player performed well, and the parts where the player did not perform so well.		Won't	

Based on the result of the functional evaluation that is presented in Table 7.1 it can be concluded that all *must have*-requirements of the product are fulfilled for the prototype that resulted from this project. Additionally, most of the *should have*-requirements were fulfilled as well. The *should have*-requirement of haptic feedback was only partially fulfilled, since no buzz was added to the game. The Hololens did not provide a way to do this except for making the head-mounted display itself buzz, which is seen as an undesirable thing to do during skiing. Another *should have*-requirement that was only fulfilled partially was the requirement of displaying an overall ranking containing the player's points and the points of his/her opponents. A ranking was made, but only two players were included

in it and the information it provided was not right. The information displayed in the ranking was the score of the player himself, and an imaginary score of the opponent which was not the actual score of the opponent. Finally, two of the *could have*-requirements were not met. The requirements that were not met are the requirement of the game being an asymmetrical game, and having triangularity in the game. These features were not added to the game due to the limited time that was allocated to this project.

7.2 User evaluation

This section describes the user evaluation that was carried out through a number of user tests. There were two rounds of user tests. In the first round the game that resulted from the realisation process, as described in Chapter 6, was tested by different users. Based on the results from these tests, a small change was made to the design of the game. The second round of user tests was executed to evaluate the changed game.

7.2.1 First Round of User Tests

This section describes the first round of user tests that were executed with the game that was the end result from Chapter 6. First a description of the test participants will be given. This will be followed by an explanation of the test setup. Finally, the results of the first round of user tests will be given.

7.2.1.1 Participants

The first round of user tests was executed with a total of 17 test participants, of whom 10 were male and 7 were female. Among the test participants were 15 students and 2 employees of the University of Twente. Test participants had to indicate their level of skiing experience. For indicating this, they could choose from “not so experienced”, “moderately experienced”, “experienced”, and “very experienced”. Out of the 17 test participants, 5 had moderate skiing experience, 6 indicated to be experienced skiers, and 6 indicated to be very experienced skiers. Figure 7.1 shows the mentioned characteristics of the group of test participants in pie charts.

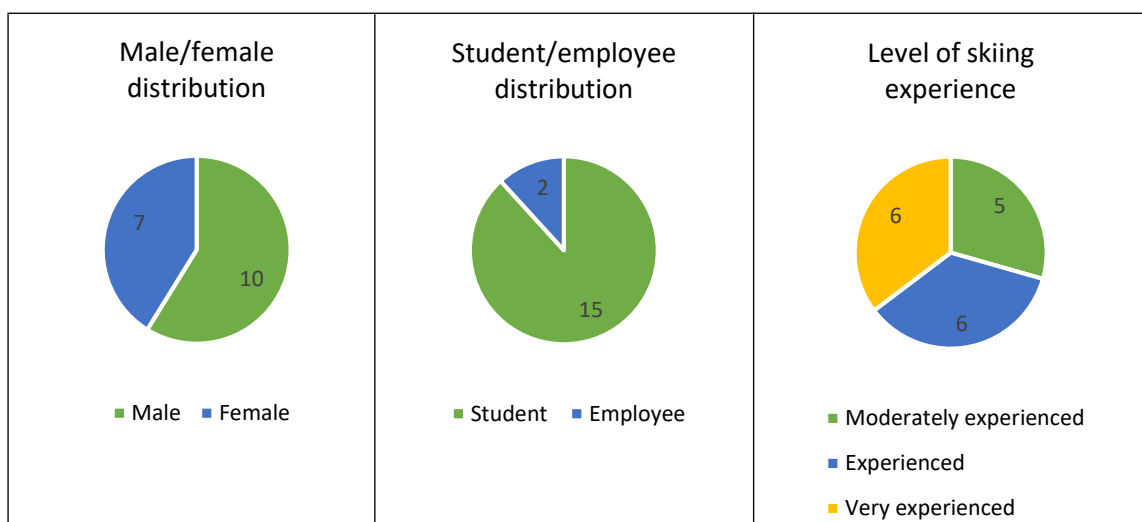


Figure 7.1: Pie charts containing the characteristics of the test participants of the first round of user tests.

7.2.1.2 Test Setup

Every test had the same setup and followed the same steps. During a test session the participant was asked to ski on the revolving ski slope while wearing the Hololens with the skiing game on it. The revolving ski slope was set on speed two out of ten, to be able to execute the test on people with different levels of experience. Before putting the Hololens on, the participant had the option to practice on the revolving ski slope. Once the participant was wearing the Hololens, he/she got the simple instruction to play the game according to his/her own insights of what must be done in the game in order to achieve a good result. Every participant got the same version of the game, in which they had to ski through gates and avoid obstacles for three minutes. After that time period, a finish line was shown and the game ended with a score overview. When the skiing experience was over, the test participant was asked to fill out a list with questions about perceived cyber sickness symptoms, intrinsic motivation while playing the game, and some extra questions about the multiplayer aspect of the game and using the game as a tool for learning how to ski. The questionnaire that the participants were presented to can be found in Appendix L: Questionnaire First Round of User Tests.

7.2.1.3 Results

The results from the first round of user tests can be divided into four different categories: cyber sickness, intrinsic motivation, multiplayer, and the skiing game as learning tool. For every category, the results from the user tests will be given and explained. Apart from the results from the categories, options for possible improvements that can be made to the skiing game that were mentioned by the participants during the test are included in this section.

Cyber sickness

The cyber sickness category of the test results consisted of the answers that were given to the questions from the Simulator Sickness Questionnaire [16], which was already explained in Chapter 2, under section 2.2.6.2. Participants were asked to indicate how much the symptoms listed in Figure 7.2 were affecting them directly after the augmented reality skiing experiment. They could choose from “none”, “slight”, “moderate”, and “severe”. In Figure 7.2, “none” equals 1, “slight” equals 2, “moderate” equals 3, and “severe” equals 4. Since a score of 1 is equal to “none”, it is decided that the horizontal axis of Figure 7.2 starts at 1. The symptoms listed in Figure 7.2 are divided over the categories oculomotor, disorientation, and nausea, based on how they affect people. For every symptom, the mean score is calculated and given in Figure 7.2. Based on the scores it can be seen that the test participants were barely affected by cyber sickness symptoms directly after the experiment, as most of the mean scores lie around 1 (not affected by the symptom) and none of the mean scores reaches 2 (slightly affected by the symptom). The symptom that received the highest mean score is sweating. However, since the test participants had to deliver physical efforts during the experiment it seems safe to argue that this is the cause, instead of sweating being a symptom of cyber sickness.

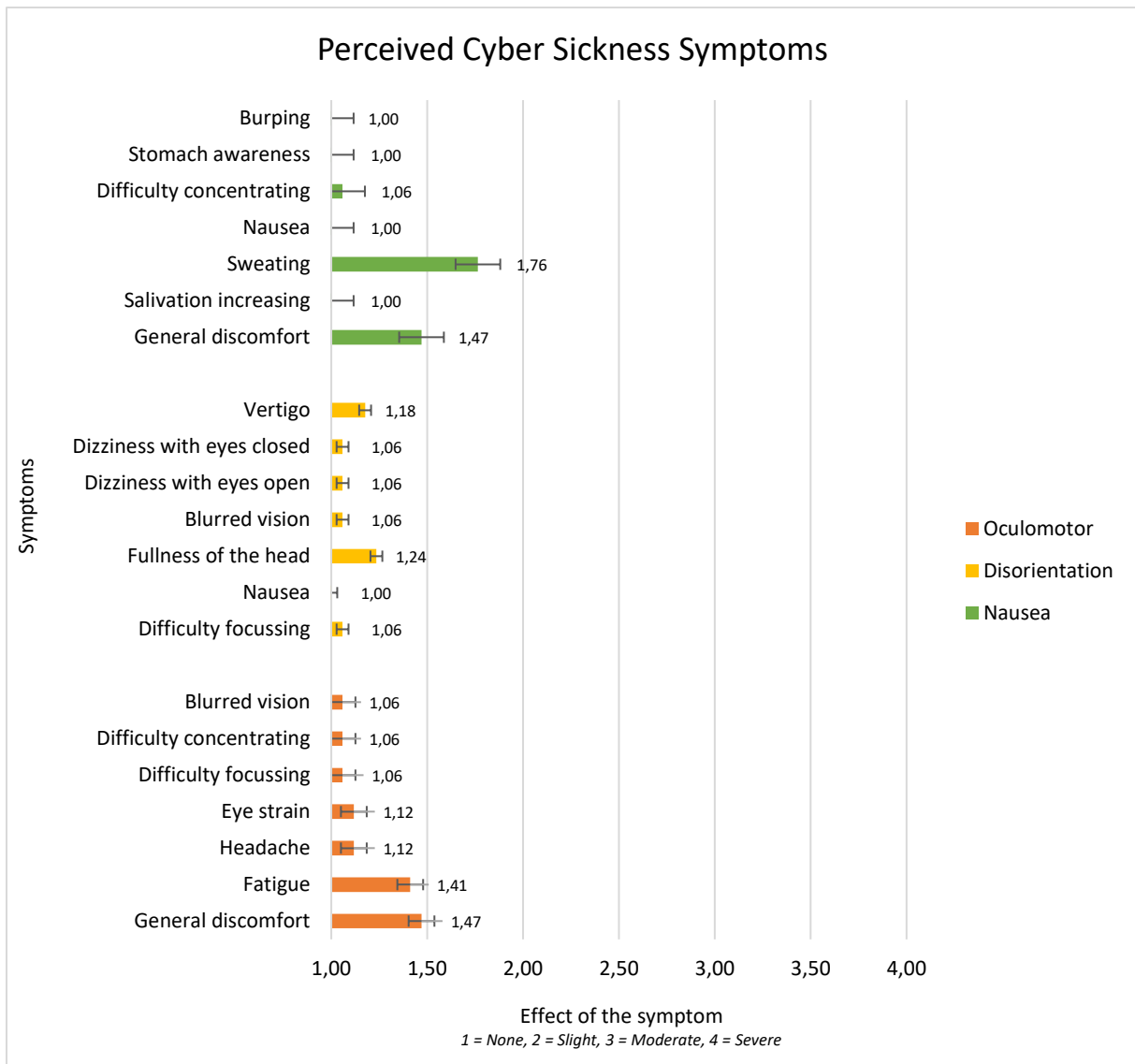


Figure 7.2: Means and standard deviation of participants' perceived cyber sickness symptoms.

Intrinsic motivation

Test participants' intrinsic motivation while playing the game was measured using questions from the Intrinsic Motivation Inventory (IMI) mentioned by Van Delden [17], which was explained in Chapter 2, under section 2.2.6.2. The questions that were included in the test measured participants' effort put in the game and perceived importance of the game, perceived competence, and their interest and enjoyment in the game. Participants were presented to statements about their experience while playing the game. They could rate the statements from one (not true at all) to seven (very true), where four is the middle and therefore neutral. As was already explained in Chapter two, participants had the option to indicate their answers on a scale from one to seven because the IMI test uses that scale as a standard, which makes the calculation of the scores based on the standard procedure described in [18] easier. Besides that, a scale with seven points gives people the option to indicate how certain or uncertain they are about their answers, because it allows them to say that a statement is slightly or very true/not true, instead of only offering the option to say something is true or not true. The means of the answers given to the questions form the bar graph that is presented in

Figure 7.3. In Figure 7.3, all questions are written down per category (effort/importance, perceived competence, interest/enjoyment), with the result that some questions are listed twice.

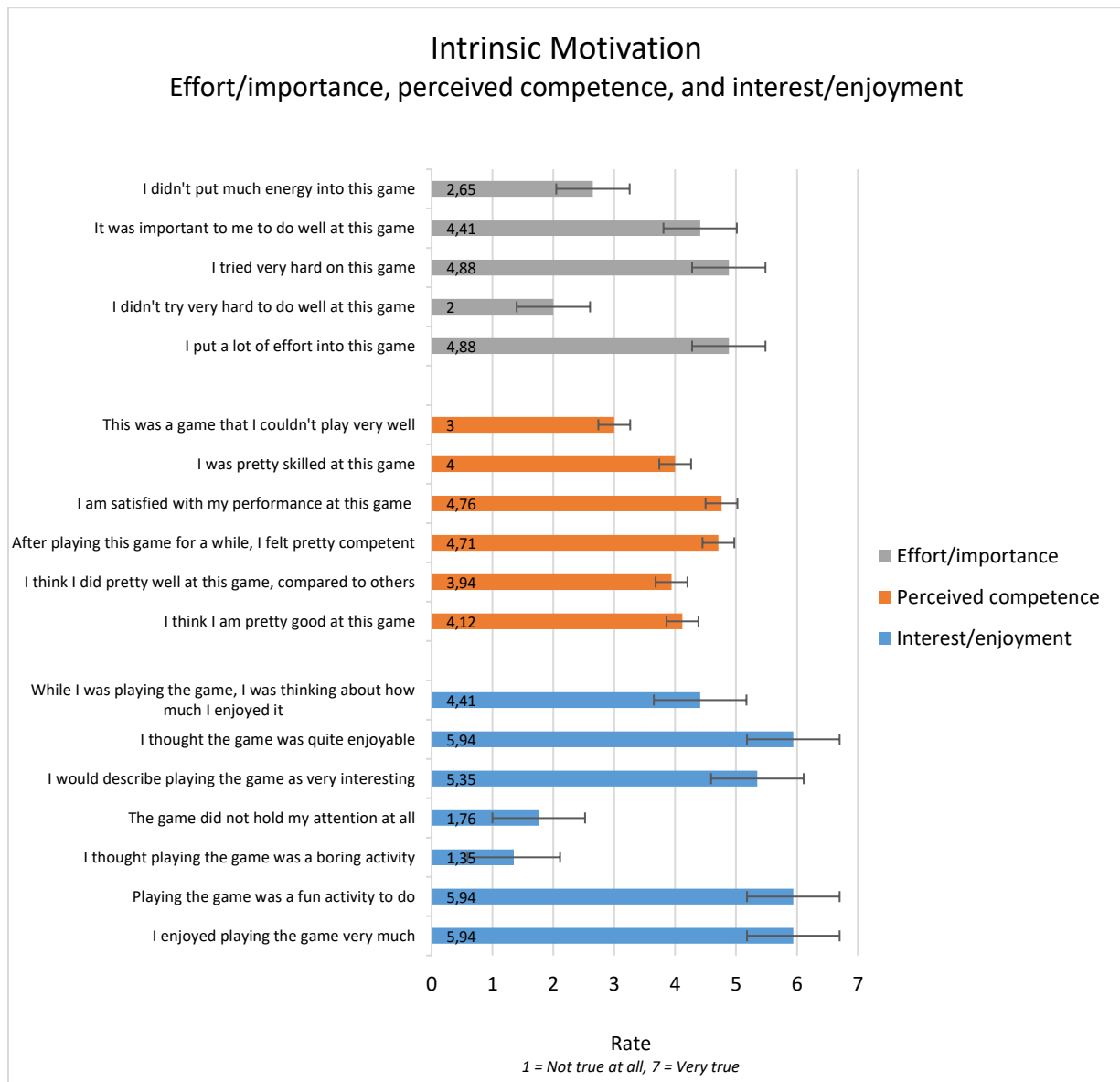


Figure 7.3: Overall scores for the questions about intrinsic motivation, divided in the categories effort/importance, perceived competence, and interest/enjoyment.

