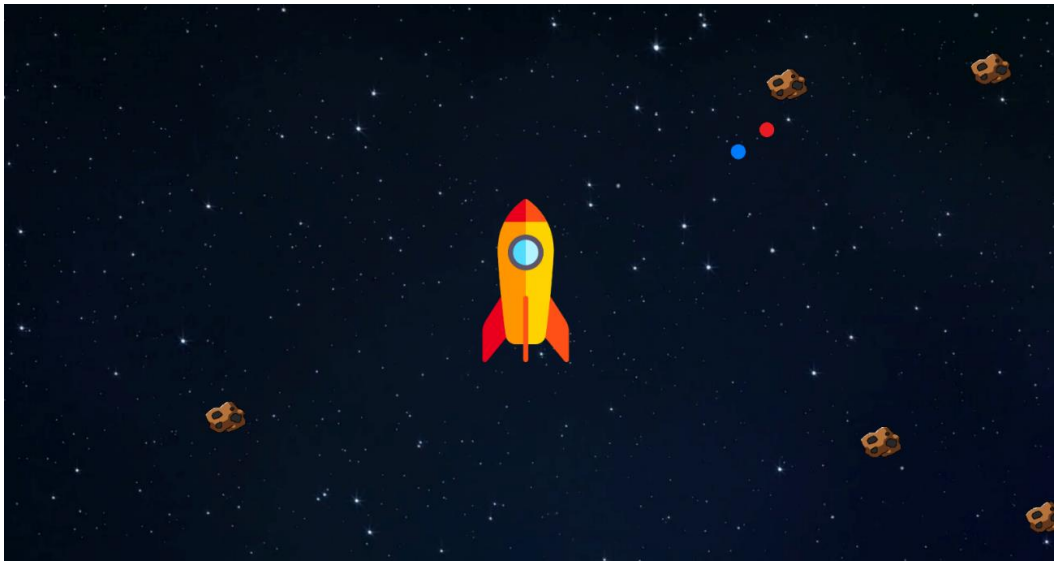
Western Norway University of Applied Sciences

Improvement of Games using Eye Tracking for Oculomotor Training

Forbetring av spel som brukar augesporing for okulomotorisk trening



Gudsteinn Arnarson, Kristian Eliassen, and Stian Grønås

Bachelor, Computer Engineering

Department of Computing, Mathematics and Physics

Faculty of Engineering and Science

Ilona Heldal and Qasim Ali

04.06.2021

Report title: Improving Games using Eye Tracking for Oculomotor Training	*Date: 04.06.21*
Author(s): Gudsteinn Arnarson, Kristian Eliassen, Stian Grønås	*Number of pages: 60* *Number of appendix pages: 8*
Education programme: Computer Engineering	*Number of CDs/DVDs:* 0
Contact person at HVL: Ilona Heldal	*Classification:* Open
Remarks: Done in the COVID-pandemic	

External company/contact: Høgskulen på Vestlandet / Western Norway University of Applied Sciences	*External contact reference:*
External contact person: Ilona Heldal	*Phone:* +47 55 58 75 24

Summary:

This project deals with 6 serious games made in Unity using Eye Tracking to train eye movement. The games were previously made through a Master's Thesis. The project aims to improve the games by saving data from them, adding new user interfaces and implementing a replay function to the games.

Dette prosjektet omhandlar 6 seriøse spel som er utvikla i Unity-motoren ved bruk av augesporing til å trene augerørsle. Spela blei utvikla i ein tidlegare masteroppgåve. Prosjektets mål er å forbetre spela ved å lagre data frå dei, leggje til eit nytt grensesnitt og implementera ein replay-funksjon.

Keywords:

Eye Tracking	Vision problems	Serious Games
Unity	Oculomotor dysfunction	Replay
User interfaces	Vision training	Tobii

Western Norway University Of Applied Sciences, Faculty of Engineering and Science
Postal address: Postboks 7030, 5020 BERGEN Address: Inndalsveien 28, Bergen
Phone: 55 58 75 00 Fax: 55 58 77 90 E-mail: post@hvl.no Home page: http://www.hvl.no

PREFACE

The group would like to thank Ilona Heldal for all the help and guidance throughout the project with the goals and project definition, writing the report and more.

Secondly, the group would like to thank Qasim Ali for all the help with the project, especially with programming and in understanding more about Eye Tracking and Unity, as well as help with analysis of the eye data.

The group would also like to thank the expert in screening and training oculomotor problems for participation and users who helped with evaluations of the program and for giving valuable feedback.

Additionally, the group would like to thank Atle Birger Geitung, Carsten Gunnar Helgesen and Henrik Borgli for support.

Finally, the group thanks our families and friends for emotional support.

TABLE OF CONTENT

1 INTRODUCTION

1.1 Motivation and goal

1.1.1 Motivation

While eye testing is part of obligatory health checks of children around the age of 5, this does not include functional vision screening. Late, when children begin their school education, their vision, and functional vision, is not tested either. In general, a large number of school-aged children have vision problems but might not get diagnosed at all throughout their life, since their problem is not visible. Approximately 20% of children have non-detected vision problems (Falkenberg, Langaas and Svarverud, 2019). Vision problems, in general, may cause a number of societal problems later, e.g., problems in academic progress and integration difficulties. If a young child would get diagnosed with oculomotor deficiency (OMD), a functional vision problem, vision training can help them (17-30%) (Falkenberg, Langaas and Svarverud, 2019). However, to identify these 17-30%, including undetected vision problems, would require screening all children's vision. Screening all school aged children would require huge societal resources, and a number of experts involved which are not available today.

Many functional problems, as OMD, are due to coordination deficit in the eye muscles between the left and right eyes. As all muscle problems, this can be measured by following the left and the right eye movements precisely enough with help of many modern and affordable Eye Tracking technologies.

Developing a cheap and easy to use technology-based solution for both specialists and non-specialists enabling to discover OMD more easily in children would allow vision screening for many people. However, screening alone is not enough. If there are not enough societal resources for screening, there is not enough for helping all children with problems. Therefore, the help from technologies to support training would also be appreciated. Since the target audience for this work are school aged children, easily applicable technologies would be required, both for screening and training. Serious games and gamification promise to help screening and training in many areas. While there are some technologies for screening, there are not many existing tools for supporting training functional eye problems on the market. Existing games and tools for screening and rehabilitation of OMD problems are expensive and often use expensive and proprietary equipment (RightEye, 2021) or are closely connected to optometric vision therapy programs (Haraldseth Software, 2021).

1.1.2 Problem Description

Through a previous Master Thesis there has been developed 6 serious games in the game engine Unity which uses a Tobii Eye Tracker for training oculomotor problems. The games are initial prototypes, but they do not save and store any relevant data from the games after a patient has

used them (Pettersen, 2018). So, one does not have an application to see the effects of the playing or what the patients achieved during playing the games. This would be necessary information, and helpful to see whether the games are helping with the training and how they are helping with improvement. To be used purposeful, the games would also be required to have more settings and challenges motivating the players to play more in the correct way.

There are several research questions interesting for this research:

How can the games be further developed for more proper eye training? How can the Eye Tracking equipment and Tobii's Pro SDK be used to for this development? How can a suitable interface be developed to manage handling of the games, both for playing and overview training? How can Unity be utilized to show the results and other information in relation to using the games? How can measurable data (e.g., on basic eye movements, such as fixations and other data) be collected with the Eye Tracker? How to save this data in a database for further use? What information is relevant for the different users of the program?

1.1.3 Goal

The overall goal of this project is to contribute to further develop serious games using Eye Tracking technology for helping persons with OMD with training. The immediate issues the group need to solve relates to developing a user interface for handling serious games, further improving the six earlier prototypes, and integrating these into the interface. The interface will allow vision teachers to examine results and handle the program to help school-aged children with training exercises with their OMD to improve their visual abilities.

Create a database. The first subgoal is to expand the program to save information and data from the games by using an Eye Tracker in a database. This is an important subgoal because without saving the data it is not possible for a vision teacher to retrieve user data and see whether the user has improved with the training during a time period.

Develop a user interface. This can help the users in choosing their own profile so data can be stored under that. This is important for the vision teachers so they get an informatic overview of the relevant data saved from the games that help define training and further therapy for the patient.

Visualizing eye-activities during playing. A third subgoal is to include a replay function so a specialist can view the patient's eye positions superimposed on top of the game exactly as the patient played it. This could also help with determining whether the training is working and could also help for seeing issues with vision / OMD.

Improve existing games. The next subgoal is to improve the existing games regarding game elements, e.g., better score-systems and also allowing measurements to see whether the user's vision is improving. The group also developed a new game important for vision training.

Examine feasibility. Test that the program is working and its functionality and perform user tests to determine further requirements for development and user opinions.

1.2 Context

This project is owned by HVL / Western Norway University of Applied Sciences. Besides of the earlier MSc thesis there are several works in the environment influencing this Thesis:

A previous software developed at HVL through a Master's Thesis is C&Look (Eide and Watanabe, 2017), and this is a software using Eye Tracking technology to identify OMD. It has several small tasks for measuring different OMD issues and one can see the results with a graph for the eye data collected. C&Look is made for identifying and can be used for checking if the person's vision has become better but is not well suited for vision training. The software is also very complete and robust but might still need some small features (Eide et al., 2019a). Also done through a Master's Thesis at HVL is 6 serious games using Eye Tracking for OMD training and therapy, and this is what this project will continue to work on. The 6 games were developed in collaboration with vision teachers to find what tasks could be made into computer games (Petterson, 2018, p. ii).

There has been several projects and research into using Eye Tracking for OMD (Eide et al., 2019a, Eide et al., 2019b, Ali et al., 2020, Heldal et al., 2021) that influenced this project. Additionally, there are also products from companies which inspired this work. Such a company is RightEye which has its own Eye Tracking hardware and has solutions for diagnosing and training of vision (RightEye, 2021). Tobii Pro has developed several Eye Trackers and a software development kit for the hardware they sell that are suited for vision research use (Tobii Pro, 2021).

1.3 Limitations

Since the COVID pandemic is still active, there will be a lack of test persons which has OMD the group can test the games on. However, there are some people that it would be possible to test the program, at the school campus and outside of it. The bigger limitation is that it is outside the scope of this project to do a larger test and evaluation of whether the games will help with training and therapy over time. Also, whether the data the program collects and later shown to vision teachers, is relevant and helpful in determining whether the patient has improved. This would require more knowledge of the theory in the field and would require more time which the group does not have for this project.

Another limitation is whether the data the program stores can be used to measure more concrete oculomotor data like fixation, saccades, smooth pursuit and more, this is something the group will investigate but will still be a limitation for the project because of time constraints.

Adding more games to get more variety and doing large improvements on the games to make them more fun and engaging, will also not be feasible for this project because of time constraints.

One final limitation is that the group is not that well versed in Unity, a program used by many for developing games in general. So, development with Unity and game related programming might take longer time and be more difficult for the group.

1.4 Resources

For tracking eye movements and data, the group needs an Eye Tracker that is supported by the Tobii Pro SDK. One Tobii Eye Tracker 4C with a Tobii Pro Upgrade Key is being lent by HVL to the group for use in developing and testing the program. Unity will be used for developing the interfaces and the games, and for running and debugging the program. Visual Studio 2019 is being used for writing the Unity scripts. GitHub is used for version control and sharing of the project files between the group members. The GitHub repository is private since the program will not be open source at this stage of the development. The Tobii Pro SDK will be used to connect and communicate with the Eye Tracker from the program. Entity Framework Core and SQLite are being used to store information and data in a local database as a file with the program. The group will also need access to the existing source code for the games, and each member will use its own computer with Windows 10 to do the work on. The group will also need access to the source code for the C&Look program for inspiration and help in developing the program.

A Microsoft SharePoint site will be used for writing the blog, and Microsoft Planner will be used for listing and distributing the different tasks that need to be done as the project goes on. Lastly the group has a private Discord server for communication internally in the group, and Zoom is used for remote meetings within the group and also with the supervisor.

1.5 Organization of the report

Chapter 1 introduces what the project is about, the problem and the motivations. It is also describing resources needed and limitations.
Chapter 2 describes the project owner, what previous work the project is built on, the requirements and initial solution idea for the project. It summarises some papers and shortly explains how Eye Trackers work, what OMD is, and what existing tools are on the market to support vision training.
Chapter 3 explains the project design, specification, and chosen tools. It is also describing used methods for development and evaluations.
Chapter 4 describes in detail the design for the program.