From the overview of mean scores that is presented in Figure 7.3 the mean IMI scores per category were calculated and visualized. The mean IMI scores per category can be seen in Figure 7.4. As is explained in Chapter 2 under section 2.2.6.2, some of the statements are “reversed statements”, which means that the scores for these statements had to be reversed by subtracting its score from eight. For more information about this, see section 2.2.6.2. After the IMI values of the reversed statements were computed, the mean score for every category was determined. With four being the neutral score, it can be seen that all the categories were rated higher than neutral. The category interest/enjoyment was rated the highest, indicating that people enjoyed the skiing game very much. This category is followed by effort/importance, which suggests that people put effort in the skiing

game and found it important to do that. The category that was rated lowest is the category of perceived competence, which means that the test participants did not think they were performing very well in the game. The perceived competence category only received a mean score that is slightly larger than the score for neutral.

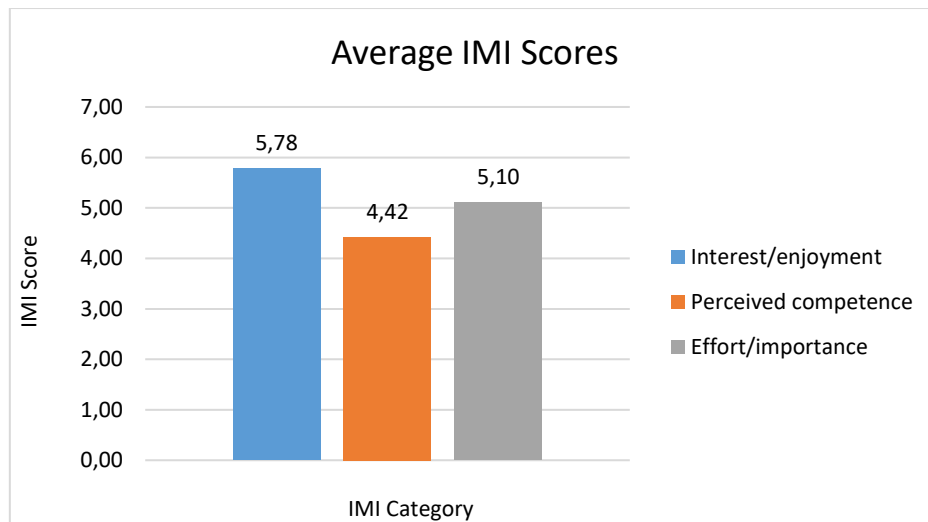


Figure 7.4: Average IMI scores per category (interest/enjoyment, perceived competence, effort/importance) over the whole population.

It was expected that the level of skiing experience of the participants can influence the participants' interest/enjoyment in the game, perceived competence while playing the game, and effort/importance put into the game. Therefore, the mean IMI scores per category were also visualized per level of skiing experience, as can be seen in Figure 7.5. The differences per user group (moderately experienced, experienced, or very experienced) are rather small and the meaning of these differences is probably not significant. Every user group shows the same pattern: interest/enjoyment has the highest score, followed by effort/importance, and then perceived competence. However, a remarkable difference in the mean scores per user group is that the group of experienced skiers shows the lowest perceived competence, while this would be expected for the group of moderately experienced skiers instead. Also, it can be seen that the group of experienced skiers has given the lowest score for all three categories. However, since every user group only consisted of five to six participants, no statistical significance can be derived from these results which means that no conclusions can be drawn for the different user groups.

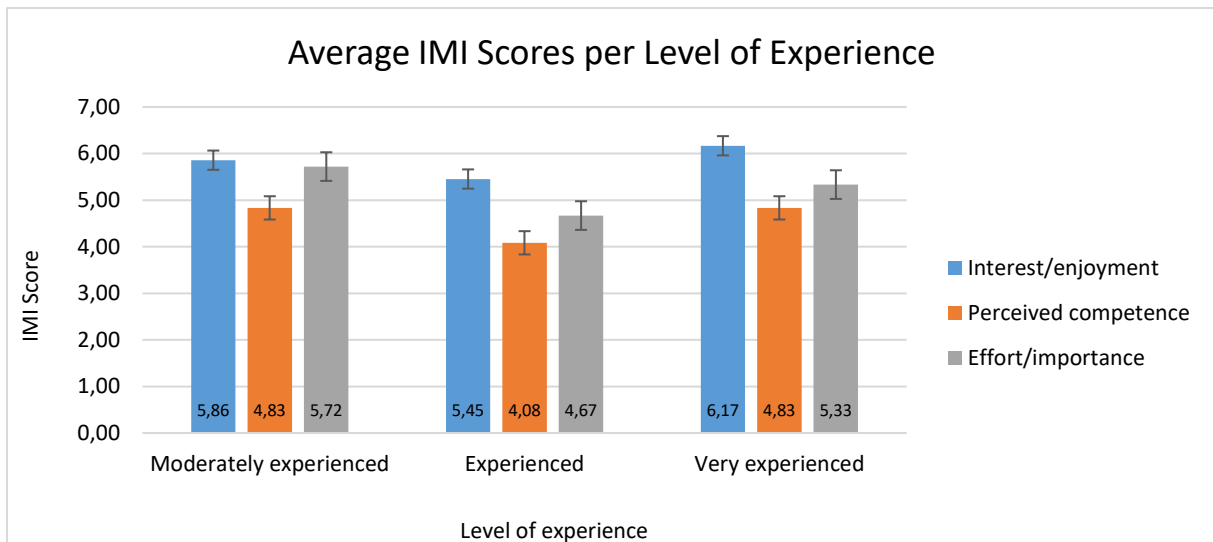


Figure 7.5: Average IMI scores per category (interest/enjoyment, perceived competence, effort/importance) per level of experience.

Multiplayer

The multiplayer aspect of the game and how it is perceived by players was also tested in the user evaluation. Test participants were presented to three statements about the multiplayer aspect of the game, which are included in Figure 7.6. The statements could be rated from one (not true at all) to seven (very true) again, with four being neutral. The mean scores per statement can be seen in Figure 7.6. It must be noted that the multiplayer version of the game only worked in eight out of the seventeen tests, due to the bad quality of the internet connection at the location of the test sessions. Therefore the results from this part of the test are less reliable than the results from other parts of the test.

As can be seen in Figure 7.6, the mean scores for the multiplayer aspect are centred around neutral, which suggests that people do not have a strong opinion about it. Out of the eight test participants that had a fellow player in their game, three did not even notice the fellow player. The five other players *did* notice the fellow player, which explains the mean score given to the statement “I did not notice the fellow player”. Since the people who did not notice the fellow player cannot say anything about if they liked the fellow player or if the multiplayer aspect of the game motivated them, a second visualization was made for these statements. In the second visualisation, which can be seen in Figure 7.7, only the scores given by the people that noticed the fellow player are given. Therefore, Figure 7.7 contains the scores given by a total of five test participants. However, it can be seen that the test participants who were aware of the fellow player liked the fellow player and felt motivated by its presence. Again, because of the limited size of the test group that gave this judgement about the multiplayer aspect of the game, no formal conclusions can be drawn from it.

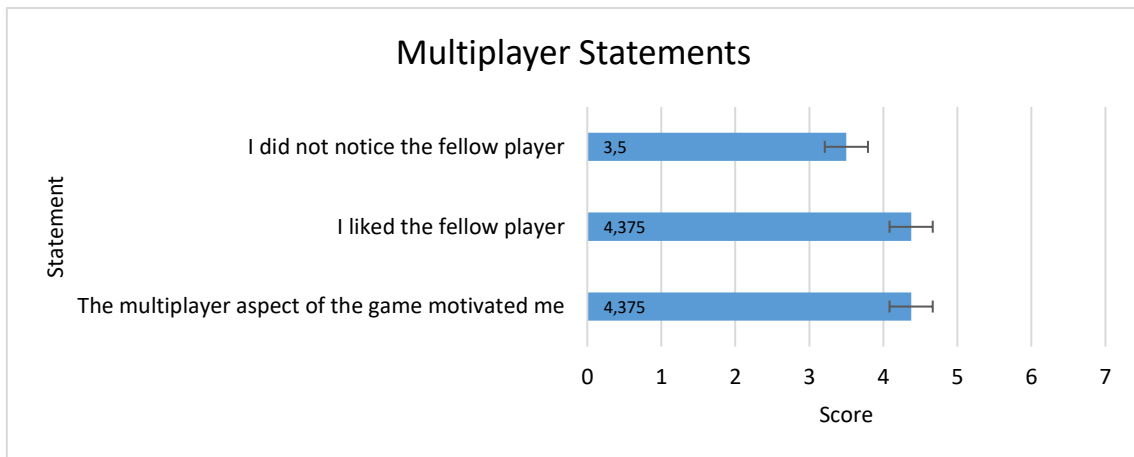


Figure 7.6: Mean scores of the multiplayer statements in the test.

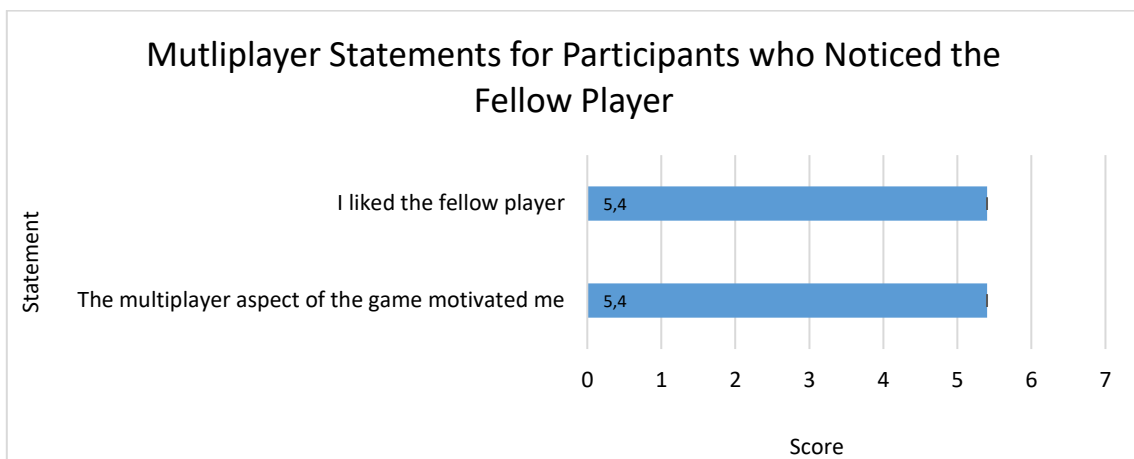


Figure 7.7: Mean scores of the multiplayer statements in the test for people who noticed the fellow player.

Game as Learning Tool

The last part of the test measured participants' view on the use of the game as a tool to learn how to ski. The participants were asked if they thought the game is useful for teaching people how to ski, and if they thought that playing the game more often would help them improve their own skiing skills. Because it was expected that, especially for the last statement, the answers differ per user group (moderately experienced, experienced, very experienced), the scores were visualized per user group and for the total population. The visualization of these scores can be seen in Figure 7.8. It can be seen that every group thinks that the game is a useful tool for teaching people how to ski. Again, the group of experienced skiers gave the lowest score, as was also the case for the IMI scores. The moderately experienced group indicated that they think that playing the game more regularly would help them improve their skiing skills. The experienced skiers and the very experienced skiers do not think this is the case for them. Some of the participants from these two groups mentioned this is because the test was at a very low pace, which does not challenge them enough to learn something new from it. They added to it that if the pace of the revolving ski slope and the game would be increased, they might be able to improve their skills by playing the skiing game.