Chapter 5 is about the groups evaluations methods and results to check if the completed project meets the goals.
Chapter 6 is a short discussion of the work done throughout the project.
Chapter 7 concludes the project and lists items that needs to be worked further in the future.
Chapter 8 lists the references used in the report.
Chapter 9 is the appendix.

2 PROJECT DESCRIPTION

2.1 Practical background

2.1.1 Project owner

The owner of the product that the group is developing is HVL. Ilona Heldal, the supervisor of the project, works closely with the group to help them understand her vision and what needs to be done for the bachelor project together with the second supervisor, Qasim Ali, responsible for technical development. Ilona and Qasim are passionate about developing technologies for both diagnosing and training school aged children with OMD. The project that the students have set themselves out to do is only a part of a bigger goal but important, nonetheless.

2.1.2 Previous work

As previously mentioned, the students are improving 6 games developed earlier in a previous MSc Thesis (Pettersen, 2018). These games are all tailored to be used to complement training children and people with OMD. The 6 games are:

Game one is set up with two words in a fixed position on the screen. The player's task is to look at each word and then press one of two buttons determining if they think the words are alike or not.

In the second game the player can see their eye positions on the screen in the form of two circles, one blue and one red. The player must look carefully through the labyrinth without touching the walls with their eyes, and at the end reach a goal with the eye positions. To get a point one must first reach the goal and then travel back to the start position.

For the third game there is a spaceship placed in the middle of the screen that the player must protect from asteroids. The asteroids will appear at a random place on the edge of the screen and will move towards the spaceship. By looking at an asteroid the player will destroy it and save the spaceship.

The fourth game has the player presented with a crowd of objects represented by men or cats. The player must look through the crowd to find the target represented by a woman or a dog. When the player spots the target, the game will show a new crowd, and the player must seek a new target. The player will win after spotting the target 5 times.

In the fifth game, the player's task is to focus on a moving ball on the screen. As the game progresses, there will be a sudden distraction of a figure showing up on the screen. The players score will increase when the player focuses on the ball. After achieving 2000 points the player wins the game.

Finally in game six, the player is tasked with looking at several different figures scattered across the screen, remembering them within a time frame, and then be presented with the same figures aligned on the screen with an additional figure which was not presented to the player on the first screen. Then the player must choose what figure was not shown in the first picture of the game to win.

2.1.3 Initial requirements specification

The requirements are divided into three parts; (i) improve the games and save data in a database, (ii) make a new user interface for the program, (iii) make a replay function for the played games.

Improving the games and saving data involves adding new scores and fixing existing scores in the different games and saving this data in a database. Further, eye positions for left and right eye, time for each eye position, the date and how long the user played should be saved in a database. Additionally, to add more settings and difficulties that can be changed for each game would be beneficial.

For making new user interfaces there should be an option so that the user can create a new user profile and then choose that she or he wish to play games with or show the results from previous games. For playing games the existing interface should be used. In showing the results the program should show the different dates the user played, and what games were played on that date, and then one can select a specific game that was played and show relevant information from that game. The user should also be able to specify how much information should be shown, for example a specialist should be shown more advanced information than a patient. A graph to show the eye data for those sessions should also be there.

Finally, for the replay function, one should be able to play back a game that a patient had previously done, exactly as what that person saw on the screen and then superimpose the eye positions over the screen so one can see what the user looked in the game session.

Another smaller requirement that was added was to try to measure and analyse the eye data for concrete information giving indication about the status of vision impairment. This type of information is fixations, saccades, and smooth pursuit, but also the distance and time travelled by the eyes in horizontal, vertical, and diagonal distances.

2.1.4 Initial solution idea

Ilona and Qasim suggested that the group should improve the games using Unity. By saving data from the games the group had to extract the existing data that is gathered during the gameplay sessions and save it in a database file locally on the computer

instead of a server would be easier for a non-technical user as there is no hassle with connecting to an external server and users can play the games offline without the need of an internet connection. The Tobii Pro SDK was already used in the games and can be used to retrieve eye data from each eye, as well as the time for that point.

For the interface, the group had a look into Unity and its tools for creating interfaces like Unity UI toolkit.

To make a universal replay function the group needed to use the stored eye data from the database and play the game again by using the eye data as input superimposed to the game. This would also save a lot of space instead of recording video of each game session. Since each game has some sort of score and events happening like pressing a button, the group also needed to save specific data in conjunction with the universal data for each game.

2.2 Literature background

As the group is working on a previous MSc project, looking through that Thesis is important to understand what has been done and what the results were. The goal for the MSc project was to design and develop games based on oculomotor training with the use of an Eye Tracker. The results were that the games could be useful for vision training as the games, excluding game 6, were tested on 5 vision teachers. However, the games would not replace current methods of vision training but instead be added if the games prove to improve the oculomotor vision (Pettersen, 2018, p. ii, p. 64-66) and complement a whole training battery used by vision teachers and optometrists. This battery involves games using the whole place, they need to be played often and can be quite boring for young children.

The games that were developed for the MSc Thesis (see Petterson, 2018) were also included in a study about supporting school aged children to train their vision by using serious games developed by computer engineers together with vision teachers. The study has the games evaluated by five vision teacher students. The results from evaluating the games were positive, however some games needed more settings where one can choose the level of difficulty. The games also needed a better description of what the game was about. The games are good but not a good replacement for existing training methods for visual training (Heldal et al., 2021).

A lot of inspiration for improving the games comes from the MSc Thesis where C&Look was developed. Although this software was developed to detect OMD, it was important to look at the results and include similar traits to the games that are to be improved. The results are that it is possible to use an inexpensive Eye Tracker to detect some OMD (Eide and Watanabe, 2017).

2.3 Eye Tracking

Tobii has relatively inexpensive gaming-oriented Eye Trackers that also has an SDK for research, like the EyeX and the 4C models used in earlier studies (Tobii, 2021). To work with these models require a purchase of a Tobii Pro Upgrade Key to access extended data access like eye data for left and right eye separately (Tobii Pro, 2018). While there are cheaper newer models aimed to be used for games, like the Tobii Eye Tracker 5 and 5L, these cannot be upgraded to track each eye separately by using the Tobii Pro Upgrade Key. Newer eye trackers that support this upgrade key or Tobii Pro directly are very expensive.

The group uses the Tobii Eye Tracker 4C which is primarily made for gaming purposes. The Eye Tracker is fitted with two cameras and two infrared lights. Each camera and light focus on each eye. The infrared light lights up the eye of the user and then the camera can track where the eye is looking on the screen. Figure 2.1 illustrates how the Eye Tracker works.

The data that the eye tracker gathers is x- and y-positions for each eye, as well as the time. The tracker tracks this information at 90 hertz.

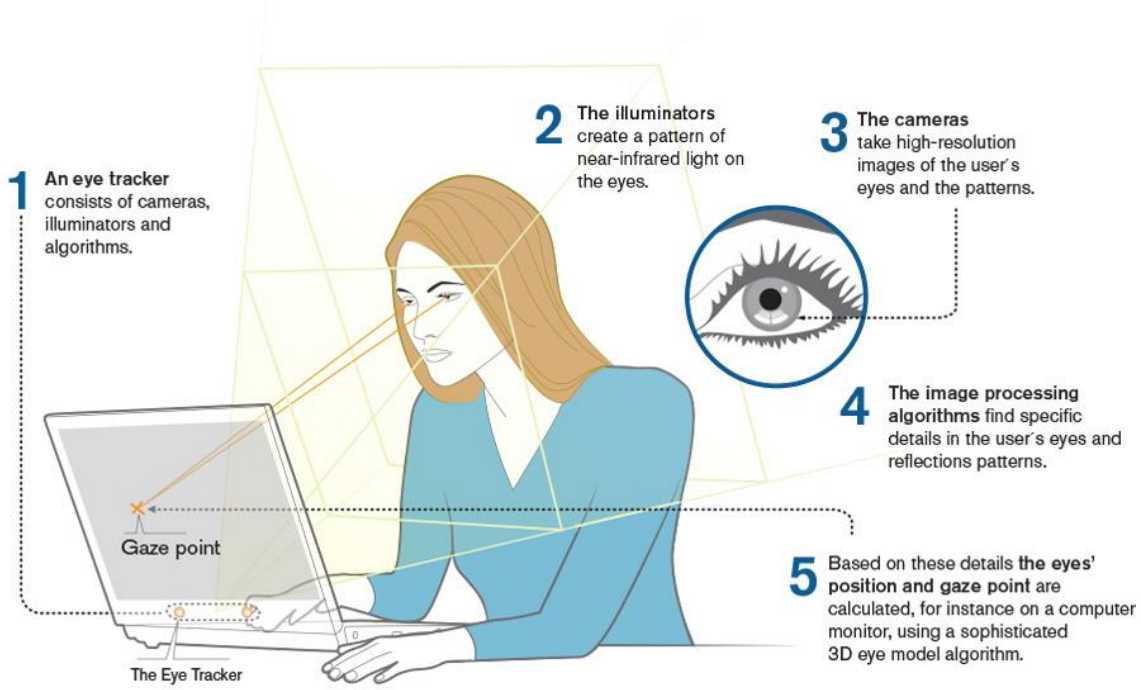


Figure 2.1 – How eye tracking works (Available at: https://www.tobiipro.com/imagevault/publishedmedia/cbkep390ahnt90u2dy75/How_DoesEyetrackingWork_ScreenBased.jpg)

2.4 Oculomotor dysfunction (OMD)

The movements of how our eyes move to look around consists of 6 muscles per eye. There is one muscle for each direction to look (right, left, up and down), as well as two muscles that rotate the eye (with and against the clock). Figure 2.2 illustrates the eye muscles and their names. These muscles must work together to be able to focus on the same spot with each eye. The Advanced Vision Therapy Center defines oculomotor dysfunction as: *"Oculomotor Dysfunction occurs when there is the absence or defect of controlled, voluntary, and purposeful eye movement."* (Johnson, 2016). If someone has OMD, it is hard for the person to concentrate and focus on visual tasks for longer periods of time. OMD can also be described as having to see the same object overlapping with itself. For example, one might see the number 800 but sees this number as 8000 instead because of shadows and overlap. This causes major learning disabilities for children, and it is also hard to catch especially in Norway since Norway does not have a very robust screening program for testing vision impairments.

Vision screening can be used to identify OMD. Vision screening can be divided into two different types of screening: normal vision screening and functional vision screening. A normal vision screening or an eye examination is, defined by MedlinePlus as: "*A vision screening, also called an eye test, is a brief exam that looks for potential vision problems and eye disorders.*" (MedlinePlus, 2020). Functional vision screening focuses on the movement of the eyes and how they work together. The group focuses on training of functional eye movements.