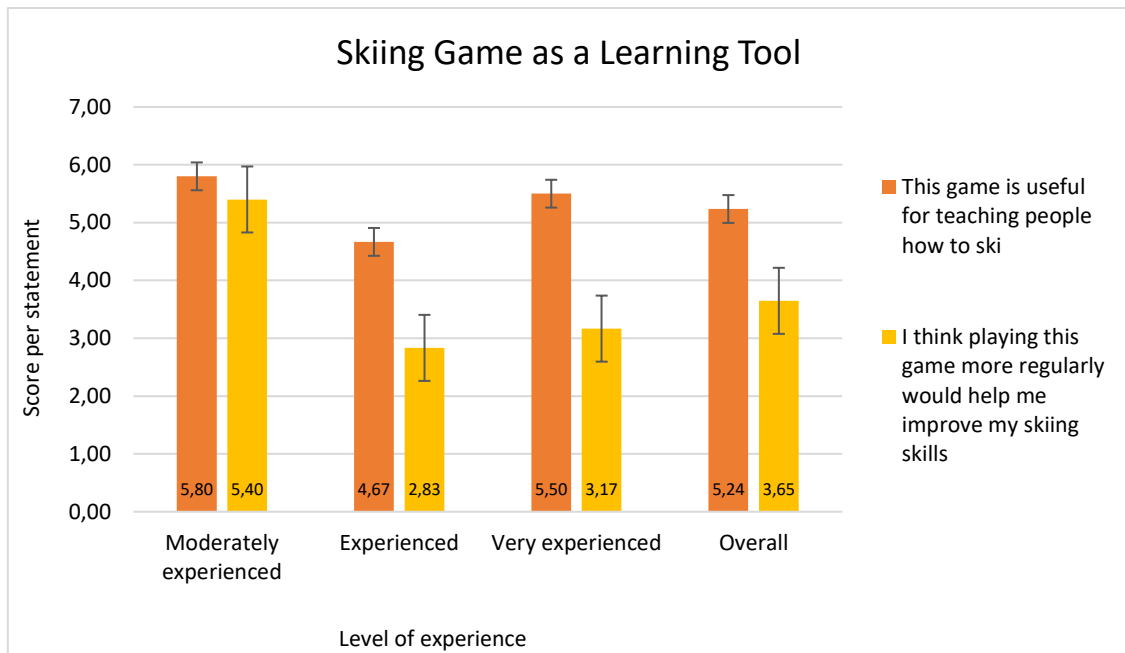


Figure 7.8: Participants' appreciation of the skiing game as a learning tool.

Suggestions for Improvement

At the end of the test, participants were asked to give suggestions for improvements that could be made to the game. Most suggestions could be summarized into three categories: speed, multiplayer, and feedback.

With regard to speed, most experienced and very experienced skiers suggested to increase the speed of the revolving ski slope and the game. They said that skiing becomes easier with increased speed, because that allows them to ski parallel instead of having to ski in snowplough position all the time, which makes manoeuvring on the ski slope easier. Some of them also mentioned adding speed differences to the game based on performance. The latter was already included in the game by decreasing a player's speed when there is a collision with an obstacle. However, this was clearly not noticed by all participants.

In the category multiplayer it was suggested to give a player the option to decide the level of the fellow player and to make the fellow player more clearly present in the game. The latter was mainly suggested by players who discovered there was a fellow player in the game only after they ended the experiment. No suggestions were made on how the presence of the fellow player could be made more clear.

For the third category, feedback, it was mainly the case that people did not realize that the red blocks are obstacles that actually cause a *loss* in points. Most people thought that points could be earned by skiing through the blue gates and also by picking up the red blocks. They assumed that the game only offered the option to gain points and that there was no threat of losing points in the game.

7.2.2 Second Round of User Tests

After the first round of user tests, a second round was held. For the second round, two alterations were made to the skiing game based on the comments about what could be changed to the game

that were received from the participants of the first round. Based on these comments, it was decided that the feedback system of the game could be improved. Most test participants did not realize that the red blocks were obstacles that cause a loss in points when being hit. On the contrary, they thought that colliding with the red blocks also delivered them points. Two solutions were found to solve this problem and to provide the player with better feedback with regard to what should and what should not be done in the game. The first solution was to change the appearance of the red blocks. Their colour remains red, because red is often associated with intensity and anger [40]. Therefore, red seems to be the right colour to indicate a loss in points. The shape of the obstacles was changed into a dangerously looking ball with spikes on it, as can be seen in Figure 7.9. The shape was changed to a spiky object because it was expected that this will indicate that the object is dangerous and that colliding with it causes a negative result.

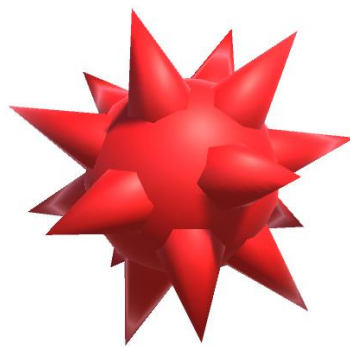


Figure 7.9: The updated appearance of the obstacle game object.

The second solution was to provide extra feedback once an obstacle is hit. In the first version of the game, this was only done by a sound that was perceived as negative. For the second version of the game it was decided to add a red glow to the screen when the obstacle is hit. The red glow appears at the moment the player collides with the obstacle and fades away one second later. This means that when the player collides with the obstacle, the player sees the red glow, hears the negative sound, and sees the number of points decrease with one hundred.

The expectation was that the updated shape of the obstacle and the red glow after collisions with obstacles provide the player with enough feedback to realize that colliding with the obstacles has negative consequences. This hypothesis was tested in the second round of user tests, which is described in this section.

7.2.2.1 Participants

The second round of user tests was executed with four participants in total. All of the four participants had participated in the first round of user tests as well, which means that they had already played the game before. All the participants were male. Two of them indicated themselves to be experienced skiers, the other two said they were very experienced skiers. Among the four test participants were three students and one employee of the University of Twente. Figure 7.10 shows the mentioned characteristics of the group of test participants in pie charts.

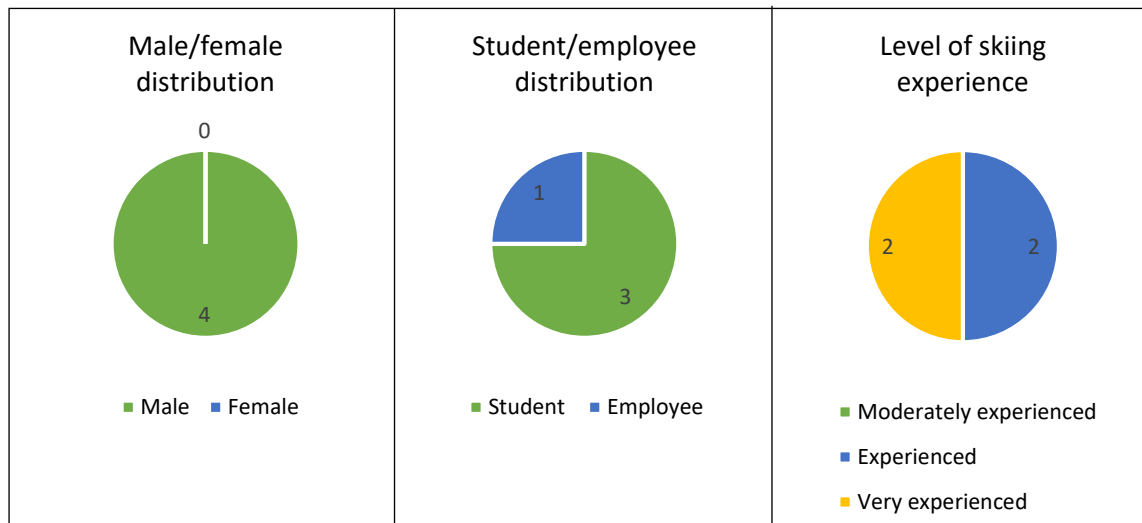


Figure 7.10: Pie charts containing the characteristics of the test participants of the first round of user tests.

7.2.2.2 Test Setup

During the second round of user tests, the same test setup was used as during the first round. This test setup is explained under 7.2.1.2. The only difference was that the participants were presented to a changed version of the game and another questionnaire that had to be filled out after the experiment, which can be found in Appendix M: Questionnaire Second Round of User Tests.

7.2.2.3 Results

The results from the second round of user tests can be divided into two categories: cyber sickness and feedback. In this section the results of the user tests will be discussed per category.

Cyber Sickness

In the second round of user tests the same questions were asked regarding cyber sickness that were asked in the first round of user tests. This was done because it was expected that the addition of the red glow to the game could possibly increase the cyber sickness symptoms, as it is very bright and appears rather abruptly. Figure 7.11 gives an overview of the mean scores for how much the participants in test round two were affected by the listed cyber sickness symptoms. Again, participants could indicate how much they were affected by certain symptoms by choosing from “none”, “slight”, “moderate”, and “severe”. Because 1 equals “none” again, it was decided that the horizontal axis of Figure 7.11 starts at 1. Additionally, the cyber sickness symptoms are divided in the categories nausea, disorientation, and oculomotor again, based on how people are affected by them. The results from the second round of user tests with regard to participants’ perceived cyber sickness are very similar to the results from the first round of user tests, which can be seen in Figure 7.2. Based on this test it can be concluded that the addition of the red glow to the skiing game has no effect on a user’s perceived cyber sickness symptoms.

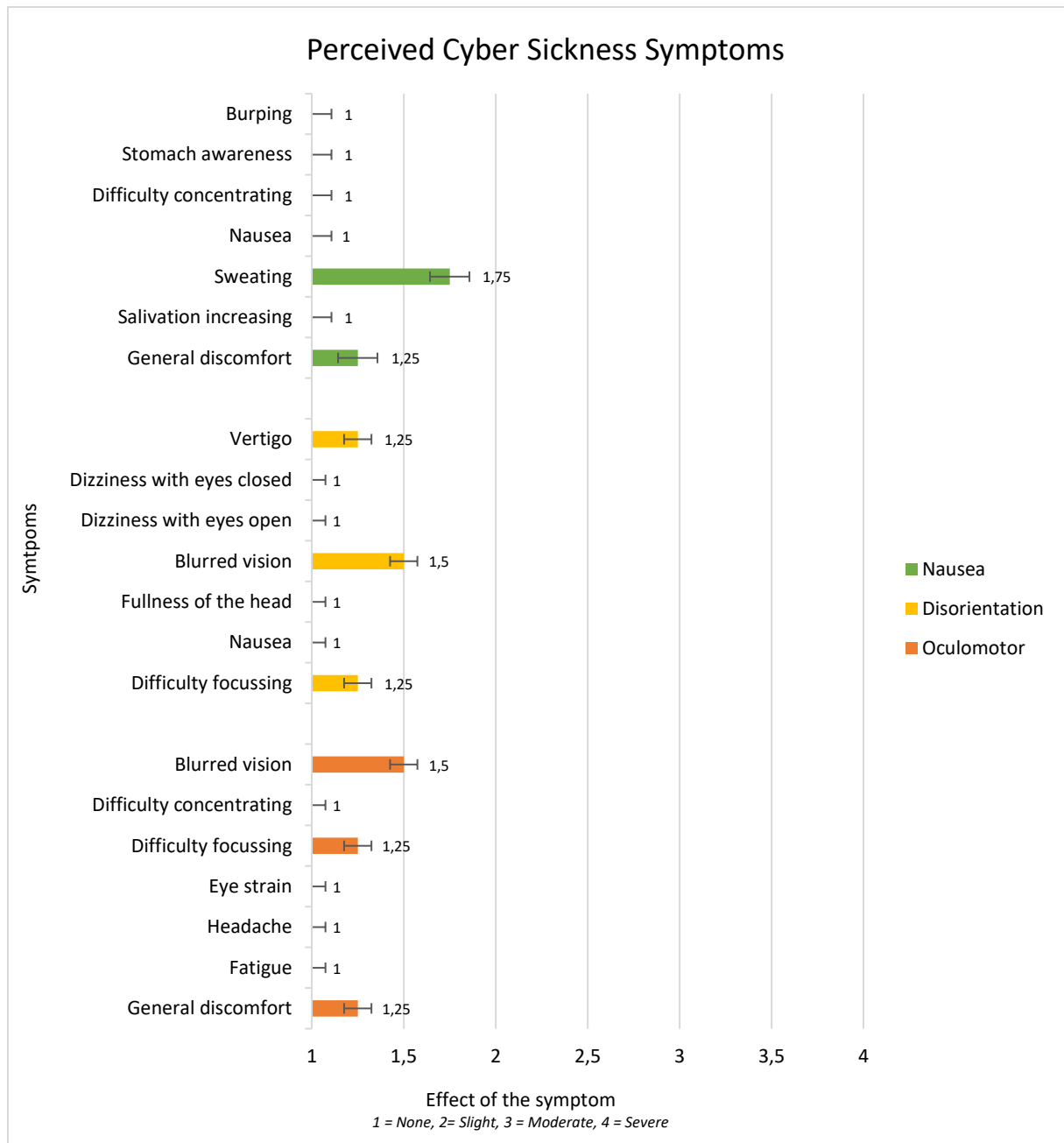


Figure 7.11: Means and standard deviation of participants' perceived cyber sickness symptoms.

Feedback

Participants of the second round of user tests were asked to rate a few questions that are related to the feedback provided by the game. The questions and the mean scores that were given to them by the test participants are shown in Figure 7.12. All four participants agreed that the updated version of the game made them more aware of their mistakes in the game. One of them even thought that the game had changed in such a way, that the obstacles and the subsequent subtraction of points when an obstacle it hit were added to this version of the game. When this person was asked further about this, it became clear that this person was certainly not aware of the obstacles being obstacles in the first version of the game. Besides this, one of the participants said that the game also made him more aware of the things he did right in the game, while the other participants said it did not. Based on the results displayed in Figure 7.12, it can be said that the updated version of the game

makes players more aware of their mistakes, does not influence player's perceptions of what they do right in the game, does not distract people, and is more clear than the previous version of the game. Apart from the results that are shown in Figure 7.12, all test participants indicated that they preferred the updated version of the game over the previous version.

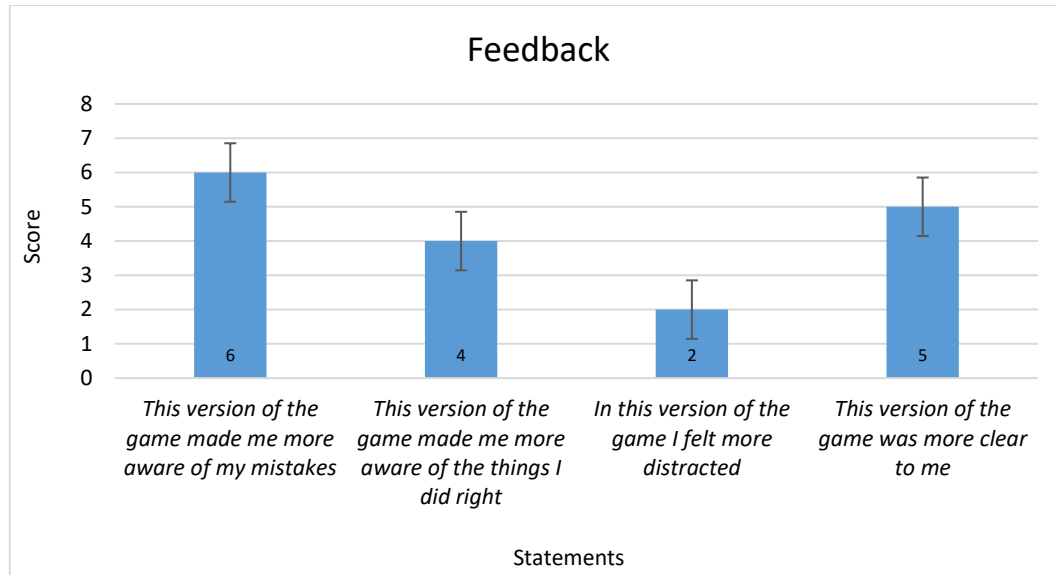


Figure 7.12: Mean scores for statements related to feedback, given during the second round of user tests.

7.3 Conclusion and Discussion of Test Results

This section contains the conclusion and discussion of the test results that were gathered throughout the evaluation phase of this project. This conclusion and discussion is divided in three different sections, which represent the three different parts of the evaluation phase. Section 7.3.1 contains the conclusion and discussion for the functional evaluation, section 7.3.2 for the first round of user tests, and section 7.3.3 for the second round of user tests.

7.3.1 Functional Evaluation

The functional evaluation showed that all *must have*-requirements were met by the prototype of the skiing game. All *should have*-requirements were met, except for the requirement of multimodal feedback since no haptic feedback was included in the prototype. Finally, the *could have*-requirements were only partially met, due to the limited time available for the development of the prototype. The *could have*-requirements that were not met were the ranking, triangularity in the game, and the game being an asymmetrical game. Based on the functional evaluation it was concluded that the prototype functions well enough for people to use it as a skiing game.

7.3.2 First Round of User Tests

In the first round of user tests, the participants' perceived cyber sickness symptoms, the participants'

intrinsic motivation, the multiplayer aspect of the game, and the learning aspect of the game were investigated for the first version of the skiing game.

With regard to perceived cyber sickness it can be concluded that the augmented reality skiing game does not evoke cyber sickness symptoms. This confirms the theory by Rebenitsch and Owen [32] that inclusion of the real world in an application, which is done in augmented reality, can prevent cyber sickness. However, it must be noted that this was only tested on a small group of seventeen participants, which means that no statistical significance can be derived from these results. Besides that, it could be the case that the exposure did not last long enough for participants to be affected by cyber sickness symptoms. According to research by Bruck and Watters [30], cyber sickness symptoms were only increased after six to ten minutes of exposure, while the experiment lasted only three minutes.

The intrinsic motivation scores that were obtained from the test were divided into the categories interest/enjoyment, perceived competence, and effort/importance. The categories interest/enjoyment and effort/importance were rated higher than neutral. Based on these results, it can be concluded that the participants felt interested and enjoyed while playing the skiing game and that the test participants put effort into the skiing game and found it important. The category perceived competence scored lower, only slightly above neutral. However, this is not seen as a negative score since perceived competence should not be too high, as being very competent in the skiing game already might cause boredom for the players. Again, these results were obtained out of a group of seventeen test participants, which means that they are not statistically valid.

With respect to the multiplayer aspect of the game, only eight out of seventeen test participants were presented to the multiplayer version of the game since the connection failed in the other cases. Out of the eight participants who played the multiplayer version of the game, only five noticed the fellow player. The five participants who noticed the fellow player indicated that they liked the fellow player and that its presence motivated them. However, no statistical significance can be derived from these results since they were generated by such a small group.

Finally, the learning aspect of the game was evaluated very positively. The test participants indicated that they think that the skiing game is a useful tool for learning how to ski. The test participants who were moderately experienced at skiing also indicated that playing the game more regularly would help them increase their own skiing skills. The other two groups of test participants, the experienced skiers and the very experienced skiers, said that they did not think they can learn anything from the game. Some of them indicated that this was mainly due to the low pace that the game is being played at and that they might be able to increase their skiing skills by playing the game if the speed of the game would be increased. Again, these results do not have any statistical relevance since they are generated by a group of seventeen test participants.

7.3.3 Second Round of User Tests

The second round of user tests was executed with only four participants, who also participated in the first round of user tests. Therefore the conclusions drawn in this section are not statistically valid. Because the test participants had already participated in the experiment, their judgement about the skiing game could be biased. The goal of the second round of user tests was to investigate if the feedback system of the game had improved.

It can be concluded that the improved version of the game did not cause any cyber sickness symptoms, as was also the case for the first version of the game. Additionally, the feedback that was

given in the improved version of the game made the test participants more aware of the things they did wrong in the game. All four test participants preferred the improved version of the skiing game over the first version.

Chapter 8 – Discussion and Conclusion

This chapter gives the conclusions and discussion of this research. Section 8.1 provides the conclusions, which answer the research questions that were posed in Chapter 1. Additionally, section 8.2 provides a discussion that places this project and its relevance in a broader perspective.

8.1 Conclusions

In Chapter 1, four research questions were proposed to be investigated throughout this project. The main research question of this project was: *“How to design a game in augmented reality that supports the ski-learning process?”*. Besides the main research question, three sub-questions were formulated to guide this project in a certain direction. The first sub-question was: *“What is the added value of a game in augmented reality for skiing classes on a revolving ski slope?”*. The second sub-question was: *“How do people perceive a game in augmented reality that is meant to support the ski-learning process?”*. Finally, the third sub-question was: *“Does the augmented reality skiing game prototype that resulted from this project cause its players to suffer from cyber sickness symptoms?”*. This section first provides the answers to the three sub-questions, after which the main research question will be answered.

With respect to the added value of a game in augmented reality for skiing classes on a revolving ski slope it can be concluded that the game that was developed over the course of this project adds value through its multiplayer aspect and its entertaining factor. From the background research that was done at the beginning of this project, it became clear that multiplayer games offer competition, enhance the replayability of games, and add a social aspect to games. These three factors add to the motivation of players to play the game. These findings were confirmed in the user evaluation, in which the participants of the experiment stated that they liked the fellow player and that its presence motivated them. This can be seen as added value offered by the augmented reality skiing game, since skiing together or against each other is not easily possible during regular ski classes on a revolving ski slope. The augmented reality skiing game makes it very easy to ski together, even when people are not physically present at the same location, by making a connection over the internet. Secondly, the augmented reality skiing game adds value to skiing classes on a revolving ski slope by its entertaining factor. The results from the background research indicated that games are seen as entertaining, which makes learning through games a fun activity to do. These findings were also confirmed during the user evaluation. The test participants gave high values for their perceived enjoyment while playing the game and their interest in playing the game. Based on these results, it can be concluded that an augmented reality skiing game adds entertainment to skiing classes on a revolving ski slope.

Regarding the way people perceive a game in augmented reality that is meant to support the ski-learning process it can be concluded that people perceive the game as interesting, enjoying, something to put effort into, important, and suitable for learning purposes. These conclusions are based on the user evaluation that was carried out during the evaluation phase of this project. As was already mentioned in the previous section, people gave high values for their perceived enjoyment while playing the skiing game and thought that it was an interesting thing to do. Apart from these results, people rated the effort they put into the skiing game and the importance they attached to it slightly less high, but still convincingly high to conclude a positive result. The test participants did not

necessarily think they were good at playing the game, since they rated their perceived competence around neutral. However, this might add to the challenge that is offered by the game and prevent boredom while playing it, as people do not perceive the skiing game as too easy or too difficult. During the user evaluation, the participants were confident that the skiing game that was developed throughout this project is a useful tool for teaching people how to ski. However, only the least skilled group of test participants thought that the skiing game could help them improve their own skiing skills.

Additionally, it can be concluded that the skiing game in augmented reality does not cause its players to suffer from cyber sickness symptoms. This conclusion is based on the results from the user tests that were carried out during the evaluation phase of this project, in which the participants indicated that they were not or only slightly affected by the different cyber sickness symptoms that were listed in the test. These results indicate that people were affected by cyber sickness symptoms to the most minimal and even negligible extent.

Finally, the main research question of this project can be answered. From the background research that was carried out at the beginning of this project it became clear that games can be seen as a useful tool for teaching purposes since they keep the players motivated, provide them with feedback, and create a learning effect. Games can create a learning effect by offering the player clear tasks and explanations, by addressing their prior knowledge, by using rules to clarify the game, and by decreasing the guidance that is offered in the game. Based on this background knowledge, an ideation phase was started to generate ideas about possible implementations of an augmented reality game that is meant to support the ski-learning process. Some of the ideas that were generated during the ideation phase were taken into further consideration in the specification phase. Based on the specified requirements that the skiing game should fulfil, an augmented reality application was made for the Microsoft HoloLens. To do so, three-dimensional models were made in order to be used as the game objects that had to be placed in the game. A total of nine scripts were written for the game to function. Based on the combination of these nine scripts, a working prototype was made. The scripts that were used in the prototype of the skiing game fulfilled the following purposes: the detection of collisions between the player and the game objects, the positioning of the UI elements, updating the information that is displayed in the UI elements, the representation of a finish line, the network settings, the identification of the camera per player, the rotation of the skies of the player game object, and the initiation and spawning of the game objects. In the final phase of this project, the evaluation phase, it was discovered that the players of the game did not understand that the obstacles in the game were meant as obstacles and caused a decrease in points. Therefore, a design iteration was made to improve the feedback system of the game. The final version of the working prototype of the skiing game was changed in such a way that the obstacles appeared more dangerously looking and that a red glow is shown when an obstacle is hit. The test participants indicated that they preferred this version of the game over the first version.

8.2 Discussion

Besides the research questions that were answered in the conclusion section, this research can be viewed in a broader perspective. This section provides the discussion of this research, which puts the research in a wider context and highlights certain other directions that could have been or potentially can be chosen for this project.

First of all, one of the main things that could have been changed to the skiing game is the device that the game has been built for. The current version of the game works on the Microsoft HoloLens, which

enables the tracking of the player's movements and position in the physical world and uses that as input to determine the accompanying view that the user is presented to in the application. However, the HoloLens might not be the most suitable or desirable device that the skiing game could have been built for, since it is a rather big and heavy device, and the overarching strap that tightens the HoloLens around a person's head sometimes reduces the field of view of the person, especially at the top and the sides. Additionally, the HoloLens comes with a certain price tag that cannot be categorized under consumer goods prices, which makes the device inaccessible to the general public. A more lightweight and affordable device, like the Moverio Epson BT-300 or a device that was already designed as skiing goggles, is desired. It is expected that such a device fits better during the activity of skiing and makes the skiing game accessible to the general public. The game could even be implemented on smartphones that can be placed in a cardboard holder, which is probably the cheapest but not the most ergonomically optimal option. However, if the game would be implemented on another augmented reality device, it means that the issue of tracking the user's movements and positions and the communication of the tracked information to the game should be solved.

Second, it must be noted that the skiing game prototype that resulted from this project serves as a proof of concept, not as an end product. The game can be extended in a number of ways to make it more fun and interesting to use. The current version of the game consists of one level, which served well during the user tests of this project. However, this is not desirable for a final version. In order to keep the game interesting for its players, the game's levels and possibly its reward system should be extended. The question remains if a graphical user interface should be added to the game, because it is doubtful whether this option is desirable. When people are skiing on the ski slope, it has proven to be very difficult for them to navigate through a graphical user interface at the same time. However, if the skiing game would be extended, some form of a graphical user interface is probably needed to allow the user to navigate himself/herself through the different options that are offered by the game.