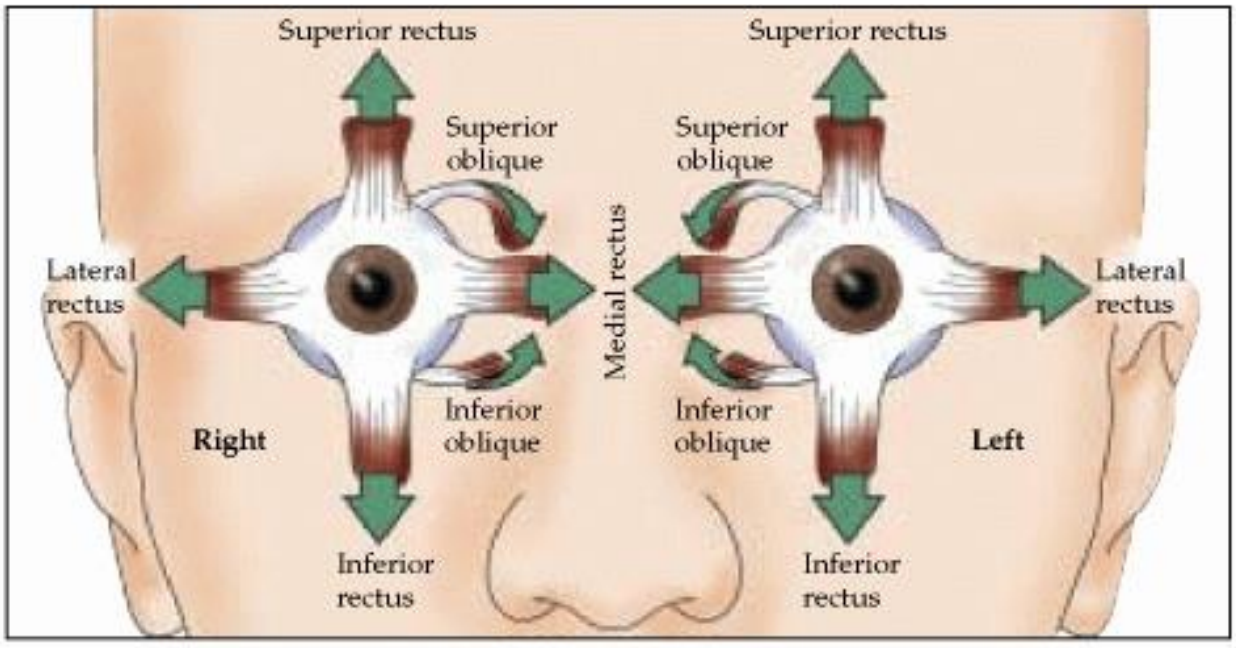


Figure 2.2 – The six extraocular muscles illustrated (Available at: http://i680.photobucket.com/albums/vv166/trimurtulu/eyemovements1.gif)

2.5 Existing products, tools, and games

2.5.1 VisionBuilder

VisionBuilder was created by Haraldseth Software in Norway, however it is more aimed at businesses internationally, as the webpage is entirely in English. It is a web based Optometric vision therapy program aimed at training several different types of eye movement like focus, alignment, depth and more. This program seems to have many different activities going by what it lists on the web page. However, the tool does not seem to record, measure, and analyse concrete data with for example an Eye Tracker. There exists a version for office clinical use and a home version for patients to train and use at home. (Haraldseth Software, 2021)

2.5.2 Lexplore

Lexplore was created in Sweden and has multiple offices and branches around the globe, and the website and tool have several languages. It is a tool for measuring the reading ability in school aged children. Lexplore uses Eye Tracking combined with AI and machine learning to measure accurate and concrete data about the reading ability of the children and if they need more assistance with their reading training. This tool does not seem to focus on other areas in vision problems like OMD, however. Figure 2.3 shows the AI analysis of a test, and figure 2.4 shows an overview screen of Lexplore in Swedish. (Lexplore, 2021)

Figure 2.3 – Lexplore analysis (Retrieved from: https://www.youtube.com/watch?v=1Z14v9N--3E)

Figure 2.4 – Lexplore tool (Retrieved from: https://www.youtube.com/watch?v=1Z14v9N--3E)

2.5.3 RightEye

RightEye is a company created in USA and is primarily English based. They have several different products for training eyes in different ways, and they have modules and products for training and improving eye movement, but they also have modules for reading and sports usage. RightEye claims that focus, concentration, balance and more will be improved after using the product. RightEye have their own Eye Tracking technology which consists of an Eye Tracker and computer combined into one device, but also has an at home component for training without Eye Tracking. It is unsure how the training is done, and if there is usage of serious games, or something else. The Eye Tracker device can be seen in figure 2.5. (RightEye, 2021)

Figure 2.5 – RightEye Eye Tracking (Available at: https://righteye.com/wp-content/uploads/2020/11/Website-VisionTracker-Gif-V4.gif)

2.5.4 AmblyoPlay

AmblyoPlay is created by Smart Optometry in Slovenia but has an English version and all websites are in English. It does not use Eye Tracking, and therefore no saving and measuring concrete data. The product can train problems like lazy eye, strabismus and more, and are trained by playing games and doing other gamified exercises. Further, the games used are similar to existing mobile games going by pictures on their website, and AmblyoPlay has software versions for Android, iPad, Windows, and Mac OS. Figure 2.6 shows a mock-up of the games. (Smart Optometry, 2020)

Figure 2.6 – AmblyoPlay mock-up (Available at: https://www.amblyoplay.com/wp-content/uploads/2020/09/amblyoplay_2x_mockup.png)

2.5.5 Vivid Vision

Vivid Vision is a company in USA and the product is English based. Here Virtual Reality is used, but not with any kind of Eye Tracking looking at compatible virtual reality headsets, but it might be possible, so there is likely no measurement of eye data, and Vivid Vision uses Virtual Reality games for the training. The product is used for training and treating lazy eye and not general eye movement problems. Figure 2.7 is an image from a video explaining how Vivid Vision works. (Vivid Vision, 2021)

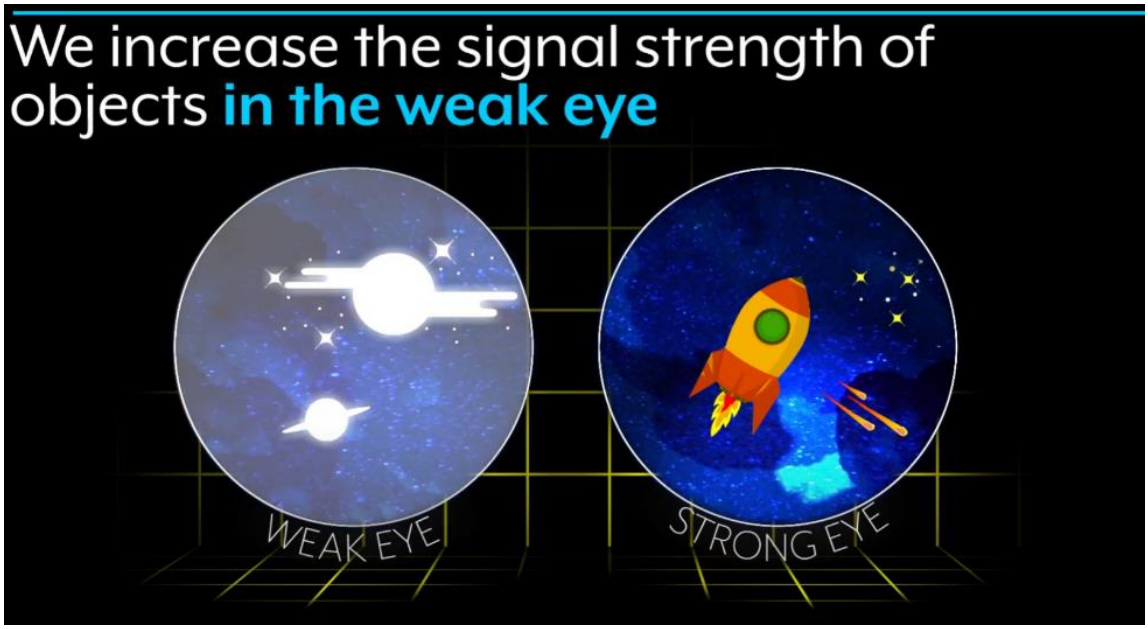


Figure 2.7 – How Vivid Vision works (Retrieved from: https://www.youtube.com/watch?v=5Sr42ZdInfE&t=24s)

2.5.6 COGPACK

COGPACK is made by marker software which is a company based in Germany but has English versions of the software and website. It has training components for many different items, like a program for visiomotor skills, but also other problems not related to vision. COGPACK does not use Eye Tracking and has both a clinical version and a home version for patients to use. Figure 2.8 shows screenshots of two tests in COGPACK. (marker software, n.d.)