Third, the multiplayer aspect of the game seemed to be not very clearly present to the players. Out of the eight test participants who were presented to the multiplayer version of the game, three participants did not even recognize the fellow player. This could be partially due to the fact that they did not know they were competing against someone else. However, it also indicates that the fellow player's presence might be too subtle in the current version of the skiing game.

Fourth, the skiing game is made to support the ski-learning process, while no skiing teachers are involved in the design of the current version of the game. A design option was considered that included the teacher in the system of the skiing game through a second device with its own interface that could be used to influence the parameters of the game. However, due to the limited timeline of this project, that design option was categorized under the *won't have*-requirements of this project. The current version of the game can be used by skiing teachers to allow their pupils to practice on their own and to compete against each other. However, it is desirable to develop a version of the skiing game that allows for interventions by the teacher.

Finally, the question remains why this project could not have been executed on a virtual reality device, such as the HTC Vive. The HTC Vive offers reliable tracking in the physical world, which is used in its applications to determine where the user is present in the virtual environment. Apart from the tracking possibilities, virtual reality can be argued to be more immersive and provide the user with a better skiing experience than augmented reality. However, the outcomes of this research demonstrate that the use of the HTC Vive or a comparable virtual reality head-mounted display to

play the skiing game on is highly inadequate and undesirable. The main reason is that a virtual reality head-mounted display prevents the user from being able to see his/her normal surroundings in the real world, which causes dangerous situations on the revolving ski slope because users do not get sufficient visual clues that suggest that they have to correct their movements to prevent accidents. However, even if a way would be found to make the situation on the ski slope safer, virtual reality still remains an insufficient technology for this project since the risk for cyber sickness symptoms is significantly higher with such devices. This project has proven that the use of an augmented reality device prevents cyber sickness symptoms from occurring, which is a valuable insight that is strongly advised to be adhered to.

Chapter 9 – Future Work

Based on the research that was carried out throughout this project, some recommendations for further research can be made. This section provides an overview of possible future work that can be done for this project.

First of all, it is recommended to make the skiing game applicable for multiple augmented reality devices. By doing so, the skiing game becomes suitable for low-end devices as well (such as smartphones that can be placed in a cardboard holder), which makes the product available to a greater audience. In order to do so, further research must be done with regard to the possibilities for tracking a user's position and movements in the physical world.

Second, it is recommended to do further research with regard to the representation of the fellow player in the game. With respect to the fellow player, two problems are recommended to be solved. The first problem is the problem of the fellow player not being clearly present to the other players of the game. The second problem is the positioning of the representation of the fellow player in the game during gameplay, which is not correct along the x-axis in the current version of the game.

Third, further research could be done to create a correct ranking at the end of the skiing game, which provides the players with summary feedback. In order to do so, it must be investigated how an overview of the players' names and the scores that they achieved in the game can be created.

Fourth, it is advised to extend the skiing game's levels and reward system. The current version of the skiing game consists of one level only and with regard to the reward system only points can be earned. The skiing game can be made more interesting and can stay interesting for longer if it would be expanded with extra levels and a more elaborate reward system. Other options and functionalities that could be added to the skiing game include, for example, the option to make a player profile, the option to choose the level of difficulty of the game, and the option to make a group of players that can play together and keep relevant statistics of their played matches. Additionally, it is recommended to add a graphical user interface to the game once its functionality expands, to allow users to navigate themselves through the different options that the skiing game has to offer. If this option would be implemented, appropriate ways of presenting users to a graphical user interface while they are skiing on the ski slope should be researched.

Fifth, the game can be extended to a version where teachers are involved as well, by allowing them to influence certain parameters in the game through a second device, such as the speed or the placement of obstacles and gates. This idea was already listed under the *won't have*-requirements of this project, because of the limited timeline of this project. However, it is advisable and desirable to include skiing teachers in a future version of the skiing game. By including teachers, the option of non-experienced skiers using the skiing game can be explored again. Additionally, the inclusion of teachers can enhance the teaching possibilities of the skiing game.

A final recommendation for future work is to make the options for inserting parameters into the skiing game more dynamic. In the current version of the game, values can be set for, among others, the width of the slope, the speed of the game, and the angle of inclination. Once the values are set and the game is built into an application that can run on the augmented reality device, there is no option to change the parameters again. However, it is desirable to include this option to prevent having to build the application several times in order to set the appropriate values. This option could for example be implemented in the graphical user interface that the player is presented to, where

he/she can indicate the desired speed, the width, and the angle of inclination of the slope that the skiing game is being played at. Additionally, it would be desirable to make a version of the game that can measure the speed of the slope in the real world, and adjusts the speed of the game accordingly. Having this option in the game would only require the simple action of turning the speed knob that comes with the revolving ski slope to set the desired speed for the skiing game.

Appendix

Appendix A. Code Markerless Augmented Reality Prototype

The code that was used for the markerless augmented reality prototype is taken from [42] and consists of three different scripts: Webcam Script, Collision Script, and Enemy Script.

A.1 Webcam Script

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.UI;

public class webCamScript : MonoBehaviour {

    public GameObject webCameraPlane;
    public Button fireButton;

    void Start () {
        if (Application.isMobilePlatform) {
            GameObject cameraParent = new GameObject ("camParent");
            cameraParent.transform.position = this.transform.position;
            this.transform.parent = cameraParent.transform;
            cameraParent.transform.Rotate (Vector3.right, 90);
        }

        Input.gyro.enabled = true;

        fireButton.onClick.AddListener (OnButtonDown);

        WebCamTexture webCameraTexture = new WebCamTexture ();
        webCameraPlane.GetComponent<MeshRenderer> ().material.mainTexture = webCameraTexture;
        webCameraTexture.Play ();
    }

    void OnButtonDown(){
        GameObject bullet = Instantiate(Resources.Load("bullet", typeof(GameObject))) as GameObject;
        Rigidbody rb = bullet.GetComponent<Rigidbody>();
        bullet.transform.rotation = Camera.main.transform.rotation;
        bullet.transform.position = Camera.main.transform.position;
        rb.AddForce(Camera.main.transform.forward * 500f);
        Destroy (bullet, 3);
        GetComponent<AudioSource> ().Play ();
    }

    void Update () {
        Quaternion cameraRotation = new Quaternion (Input.gyro.attitude.x, Input.gyro.attitude.y, -
        Input.gyro.attitude.z, -Input.gyro.attitude.w);
        this.transform.localRotation = cameraRotation;
    }
}
```

A.2 Collision Script

```
using UnityEngine;
using System.Collections;

public class collisionScript : MonoBehaviour {

    void Start () {
    }

    void Update () {
    }
}
```

//for this to work both need colliders, one must have rigid body (spaceship) the other must have is trigger checked.

```
void OnTriggerEnter (Collider col)
{
    GameObject explosion = Instantiate(Resources.Load("FlareMobile", typeof(GameObject))) as
    GameObject;
    explosion.transform.position = transform.position;
    Destroy(col.gameObject);
    Destroy (explosion, 2);

    if (GameObject.FindGameObjectsWithTag("Player").Length == 0){
        GameObject enemy = Instantiate(Resources.Load("enemy", typeof(GameObject))) as GameObject;
        GameObject enemy1 = Instantiate(Resources.Load("enemy1", typeof(GameObject))) as GameObject;
        GameObject enemy2 = Instantiate(Resources.Load("enemy2", typeof(GameObject))) as GameObject;
        GameObject enemy3 = Instantiate(Resources.Load("enemy3", typeof(GameObject))) as GameObject;
    }
    Destroy (gameObject);
}
}
```

A.3 Enemy Script

```
using UnityEngine;
using System.Collections;

public class enemyScript : MonoBehaviour {

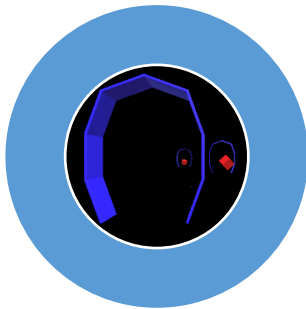
    void Start () {
        StartCoroutine ("Move");
    }

    void Update () {
        transform.Translate(Vector3.forward * 3f * Time.deltaTime);
    }

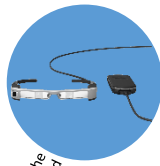
    IEnumerator Move() {
        while (true) {
            yield return new WaitForSeconds (3.5f);
            transform.eulerAngles += new Vector3 (0, 180f, 0);
        }
    }
}
```

Appendix B. Realisation Timeline

Week 1: First Version



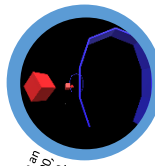
Version made for the Epson Moverio BT-300, the gyroscope is used for looking around



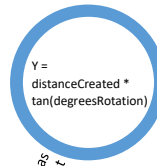
- Gyroscope did not work reliably



Immersiveness: Give the application a black background to provide a see-through on the glasses



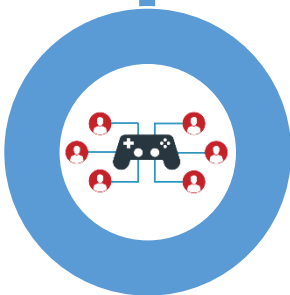
Random placement of obstacles and gates, collision with an obstacle decreases the total amount of points by 100, collision with a gate increases this with 200.



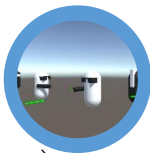
Slope: compute the position that the player has to move towards based on the tangent

- $Y = \text{distanceCreated} * \tan(\text{degreesRotation})$

Week 2: Multiplayer



Networking tutorial by Unity



Network Lobby by Unity: offers the opportunity to make 'rooms' in which players can connect to each other



Problem: How can players see one another?



- Extra screen at the bottom of the game to show where the fellow player is, probably too distracting
- Differences in speed allow the player that is behind to see the player(s) in front of him

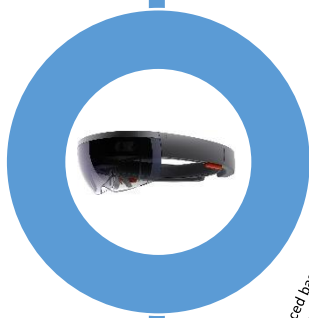


Multiplayer version of the skiing game, in which players can connect via 'rooms' or each other's IP address

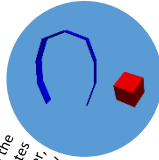


Give every player a forward speed, so that speed differences can be implemented and players can move through the game

Week 3: Multiplayer & Hololens Implementation



Obstacles and gates are placed based on the location of the player. That means that the gates which is not fair for the location of one player, obstacles and gates are made based on the position of previous obstacles and gates.



The speed of the player is influenced by them skilling through the gates (+) or by colliding with an obstacle (-), later on this was changed to only decreasing the speed when an obstacle is hit.



Game on HoloLens: UI elements disappeared, UI needs to be in World Space mode. Creating a GUI needed to determine in which mode (host, client, server) the game is being played, but was decided that the HoloLens starts as a host automatically and that others can connect via the HoloLens IP address.



HoloLens camera could not be moved forward in the game at a certain speed. Instead, it was given a parent that moves forward at a certain speed, x-position of the camera child with it. Problem: the position of the camera, so the representation of the HoloLens player along the x-axis is incorrect.



Week 4: Completion



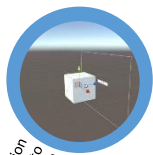
Placement of trees in the game, based on the same principle as was used for the placement of gates and obstacles.



- HoloLens cannot handle a great number of game objects (trees) to be created at once. This was solved by only making trees at a certain distance from the player.
- This principle was then also applied to the gates and obstacles.



Inclusion of audio. When an obstacle is hit, a negative (low) sound is being played. When a gate is hit, a positive (high) sound is being played.



Canvas position was set based on the position of the player and the rotation of the slope (to prevent the player from having to look up, in the case of a steep slope).



Inclusion of a finish line at the end of the game. The word 'Finish' was modeled in Maya and floats above it.



Snow was added to the scene, based on the same principle as the trees.



A count down timer was added at the beginning of the scene so that the skier can get himself ready on the slope before the game starts.

Appendix C. Spawner Script

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.Networking;

public class Spawner : NetworkBehaviour {

    public GameObject obstaclePrefab;
    public GameObject arcPrefab;
    public GameObject treePrefab;
    public GameObject finishPrefab;
    public GameObject snowPrefab;

    public GameObject obstacles;
    public GameObject arcs;
    public GameObject trees;
    public GameObject snows;

    private GameObject obstacle; //to store the created obstacles in
    private GameObject arc; //to store the created arcs in
    private GameObject treeleft; //to store the created trees in
    private GameObject treeright;
    private GameObject finish;
    private GameObject snow;

    public float timeToDestroy; //the time after which obstacle/arc is destroyed
    public float timeToDestroyTree; //the time after which trees are destroyed
    public int minX; //the left boundary of the obstacles and arcs
    public int maxX; //the right boundary of the obstacles and arcs
    public int minXTree; //the minimum distance of the trees from the center of the game area
    public int maxXTree; //the maximum distance of the trees from the center of the game area
    public int minDistanceBetweenObjects; //an integer that stores the minimal distance from the player
    at which the obstacle/arc is created
    public int maxDistanceBetweenObjects; //an integer that stores the maximum distance from the player
    at which the obstacle/arc is created
    public int distanceCreated; //the distance from the player at which prefabs are spawned

    private static float radRotation; //variable that stores the radians that the slope is tilted
    private int distanceBetweenObjects; //variable to store the distance at which the obstacle/arc is
    created from the player
    private int positionX; //the x position of the arcs and obstacles
    private float posYObj; //the y position where the objects will be placed
    private float posYTree; //the y position where the trees will be placed
    private Vector3 playerPosition; //store the position of the player for the first obstacle and arc
    that will be placed

    void Start () {
        if(isServer){ //only the server can invoke obstacles, arcs, and trees
            Invoke("PlaceFirstObstacle", Random.Range(2,4)); //place the first obstacle
            Invoke("PlaceFirstArc", Random.Range(1,2)); //place the first arc
            Invoke("PlaceFirstTreeLeft", Random.Range(1,1)); //place the first tree on the left side
            Invoke("PlaceFirstTreeRight", Random.Range(1,1)); //place the first tree on the right side
            Invoke("PlaceFirstSnow", Random.Range(1,1)); //place the first snow
        }
    }

    void Update () {
        //keep checking if more trees are needed
        if (treeright.transform.position.z <= PlayerManager.positionZ + distanceCreated) {
            Invoke ("PlaceTreesRight", 0);
        } else if (treeleft.transform.position.z <= PlayerManager.positionZ + distanceCreated) {
            Invoke ("PlaceTreesLeft", 0);
        } else if (obstacle.transform.position.z <= PlayerManager.positionZ + distanceCreated) {
            Invoke ("PlaceObstacles", 0);
        } else if (arc.transform.position.z <= PlayerManager.positionZ + distanceCreated) {
```

```

        Invoke ("PlaceArcs", 0);
    }
}

// Place the first obstacle, then place all the other obstacles based on the first obstacle's
// position
void PlaceFirstObstacle(){
    playerPosition = new Vector3 (100, 0, 0);
    distanceBetweenObjects = 35;
    radRotation = PlayerManager.radRotation;

    obstacle = (GameObject)Instantiate(obstaclePrefab, new Vector3(playerPosition.x,
-1.5f - (distanceBetweenObjects * Mathf.Tan(radRotation)), distanceBetweenObjects +
playerPosition.z), Quaternion.identity);
    Invoke("PlaceObstacles", 0); //take 1-3 seconds to call PlaceObstacles()
}

// Place the first arc, then place all the other arcs based on the first arc's position
void PlaceFirstArc(){
    playerPosition = new Vector3 (-1, 0, 0);
    distanceBetweenObjects = 35;
    radRotation = PlayerManager.radRotation;

    arc = (GameObject)Instantiate(arcPrefab, new Vector3(playerPosition.x,
-2- (distanceBetweenObjects * Mathf.Tan(radRotation)), distanceBetweenObjects +
playerPosition.z), Quaternion.identity);
    Invoke("PlaceArcs", 0); //take 1-2 seconds to call PlaceArcs()
}

// Place the first tree on the left side, then place all the other trees based on the first tree's
// position
void PlaceFirstTreeLeft(){
    playerPosition = new Vector3 (-1, 0, 0);
    distanceBetweenObjects = 20;
    radRotation = PlayerManager.radRotation;

    treeleft = (GameObject)Instantiate(treePrefab, new Vector3(playerPosition.x-4,
-2- (distanceBetweenObjects * Mathf.Tan(radRotation)), distanceBetweenObjects +
playerPosition.z), Quaternion.identity);
    Invoke("PlaceTreesLeft", Random.Range(0,1)); //take 0-1 seconds to call PlaceTreesLeft()
}

// Place the first tree on the right side, then place all the other trees based on the first tree's
// position
void PlaceFirstTreeRight(){
    playerPosition = new Vector3 (-1, 0, 0);
    distanceBetweenObjects = 20;
    radRotation = PlayerManager.radRotation;

    treeright = (GameObject)Instantiate(treePrefab, new Vector3(playerPosition.x+4,
-2- (distanceBetweenObjects * Mathf.Tan(radRotation)), distanceBetweenObjects +
playerPosition.z), Quaternion.identity);
    Invoke("PlaceTreesRight", Random.Range(0,1)); //take 0-1 seconds to call PlaceTreesRight()
}

// Place the first snow, then place all the other snows based on the first snow's position
void PlaceFirstSnow(){
    playerPosition = new Vector3 (0, 0, 0);
    distanceBetweenObjects = 20;
    radRotation = PlayerManager.radRotation;

    snow = (GameObject)Instantiate(snowPrefab, new Vector3(playerPosition.x,
5-(distanceBetweenObjects * Mathf.Tan(radRotation)), distanceBetweenObjects +
playerPosition.z), Quaternion.identity);
    Invoke("PlaceSnow", Random.Range(0,1)); //take 0-1 seconds to call PlaceSnow()
}
}

```

```

void PlaceObstacles(){
    if (obstacle.transform.position.z <= PlayerManager.endPosZ-80) {

        //only create Obstacles within a set distance (distanceCreated) from the player
        while (obstacle.transform.position.z <= PlayerManager.positionZ + distanceCreated) {
            positionX = Random.Range (minX, maxX); //calculate a random x position
            //calculate a random distance from the player
            distanceBetweenObjects = Random.Range (minDistanceBetweenObjects, maxDistanceBetweenObjects);
            posYObj = obstacle.transform.position.y - (distanceBetweenObjects * Mathf.Tan (radRotation));

            var spawnPosition = new Vector3 (positionX, posYObj, distanceBetweenObjects +
            obstacle.transform.position.z);
            obstacle = (GameObject)Instantiate (obstaclePrefab, spawnPosition, Quaternion.identity);
            NetworkServer.Spawn (obstacle);

            obstacle.transform.parent = obstacles.transform; //make the new obstacles children of the
            parent Obstacles object
            Destroy (obstacle, timeToDestroy); //destroy an obstacle after timeToDestroy seconds
            Invoke ("PlaceObstacles", 0); //do again after 3 to 5 seconds
        }
    }
}

void PlaceArcs(){
    if (arc.transform.position.z <= PlayerManager.endPosZ-80) {
        //only create Arcs within a set distance (distanceCreated) from the player
        while(arc.transform.position.z <= PlayerManager.positionZ + distanceCreated) {
            positionX = Random.Range (minX, maxX);
            distanceBetweenObjects = Random.Range (minDistanceBetweenObjects, maxDistanceBetweenObjects);
            posYObj = arc.transform.position.y - (distanceBetweenObjects * Mathf.Tan (radRotation));

            var spawnPosition = new Vector3 (positionX, posYObj, distanceBetweenObjects +
            arc.transform.position.z);
            arc = (GameObject)Instantiate (arcPrefab, spawnPosition, Quaternion.identity);
            NetworkServer.Spawn (arc);

            arc.transform.parent = arcs.transform; //make the new arcs children of the parent Arcs object
            Destroy (arc, timeToDestroy); //destroy an arc after timeToDestroy seconds
            Invoke ("PlaceArcs", Random.Range(0,1)); //do again after 3 to 5 seconds
        }
    }
    else {
        positionX = 0;
        distanceBetweenObjects = 5; // have the finish line a little before the end
        posYObj = arc.transform.position.y - (distanceBetweenObjects * Mathf.Tan (radRotation));

        var spawnPosition = new Vector3 (positionX, posYObj, distanceBetweenObjects +
        arc.transform.position.z);
        finish = (GameObject)Instantiate (finishPrefab, spawnPosition, Quaternion.identity);
        NetworkServer.Spawn (finish);
    }
}

void PlaceTreesLeft(){
    //only make trees until the position where the player is moving to
    while (treeleft.transform.position.z <= PlayerManager.positionZ + distanceCreated) {
        positionX = Random.Range (-minXTree, -maxXTree);
        distanceBetweenObjects = 10;
        posYTree = treeleft.transform.position.y - (distanceBetweenObjects*Mathf.Tan (radRotation));

        var spawnPosition = new Vector3 (positionX, posYTree, distanceBetweenObjects +
        treeleft.transform.position.z);
        treeleft = (GameObject)Instantiate (treePrefab, spawnPosition, Quaternion.identity);
        NetworkServer.Spawn (treeleft);
        treeleft.transform.parent = trees.transform; //make the new trees children of the parent
        Trees object
        Destroy (treeleft, timeToDestroyTree); //destroy a tree after timeToDestroyTree seconds
        Invoke ("PlaceTreesLeft", Random.Range (0, 1)); //do again after 0 to 1 seconds
    }
}

void PlaceTreesRight(){

```

```

//only make trees until the position where the player is moving to
while (treeright.transform.position.z <= PlayerManager.positionZ + distanceCreated) {
    positionX = Random.Range (minXTree, maxXTree);
    distanceBetweenObjects = 5;
    posYTree = treeright.transform.position.y - (distanceBetweenObjects*Mathf.Tan (radRotation));

    var spawnPosition = new Vector3 (positionX, posYTree, distanceBetweenObjects +
    treeright.transform.position.z);
    treeright = (GameObject)Instantiate (treePrefab, spawnPosition, Quaternion.identity);
    NetworkServer.Spawn (treeright);
    treeright.transform.parent = trees.transform; //make the new trees children of the parent
    Trees object
    Destroy (treeright, timeToDestroyTree); //destroy a tree after timeToDestroyTree seconds
    Invoke ("PlaceTreesRight", Random.Range (0, 1)); //do again after 0 to 1 seconds
}
}

void PlaceSnow(){
//only make snow until the position where the player is moving to
while (snow.transform.position.z <= PlayerManager.endPosZ + distanceCreated) {
    positionX = 0;
    distanceBetweenObjects = 10;
    posYObj = snow.transform.position.y - (distanceBetweenObjects * Mathf.Tan (radRotation));

    var spawnPosition = new Vector3 (positionX, posYObj, distanceBetweenObjects +
    snow.transform.position.z);
    snow = (GameObject)Instantiate (snowPrefab, spawnPosition, Quaternion.identity);
    NetworkServer.Spawn (snow);
    snow.transform.parent = snows.transform; //make the new snows children of the parent
    Snow object
    Invoke ("PlaceSnow", Random.Range (0, 1)); //do again after 0 to 1 seconds
}
}
}

```

Appendix D. Camera Collider Script

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.Networking;

public class CameraCollider : NetworkBehaviour {

    public static int count; //variable to count the score
    public static Vector3 colliderPos; //variable to store the position of the collider box
    public static bool endGame; //a boolean to determine if the end of the game is reached
    public static bool scoreOverview; //a boolean to determine when the scoreOverview can be shown
    public static int score; //new variable to communicate the counted score

    //colors and camera for extra feedback if obstacle is hit
    public Color red = Color.red;
    public Color black = Color.black;
    Camera cm;
    private float tColor;
    private float fadeTime;
    private bool obstacleHit; //a bool to communicate that an obstacle is hit and that the background
    should turn red and fade back to black

    //audio
    AudioSource correctAudio;
    AudioSource errorAudio;
    AudioSource gameEnded;
    AudiListener audiolistener;

    private bool isLocP; //boolean to store the 'isLocalPlayer' value from the parent
    public static float speed; //speed that will be changed and communicated to the playermanager
    script

    void Start () {
        count = 0;

        //get isLocalPlayer from parent
        isLocP = gameObject.transform.parent.gameObject.GetComponent<NetworkIdentity> ().isLocalPlayer;

        endGame = false; //we are not at the end of the game yet
        scoreOverview = false; //we do not show the scores yet

        speed = PlayerManager.pmSpeed;
        cm = GetComponent<Camera> ();

        AudioSource[] audios = GetComponents<AudioSource>();
        correctAudio = audios[0];
        errorAudio = audios[1];
        gameEnded = audios [2];

        tColor = 0;
        fadeTime = 1;
        obstacleHit = false;
    }

    void Update () {
        colliderPos = GetComponent<Collider>().transform.position;

        //background fade when obstacle is hit
        if (obstacleHit == true){
            tColor += Time.deltaTime / fadeTime; //count t for the fade effect
            if (tColor <= 0.75f) { //as long as the 0.75 seconds haven't passed...
                cm.backgroundColor = Color.Lerp (red, black, tColor); //... Lerp the color in 0.75 sec
            } else { // if 0.75 seconds have passed...
                tColor = 0; //... reset the values so that we can Lerp again when the new obstacle
            }
        }
    }
}
```

```

        gets hit
        cm.backgroundColor = black;
        obstacleHit = false;
    }
}

public IEnumerator OnTriggerEnter(Collider other){
    //collision between player and obstacle
    if (other.gameObject.CompareTag ("Obstacle")) {
        Destroy (other.gameObject); //destory an obstacle when it is hit
        if (endGame) {
            count = count; //do nothing to the score anymore when a player has reached the
            finish
        } else if (isLocP) {
            obstacleHit = true; //set obstacleHit to true so that the code for background fade
            can be executed in update()
            count = count - 100; //decrease count
            errorAudio.Play ();

            speed = speed - 1; //decrease speed
            if (speed <= 0) { //prevent standing still
                speed = 1;
            } else {
                yield return new WaitForSeconds (5); //wait 5 seconds
                speed = speed + 1; //set speed back to old value
            }
        }
    }
}
//collision between player and arc
else if (other.gameObject.CompareTag ("Arc")) {
    Destroy (other.gameObject); //destory an arc when it is hit
    if (endGame) {
        count = count; //do nothing to the score anymore when a player has reached the
        finish
    } else if (isLocP) {
        count = count + 100; //increase count
        correctAudio.Play ();
    }
} //if the finish is reached
else if (other.gameObject.CompareTag ("Finish")) {
    endGame = true; //game has ended
    gameEnded.Play ();
    yield return new WaitForSeconds (5); //wait 5 seconds
    scoreOverview = true; //show score overview
}
}
}

```

Appendix E. Player Manager Script