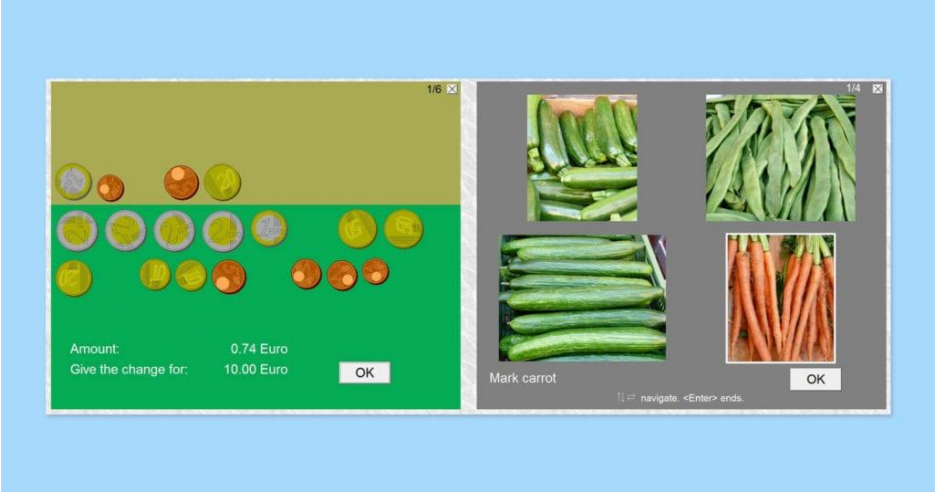


Figure 2.8 – COGPACK image (Available at: https://onemindpsyberguide.org/wp-content/uploads/2020/05/CogPackScreenShots-1024x540.jpg)

3 PROJECT DESIGN

3.1 Possible approaches

There are several constraints the group need to consider for further developing the program, e.g., earlier software and technologies the group also need to apply. The program is, from earlier, developed in Unity which makes it natural to continue with the same development tools. One of the goals of this work is to expand the program to save various data to a database. In this case the database SQLite were chosen for its simplicity and ease of use.

Other possible approaches were not an option for this project as that would take too long, and the games are already developed, and it will be easier to continue developing and expanding them rather than doing everything from scratch. All group members have done a course in graphical computing, and two of the members have also had a course in .NET/C#, but there is little prior experience with Unity.

3.2 Specification

The project is developed using Windows 10 as an operating system since the Tobii Eye Trackers has good support for Windows 7, 8.1 and 10 and is what the group is most familiar with and will be used both for developing and running the program. The programming language used for developing Unity scripts is C#, where Visual Studio 2019 is the IDE, while Unity is the game engine and editor. For development of the user interface Unity's UI Toolkit package is being used.

3.3 Selection of tools and programming languages

3.3.1 Engine / Editor – Unity

Since the games were developed in Unity, the group will use that for running, debugging, and building the program. The group will upgrade the Unity version used to the newest long term support version, 2020.3, so the group can use newer features and ensure that the version chosen will be supported with updates longer than a normal version. The group also switched over to the newer .NET Standard 2.0 as the .NET API Compatibility Level in Unity for compatibility with needed libraries.

3.3.2 User Interface – Unity UI Toolkit

The newer Unity version allows the group to use UI Toolkit, a new method of developing interfaces that builds on XML for writing a page, and uses a language called UXML to add and define visual elements in the interface. For styling UI Toolkit uses USS, like CSS and uses the flexible layout box for layout of the visual elements. UI Toolkit is still a preview version for now but is much better suited for this project's requirements regarding the user interfaces that need to be created rather than using the older Unity user interface systems. It is also the system that will be recommended for new UI development projects in Unity going forward (Unity, 2021).

3.3.3 IDE – Visual Studio 2019

The group has experience using Visual Studio 2019 (VS19) as the integrated development environment (IDE) and this also has good support with Unity and is one of the standard tools for writing C# scripts for Unity games.

3.3.4 Version Control – GitHub

GitHub is being used for version control and to share the project files between the group members more easily. The project is hosted on GitHub and will remain private since the program is not going to be open source at this point. GitHub was chosen instead of Unity Collaborate because the group is more familiar with GitHub, and VS19 has excellent support for GitHub.

3.3.5 Libraries and API

3.3.5.1 Entity Framework Core with SQLite

For access to database Entity Framework Core (EF Core) version 3 from Microsoft will be used since this supports .NET Standard 2.0 (Microsoft, 2021). SQLite will be used as the database provider in EF Core as it stores the database on a local file instead of using a database server, since this will ease install and use of the program. EF Core SQLite comes with several dependencies of other libraries that the report will not list. EF Core will be used to ease saving and retrieving data from the database since there is no need to write SQL queries with EF Core.

3.3.5.2 Tobii Pro SDK

The project uses the .NET version of the Tobii Pro SDK, version 1.8 (Tobii Pro, 2021). There also exits a Unity version but they work the same, and since the existing games uses the .NET version the group will continue to use that but will update it to the newest version. With the Tobii Pro SDK one can retrieve more information than the other APIs from Tobii, like the position for both eyes.

3.3.5.3 XCharts

For easy display of graphs, the project uses XCharts version 2.1.1 from GitHub user "monitor1394" and others (monitor1394 et al., 2021). The group did not want to spend additional time on creating a graph library, so this was found and used. This tool is made for use in Unity and was also open source and free (there are several paid alternatives on the Unity Asset Store). It is very easy to add data to the graphs and to configure the graphs, with several examples to help use the tool. There are some issues where the documentation is not fully translated from Chinese to English.