```
using System.Collections;
using UnityEngine.UI;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.Networking;

public class PlayerManager : NetworkBehaviour {

    public GameObject player;
    public Camera cam;
    public static float radRotation;

    //variables to store the position of the camera (player)
    //are used to place the canvas in front of the player
    public static float camPositionX;
    public static float camPositionY;
    public static float camPositionZ;
    public static float camRotationX;
    public static float camRotationY;
    public static float camRotationZ;
    public static float camRotationW;

    public static float degRotation; //to communicate the degrees rotation to SkiRotation script
    public static bool isLocalP; // a boolean that is set to true if this is the local player,
    is communicated to the game manager

    //variables to know how much trees can be made and when they should be destroyed
    public static int endPosZ;
    public static float positionZ;

    public float speed; //variable to store the speed at which the player moves forward (on the z-axis)
    public static float pmSpeed; //variable to bring the speed over to the camera script and back
    public int degreesRotation; //integer that stores the degrees that the slope is tilted
    public int endPositionZ; //z position that the player is moving towards
    private float endPositionY; //the y position that the player is moving towards

    void Start ()
    {
        transform.Translate(CameraCollider.colliderPos.x, 0, 0); //set the player at the right position
        to start
        posZ = 0f;
        pmSpeed = speed;
        degRotation = degreesRotation; //set degRotation so that it can be communicated to SkiRotation
        script
    }

    void Update(){
        if (!isLocalPlayer) {
            cam.enabled = false;
            (cam.GetComponent(typeof(AudioListener)) as AudioListener).enabled = false; // to prevent
            having 2 audiolisteners in the scene
            return;
        }

        endPosZ = endPositionZ; //make the endPosZ equal to the position that the player is moving to
        so that the value can be passed on to the spawner script
        positionZ = transform.position.z; //make positionZ equal to the position of the player so that
        we know when trees can be destroyed

        //move the player with arrows, for players on the PC
        var x = Input.GetAxis("Horizontal") * Time.deltaTime * 3.0f;
        transform.Translate(x, 0, 0);
    }
}
```

```

//move the player forwards at a constant speed
transform.position = Vector3.MoveTowards(transform.position,
new Vector3(player.transform.position.x, endPositionY, endPositionZ), speed*Time.deltaTime);

if (isLocalPlayer) {
    isLocalP = true; //to communicate to game manager that this is the local player
}

radRotation = degreesRotation * Mathf.Deg2Rad; //transfer rotational degrees to radians
endPositionY = -(endPositionZ * Mathf.Tan(radRotation)); //compute the y position that the
player should move towards

//store the positions and rotations of the camera for the canvas position script
camPositionX = cam.transform.position.x;
camPositionY = cam.transform.position.y;
camPositionZ = cam.transform.position.z;
camRotationX = cam.transform.rotation.x;
camRotationY = cam.transform.rotation.y;
camRotationZ = cam.transform.rotation.z;
camRotationW = cam.transform.rotation.w;

count = CameraCollider.count; //communicate the score from camera collider to here
speed = CameraCollider.speed; //update the speed with the changes from cameracollider script
}
}

```


Appendix F. Canvas Position Script

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.Networking;
using UnityEngine.UI;

public class CanvasPosition : NetworkBehaviour {
    //variables to store the x, y and z position of the player, so that the canvas can also take that
    position
    private static float positionX;
    private static float positionY;
    private static float positionZ;
    private static float rotationX;
    private static float rotationY;
    private static float rotationZ;
    private static float rotationW;

    //variables to store the position of the canvas
    private float positionXCanvas;
    private float positionYCanvas;
    private float positionZCanvas;
    private float rotationXCanvas;
    private float rotationYCanvas;
    private float rotationZCanvas;
    private float rotationWCanvas;

    private float radRotation; //variable to store the rotation of the slope

    void Start () {
        GetComponent<RectTransform> ().localPosition = new Vector3 (0, 0, 0);
        GetComponent<RectTransform> ().localRotation = new Quaternion (0, 0, 0, 0);
    }

    void Update () {
        radRotation = PlayerManager.radRotation; // get the radians that the slope is tilted
        //set the position of the canvas equal to the position of the camera (player) so that score
        will be visible at all times
        positionXCanvas = PlayerManager.camPositionX;
        positionYCanvas = PlayerManager.camPositionY - (3 * Mathf.Tan (radRotation)); //take into
        account the rotation of the slope
        positionZCanvas = PlayerManager.camPositionZ + 3; //+3 so that the canvas is a bit in front of
        the player (instead of on the player)
        GetComponent<RectTransform> ().localPosition = new Vector3 (positionXCanvas, positionYCanvas,
        positionZCanvas);
    }
}
```

Appendix G. Floating Script

```
using UnityEngine;
using System.Collections;

public class Floating : MonoBehaviour {

    public float amplitude = 0.5f;
    public float frequency = 1f;

    //variables to store the position
    Vector3 positionOffset = new Vector3 ();
    Vector3 tempPosition = new Vector3 ();

    // Use this for initialization
    void Start () {
        //store the starting position
        positionOffset = transform.position;
    }

    void Update () {
        //use a sinus to float
        tempPosition = positionOffset;
        tempPosition.y += Mathf.Sin (Time.fixedTime * Mathf.PI * frequency) * amplitude;
        transform.position = tempPosition;
    }
}
```

Appendix H. Game Manager Script

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.UI;
using UnityEngine.Networking;

public class GameManager : NetworkBehaviour {

    //Text variables
    public Text countText; // the text that displays the amount of points earned
    public Text winText; // the text that will be displayed at the end of the game
    public Text scoreOverviewText; // the text taht will be displayed at the end of the game in a score
    overview
    public Text you; // the text that says "you"
    public Text opponent; // the text that says "opponent"
    public Text opponentScore; // the text that holds the opponents score
    public Text title; // the text that says "score overview" in the score overview

    //Image variables
    public Image image; // the background for the score overview
    public Image youIsWinner; // the image that will be displayed when you win
    public Image opponentIsWinner; // the image that will be displayed when opponent wins

    private bool isLocalP;
    public int score;

    void Start () {
        //set both of the texts equal to nothing on the first frame
        SetText();
        winText.text = "";
    }

    public void Update () {
        score = CameraCollider.count; //take the countshare variable from the camera to see how much
        the score is
        SetText (); //set the count text
        isLocalP = PlayerManager.isLocalP; //get isLocalP from playermanager script
    }

    public void SetText(){
        countText.text = "Points: " + score;
        if (CameraCollider.endGame) {
            winText.text = "Game ended";
            print ("you: " + score);
            print ("opponent: " + score);
        } else {
            winText.text = "";
        }
    }

    if (CameraCollider.scoreOverview) {
        scoreOverviewText.text = score + " points";
        image.enabled = true;
        title.enabled = true;
        you.enabled = true;
        opponent.enabled = true;
        opponentScore.enabled = true;

        //fake score overview where opponent has reached 700 points
        if (score > 700) {
            scoreOverviewText.color = new Color (0, 255, 0); //make green if he is the winner
            you.color = new Color (0, 255, 0); //make green if he is the winner
            youIsWinner.enabled = true;
        } else {
            opponent.color = new Color (0, 255, 0); //make opponent green if he is the winner
        }
    }
}
```

```
        opponentScore.color = new Color (0, 255, 0); //make opponent green if he is the winner
        opponentIsWinner.enabled = true;
    }
    //if no score overview can be shown, disable all its elements
} else {
    scoreOverviewText.text = "";
    image.enabled = false;
    title.enabled = false;
    you.enabled = false;
    opponent.enabled = false;
    opponentScore.enabled = false;
    youIsWinner.enabled = false;
    opponentIsWinner.enabled = false;
}
}
}
```

Appendix I. Network Manager Script

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.Networking;
using UnityEngine.Networking.Match;

public class NwManager : NetworkBehaviour {

    NetworkMatch matchMaker;
    public NetworkManager nwManager;

    void Start () {
        nwManager.StartHost (); //if you want the device to be the host
        //nwManager.StartClient(); //if you want the device to be the client
    }
}
```

Appendix J. Obstacle Script

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class ObstacleScript : MonoBehaviour {

    public float speed;
    public GameObject obstacle;

    void Start () {
    }

    void Update () {
        transform.Rotate (new Vector3 (15, 30, 45) * Time.deltaTime); //rotates the obstacle
    }
}
```

Appendix K. Ski Rotation Script

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class SkiRotation : MonoBehaviour {

    public float rotationX;

    void Start () {
        rotationX = PlayerManager.degRotation; //get the rotation of the slope
        transform.Rotate (-rotationX, 0, 0); //rotate the skies along with the slope, so that it looks
        like they're placed on the slope
    }

    void Update () {
    }
}
```

Appendix L. Questionnaire First Round of User Tests

Gender: Male Female
Age: _____
Skiing experience: Moderate Experienced Very experienced

Circle how much each symptom below is affecting you right now

General discomfort	None	Slight	Moderate	Severe
Fatigue	None	Slight	Moderate	Severe
Headache	None	Slight	Moderate	Severe
Eye strain	None	Slight	Moderate	Severe
Difficulty focussing	None	Slight	Moderate	Severe
Salivation increasing	None	Slight	Moderate	Severe
Sweating	None	Slight	Moderate	Severe
Nausea	None	Slight	Moderate	Severe
Difficulty concentrating	None	Slight	Moderate	Severe
Fullness of the head	None	Slight	Moderate	Severe
Blurred vision	None	Slight	Moderate	Severe
Dizziness with eyes open	None	Slight	Moderate	Severe
Dizziness with eyes closed	None	Slight	Moderate	Severe
* Vertigo	None	Slight	Moderate	Severe
** Stomach awareness	None	Slight	Moderate	Severe
Burping	None	Slight	Moderate	Severe

** Vertigo is experienced as loss of orientation*

*** Stomach awareness is usually used to indicate a feeling of discomfort which is just short of nausea*

For each of the following statements, please indicate how true it is for you

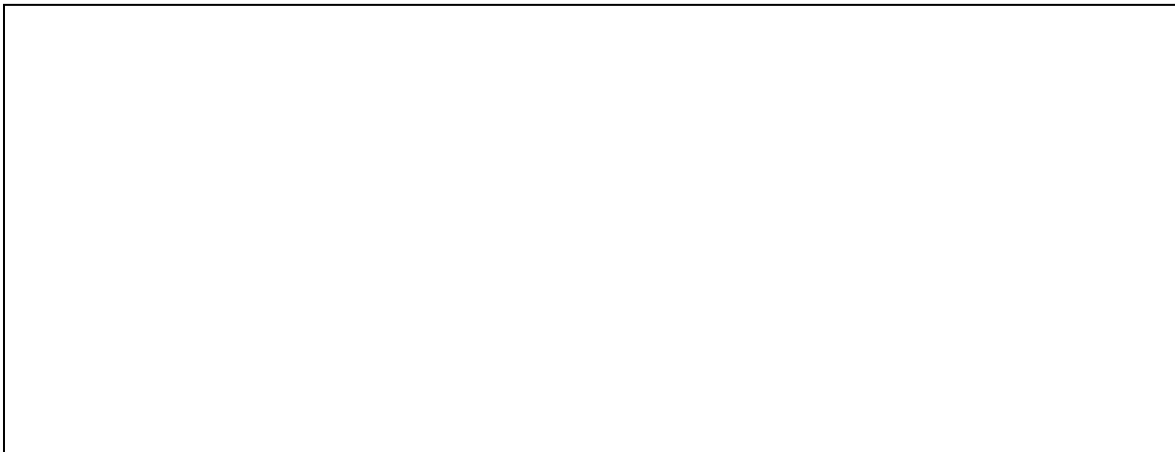
	Not at all true		Somewhat true			Very true	
I enjoyed playing the game very much	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Playing the game was a fun activity to do.	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
I thought playing the game was a boring activity.	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
The game did not hold my attention at all.	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
I would describe playing this game as very interesting.	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
I thought the game was quite enjoyable.	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
While I was playing the game, I was thinking about how much I enjoyed it.	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
I think I am pretty good at this game.	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
I think I did pretty well at this game, compared to others.	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
After playing this game for a while, I felt pretty competent.	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
I am satisfied with my performance at this game.	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
I was pretty skilled at this game.	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>

This was an game that I couldn't play very well.	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
I put a lot of effort into this game.	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
I didn't try very hard to do well at this game.	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
I tried very hard on this game.	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
It was important to me to do well at this game.	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
I didn't put much energy into this game.	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
The actions I performed in the physical world had influence on the game.	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
There was no real correlation between my actions in the physical world and the game.	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
The multiplayer aspect of the game motivated me.	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
I liked the fellow player.	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
I did not notice the fellow player.	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
I would like to play this game again.	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
This game is useful for teaching people how to ski.	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
I think playing this game more regularly would help me improve my skiing skills.	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>

Is there anything you missed in the game?



If you could choose anything, what would you like to add or change to the game? (you are allowed to give multiple answers)



Appendix M. Questionnaire Second Round of User Tests

Gender: Male Female

Age: _____

Skiing experience: Not so experienced Moderate Experienced Very experienced

Circle how much each symptom below is affecting you right now

General discomfort	None	Slight	Moderate	Severe
Fatigue	None	Slight	Moderate	Severe
Headache	None	Slight	Moderate	Severe
Eye strain	None	Slight	Moderate	Severe
Difficulty focussing	None	Slight	Moderate	Severe
Salivation increasing	None	Slight	Moderate	Severe
Sweating	None	Slight	Moderate	Severe
Nausea	None	Slight	Moderate	Severe
Difficulty concentrating	None	Slight	Moderate	Severe
Fullness of the head	None	Slight	Moderate	Severe
Blurred vision	None	Slight	Moderate	Severe
Dizziness with eyes open	None	Slight	Moderate	Severe
Dizziness with eyes closed	None	Slight	Moderate	Severe
* Vertigo	None	Slight	Moderate	Severe
** Stomach awareness	None	Slight	Moderate	Severe
Burping	None	Slight	Moderate	Severe

* Vertigo is experienced as loss of orientation

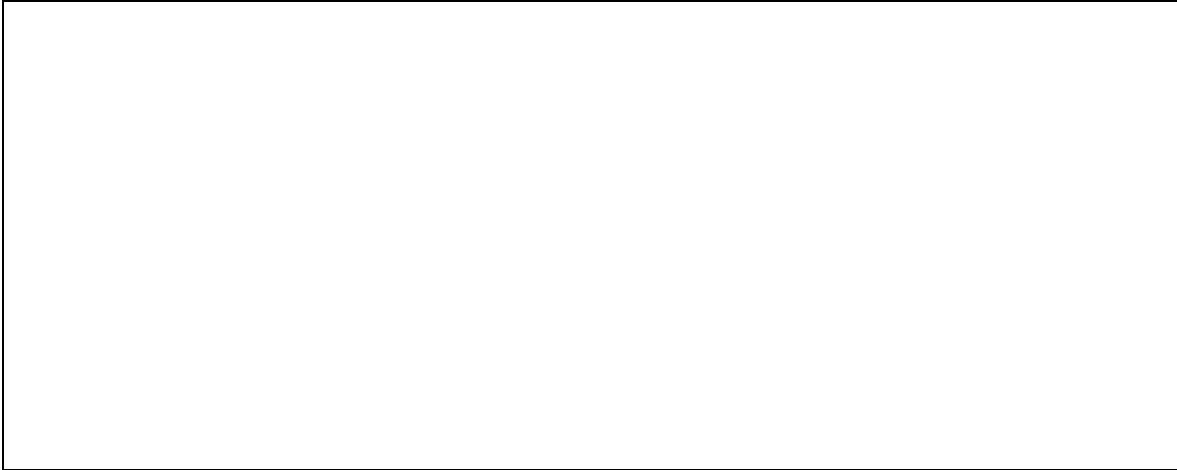
** Stomach awareness is usually used to indicate a feeling of discomfort which is just short of nausea

For each of the following statements, please indicate how true it is for you

	Not at all true		Somewhat true			Very true	
I prefer last week's version of the game over this week's version.	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
I did not notice any difference between the two versions of the game.	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
I prefer this week's version of the game over last week's version.	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
This version of the game made me more aware of my mistakes.	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
This version of the game made me more aware of the things I did right.	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
In this version of the game I felt more distracted.	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
This version of the game was more clear to me	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>

Were there differences between this version of the game and the version you played last week? If yes, please list the differences.

Which version of the game do you prefer, and why?



What would you still like to change or add to the game?



References

- [1] R. Baptista, A. Coelho, and C. V. de Carvalho, "Relationship Between Game Categories and Skills Development: Contributions for Serious Game Design," *Proc. Eur. Conf. Games-based Learn.*, vol. 2015–Janua, no. October, pp. 656–663, 2015.
- [2] S. Göbel, S. Hardy, V. Wendel, F. Mehm, and R. Steinmetz, "Serious games for health," *Proc. Int. Conf. Multimed. - MM '10*, no. October, p. 1663, 2010.
- [3] F. Ke, "Designing and integrating purposeful learning in game play: a systematic review," *Educ. Technol. Res. Dev.*, vol. 64, no. 2, pp. 219–244, 2016.
- [4] H. Tannous, D. Istrate, M. C. Ho Ba Tho, and T. T. Dao, "Serious game and functional rehabilitation for the lower limbs," *Eur. Res. Telemed. / La Rech. Eur. en Télémedecine*, vol. 5, no. 2, pp. 65–69, 2016.
- [5] S. Hardy, T. Dutz, J. Wiemeyer, S. Göbel, and R. Steinmetz, "Framework for personalized and adaptive game-based training programs in health sport," *Multimed. Tools Appl.*, pp. 5289–5311, 2014.
- [6] W. B. Cieślński, J. Sobiecki, P. A. Piepiora, Z. N. Piepiora, and K. Witkowski, "Application of the Augmented Reality in prototyping the educational simulator in sport - the example of judo," *J. Phys. Conf. Ser.*, vol. 710, p. 12016, 2016.
- [7] J. Westlin and T. H. Laine, "Calory Battle AR: an Extensible Mobile Augmented Reality Platform," *IEEE World Forum Internet Things*, pp. 171–172, 2014.
- [8] A. Mader and W. Eggink, "A Design Process for Creative Technology," no. September, 2014.
- [9] H. Sharp, A. Finkelstein, and G. Galal, "Stakeholder Identification in the Requirements Engineering Process," pp. 1–5.
- [10] K. Eason, *Information Technology and Organisational Change*. Taylor & Francis, 1987.
- [11] C. Wilson, *Brainstorming and beyond : a user-centered design method*. Morgan Kaufmann, 2013.
- [12] "Use-cases - An approach to capturing and describing software requirements and basis for use-case driven development," 2014. [Online]. Available: http://www.comp.dit.ie/rlawlor/Soft_Eng/UML/Use-cases.pdf. [Accessed: 28-Apr-2017].
- [13] A. Cockburn, "Writing Effective Use Cases," *Work*, vol. 26, no. 1, p. 94, 2001.
- [14] Jesse Schell, *The art of game design*, vol. 1. 2008.
- [15] K. Waters, "Prioritization using MoSCoW," 2009. [Online]. Available: https://cs.anu.edu.au/courses/comp3120/local_docs/readings/Prioritization_using_MoSCoW_AllAboutAgile.pdf. [Accessed: 26-Apr-2017].
- [16] R. S. Kennedy, N. E. Lane, K. S. Berbaum, and M. G. Lilienthal, "Simulator Sickness Questionnaire: An Enhanced Method for Quantifying Simulator Sickness," *Int. J. Aviat. Psychol.*, vol. 3, no. 3, pp. 203–220, 1993.
- [17] R. van Delden, "(Steering) Interactive Play Behavior," University of Twente, 2017.
- [18] "Intrinsic Motivation Inventory (IMI)," no. Imi, 1994.
- [19] A. Whitehead, H. Johnston, N. Nixon, and J. Welch, "Exergame effectiveness: what the numbers can tell us," *Proc. 5th ACM SIGGRAPH Symp. Video Games - Sandbox '10*, no. October, pp. 55–62, 2010.

- [20] R. W. Lindeman, G. Lee, L. Beattie, H. Gamper, R. Pathinarupothi, and A. Akhilesh, "GeoBoids: A mobile AR application for exergaming," *11th IEEE Int. Symp. Mix. Augment. Real. 2012 - Arts, Media, Humanit. Pap. ISMAR-AMH 2012*, pp. 93–94, 2012.
- [21] F. Anderson, T. Grossman, J. Matejka, and G. Fitzmaurice, "YouMove: Enhancing Movement Training with an Augmented Reality Mirror," *Proc. 26th Annu. ACM Symp. User interface Softw. Technol. - UIST '13*, pp. 311–320, 2013.
- [22] D. Reidsma, E. Dehling, and H. Welbergen, "Leading and following with a virtual trainer," *4th Int. Work. Whole Body Interact. Games Entertain.*, p. 4, 2011.
- [23] M. S. Hossain, S. Hardy, A. Alamri, A. Alelaiwi, V. Hardy, and C. Wilhelm, "AR-based serious game framework for post-stroke rehabilitation," *Multimed. Syst.*, vol. 22, no. 6, pp. 659–674, 2016.
- [24] N. Iten and D. Petko, "Learning with serious games: Is fun playing the game a predictor of learning success?," *Br. J. Educ. Technol.*, vol. 47, no. 1, pp. 151–163, 2016.
- [25] Z. Z. Li, Y. B. Cheng, and C. C. Liu, "A constructionism framework for designing game-like learning systems: Its effect on different learners," *Br. J. Educ. Technol.*, vol. 44, no. 2, pp. 208–224, 2013.
- [26] M. C. Swartz and E. J. Lyons, "Whats's the Point?: A Review of Reward Systems Implemented," vol. 5, no. 2, 2016.
- [27] D. H. Goh, E. P. P. Pe-than, and C. S. Lee, "Perceptions of virtual reward systems in crowdsourcing games," *Comput. Human Behav.*, vol. 70, pp. 365–374, 2017.
- [28] C. Cruz, M. D. Hanus, and J. Fox, "The need to achieve: Players' perceptions and uses of extrinsic meta-game reward systems for video game consoles," *Comput. Human Behav.*, vol. 71, pp. 516–524, 2017.
- [29] E. L. Deci and R. M. Ryan, "Self-Determination Theory : A Macrotheory of Human Motivation , Development , and Health," vol. 49, no. 3, pp. 182–185, 2008.
- [30] S. Bruck and P. A. Watters, "The factor structure of cybersickness," *Displays*, vol. 32, no. 4, pp. 153–158, 2011.
- [31] H. B. Duh, D. E. Parker, and T. A. Furness, "An ' Independent Visual Background ' Reduced Balance Disturbance Evoked by Visual Scene Motion : Implication for Alleviating Simulator Sickness," pp. 3–8, 2001.
- [32] L. Rebenitsch and C. Owen, "Review on cybersickness in applications and visual displays," *Virtual Real.*, vol. 20, no. 2, pp. 101–125, 2016.
- [33] P. Milgram, "Perceptual Issues in Augmented Reality," no. December, 2013.
- [34] G. E. Riccio and T. A. Stoffregen, "An Ecological Theory of Motion Sickness and Postural Instability," no. September, 1991.
- [35] S. M. Ebenholtz, "Motion sickness and oculomotor systems in virtual environments," *Presence Teleoperators Virtual Environ.*, vol. 1, no. 3, pp. 302–305, 1992.
- [36] R. S. Kennedy and J. E. Fowlkes, "What does it mean when we say that 'simulator sickness is polygenic and polysymptomatic'?" *IMAGE V Conf.*, 1990.
- [37] "Infinite Slope - Indoor Ski USA," 2017. [Online]. Available: <http://www.indoorskiusa.com/start/our-solutions/infinite-slope>. [Accessed: 30-May-2017].
- [38] A. Seth, J. M. Vance, and J. H. Oliver, "Virtual reality for assembly methods prototyping : a review," 2010.

- [39] C. Cruz-neira, D. J. Sandin, and T. A. Defanti, "Surround-Screen Projection-Based Virtual Reality : The Design and Implementation of the CAVE," pp. 135–142, 1993.
- [40] Z. O'Connor, "Colour psychology and colour therapy: Caveat emptor," *Color Res. Appl.*, vol. 36, no. 3, pp. 229–234, 2011.
- [41] Vuforia, "Vuforia SDK." [Online]. Available: <https://developer.vuforia.com/downloads/sdk>. [Accessed: 25-Apr-2017].
- [42] M. Hallberg, "How To MARKERLESS Augmented Reality App Tutorial for Beginners with Unity 3D," 2016. [Online]. Available: https://www.youtube.com/watch?v=T6bd_MQ2ass. [Accessed: 01-May-2017].
- [43] Unity Technologies, "Unity Learn - Multiplayer Networking." [Online]. Available: <https://unity3d.com/learn/tutorials/topics/multiplayer-networking>. [Accessed: 16-May-2017].
- [44] Tango, "Tango Developer Overview," 2017. [Online]. Available: <https://developers.google.com/tango/apis/unity/unity-simple-ar>.
- [45] ARToolKit, "ARToolKit for Unity," 2016. [Online]. Available: https://artoolkit.org/documentation/doku.php?id=6_Unity:unity_about. [Accessed: 25-Apr-2017].
- [46] Unity Technologies, "Unity." [Online]. Available: <https://unity3d.com/unity>. [Accessed: 25-Apr-2017].
- [47] Unity Technologies, "Unity Manual - GameObjects," 2017. [Online]. Available: <https://docs.unity3d.com/Manual/GameObject.html>. [Accessed: 25-Apr-2017].
- [48] Epson, "Moverio BT-300 Smart Glasses - Developer Network." [Online]. Available: <https://tech.moverio.epson.com/en/bt-300/index.html>. [Accessed: 20-Apr-2017].
- [49] Unity Technologies, "Unity Manual - Using the Network Manager," 2017. [Online]. Available: <https://docs.unity3d.com/Manual/UNetManager.html>. [Accessed: 08-May-2017].
- [50] H. S. Oluwatosin, "Client-Server Model," *IOSR J. Comput. Eng.*, vol. 16, no. 1, pp. 57–71, 2014.
- [51] Unity Technologies, "Unity Manual - Multiplayer and Networking - NetworkIdentity," 2017. [Online]. Available: https://docs.unity3d.com/Manual/class-NetworkIdentity.html?_ga=2.109135237.1469161793.1497193659-790851949.1487956469. [Accessed: 08-May-2017].
- [52] Unity Technologies, "Unity Manual - Multiplayer and Networking - Object Spawning," 2017. [Online]. Available: <https://docs.unity3d.com/Manual/UNetSpawning.html>. [Accessed: 12-May-2017].
- [53] Microsoft, "Hololens hardware details." [Online]. Available: https://developer.microsoft.com/en-us/windows/mixed-reality/hololens_hardware_details#what_you_need_to_develop. [Accessed: 18-May-2017].