3.4 Project development method

3.4.1 Development method

The group has daily meetings to discuss and work together, but work has been also done outside of the meetings. Iterative development comes into play when the group had to develop the database part to save eye data so that it could be used for replay. SCRUM has been used so the group members could have more freedom in choosing which parts of the work they want to do work with.

3.4.2 Project Plan

From the attached GANTT form in Appendix 10.2 it is shown that the group works continuously writing blog posts and the project report throughout the whole project period. The group puts off substantial amount of time prior to each assignment to make sure each assignment gets ready in time. MS Planner has been used in conjunction with the GANTT form for dividing the

large tasks/goals into smaller and easier to complete tasks. The group has had an emphasis on designing the implementation first, because otherwise there would be different interpretations and that's not efficient use of time. Developing is the next stage, this is where the group uses the design and transformed it into code. At last, testing the implementation for bugs, if it works as intended, checking for more functionality which was not thought about when designing was investigated.

3.4.3 Risk management

There will always be risk involved with any type of project, and it is important for the team to make a risk analysis. A risk analysis is an analysis where risks, which can have impact on the project, will be identified and given two scores. The scores are likeliness of the risk happening (score from 1-5) and how severe the consequence of the risk is if it happens (score from 1-5). The severity of a given risk is measured by a Risk Factor, which is the product of the likeliness and consequence. Scores between 1-8 is not a big risk, and also has a small consequence if the risk strikes. 9-17 is medium risk, which will not be catastrophic, but should be mitigated. The score from 17-25 is high risk, and should be mitigated at all costs, if possible, since these risks has big consequences for the project. Check the attached risk analysis in Appendix 10.1 for what risks the group has identified and how to combat them. In addition to giving the scores, there has also been written a short text about what should be done if the risk were to happen, and also what should be done to mitigate the risks.

3.5 Evaluation method

3.5.1 User evaluations

To develop technologies that can be used to improve health is a long-time process. The product here is a good prototype, aimed to be recognized as a potential complement for training programs by vision teachers. Therefore, the OMD patients were not in the focus for testing the product. Additionally, it will be very limited to test the product on patients with OMD because of Covid-19 and the restrictions caused by it. The group will however be able to test the product on people to investigate user-experiences and functionalities behind the basic ideas of the product. The group planned a test for the product with a vision teacher. She has extensive experiences on screening and helping children and adults with their functional vision problems. Her feedback would be more about the information displayed in the interface, what is important to show and maybe add some additional information on required functions or usability aspects. A good user experience is very important for the vision teachers as well, since they need to know which training they suggest to the "patients". The group will also test the product on one person with a form of OMD vision impairment to see basic information about how the training can work. The group will also focus on showing the progression.

3.5.2 Program evaluations

Testing the program will not have as many limits as testing the program on possible users, as it does not require coordination and booking of users. It is important for the group to use live debugging and test the program in different unexpected situations that could arise and handle them accordingly.

4 DETAILED DESIGN

4.1 Architecture

4.1.1 Unity

The previous project used the scene structure in Unity to design the program, so the group continued in the same way. The scene structure in Unity works so that each scene is its own entity or something similar to a small program in a way. This makes it easy to separate code, game objects and other items from game to game. The game controller scripts (C# code), and menus also do not need to know anything about the other games and menus other than that there is a scene that it might transition to. The new user interfaces also use this method as well. In figure 4.1 one can see the path between different Unity scenes and how to get to and from a specific scene in the finished program, but loading screens and replay is not shown for simplicity.

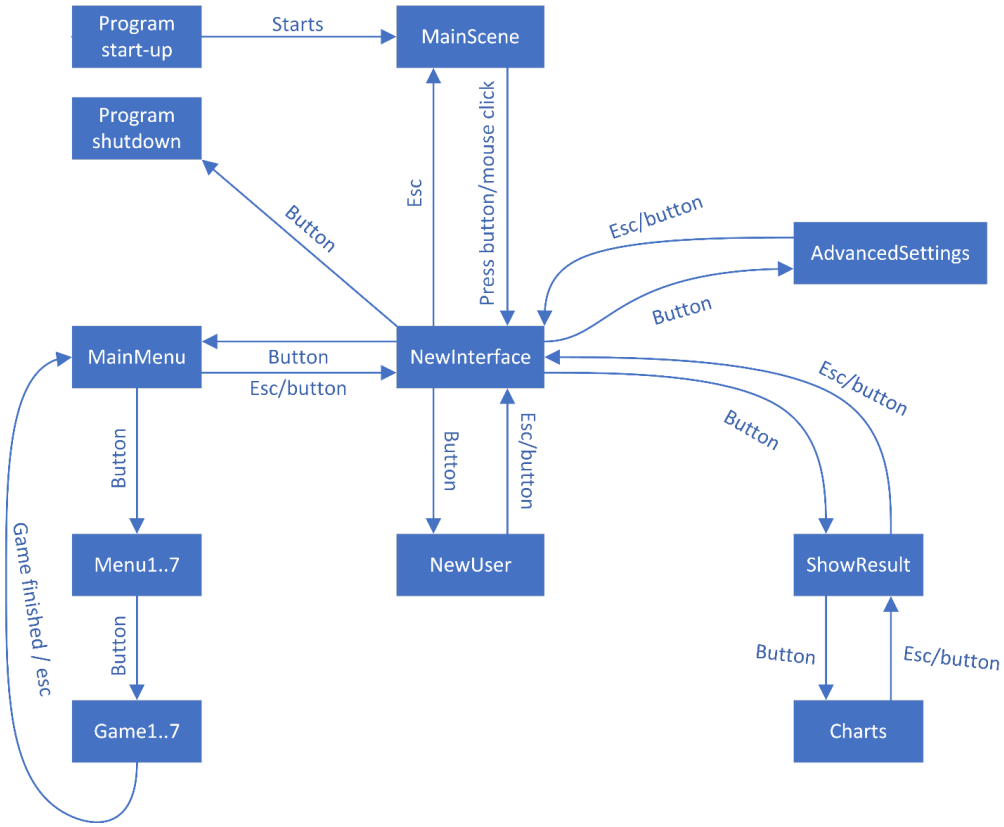


Figure 4.1 – The Unity scene structure.

For sharing data between the different scenes, like setting a game setting for the difficulty before playing the game, several methods exist. The Unity class PlayerPrefs can be used to set values that are persistent, these are saved in registry in Windows, and get or delete them in a different scene if needed (Unity, 2021). PlayerPrefs is a key value pair that is saved. Since these are persistent the game will remember the set settings for each game. Static classes with static properties are also used in the program for some things that does not need to be persistent. A database as detailed in chapter 4.2 is used for larger data like eye data and replay data.

In the program three types of "classes" are used. The ones that inherit MonoBehavior from Unity, and this is typically classes used for game and player controller scripts since these are connected to a Unity GameObject. Other utility classes are ScriptableInstance that does not need to connect to a GameObject but otherwise works the same as MonoBehavior. The rest are normal C# classes or uses some other inheritance like from EF Core.

4.1.2 How the system components are connected

The entire program is a Unity program, and everything talks through the Unity engine and its system. Since .NET Standard is used as the API Compability Level, APIs and libraries that support .NET Standard can be included easily and Unity will include all needed DLLs in the finished build (exe and accompanying data folders). The user interface done with UI Toolkit is a component of Unity that can be optionally installed as a package. So, the program does not communicate with another running program, service, or server for the normal run of the program. The only external system that the program communicates with is the Eye Tracker through the Tobii Pro SDK that will communicate directly with the drivers for the Eye Tracker.

4.2 Database

With the existing program there were no database connected, or any other way of saving data which the group could use to make a replay function, so the group had to integrate a database with the existing program. SQLite, used in conjunction with EF Core, is a free and reliable solution to this problem since it does not communicate with a server and saves the database locally. There is one drawback with EF Core, i.e., when a list of data is inserted into the database at the same time there is no guarantee of order, so the code must order the data after insertion, and one needs a column to sort by. The database gets big quickly due to lots of eye data being recorded, and this makes it necessary to perform performance testing and tuning to make sure the database is optimized for performance. Another downside is slow performance on saving large amounts of data to an SQLite database.

4.2.1 Models

There have been created model classes for everything needed for saving data to the database. The EyeData model represents eye data collected from the Eye Tracker for use in the replay function and analysing. GameData, 1 through 7, are model classes made to

save scores, settings, and difficulties in the games. ReplayData saves the necessary information for replay. Each game has its own ReplayData class, which inherits from ReplayData, because different games have different data that needs to be saved. The User class defines the user with their forename, surname, username, and age where only username is required. The PlayedGame class keeps information about each session from the game and information like the start and end times and which user that played that session. Figure 4.2 shows an overview of the domain model for the program.

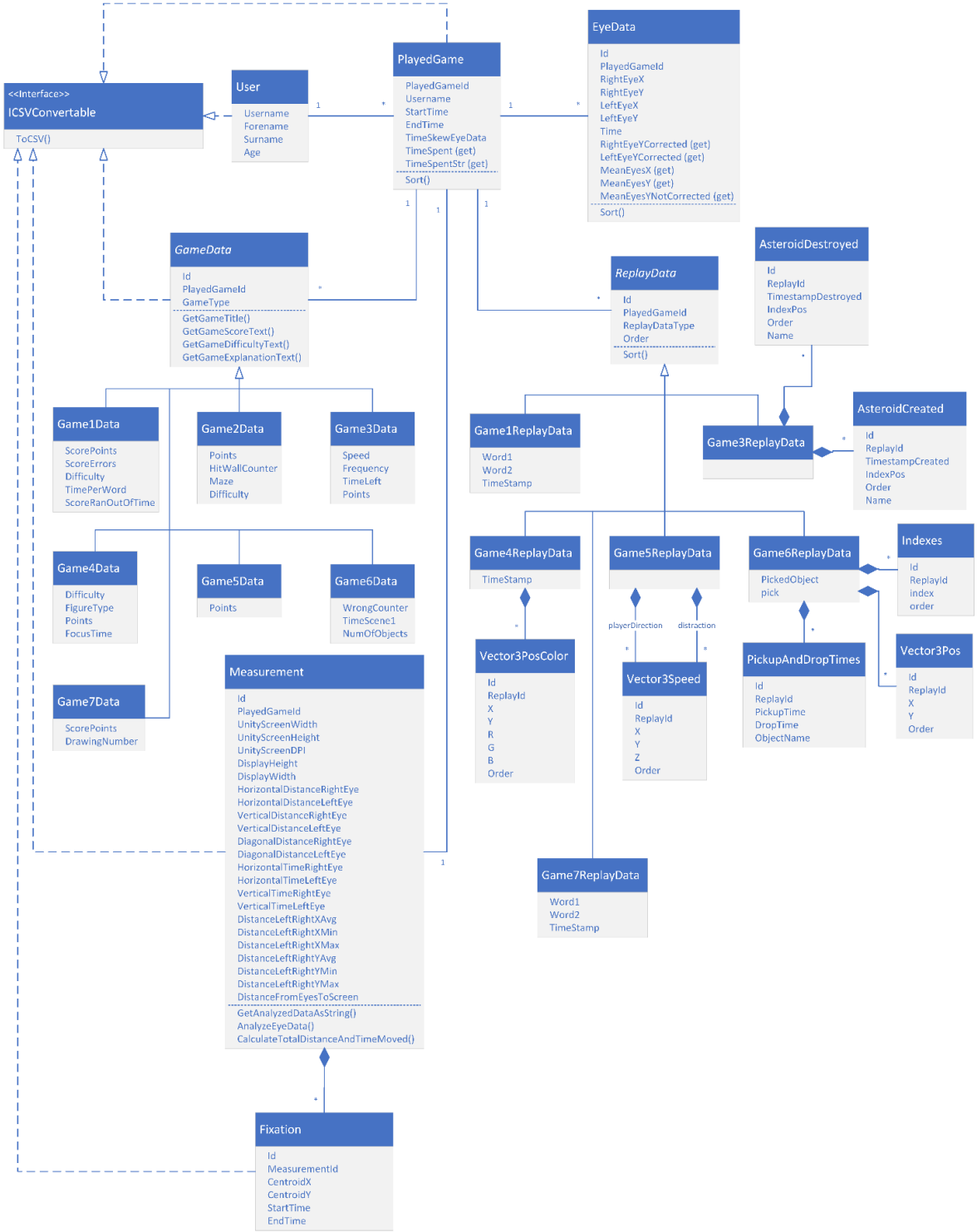


Figure 4.2 – The domain model.

4.2.2 Database Context

GameContext is a database context class which creates the database file called "gazeDatabase.db", and this file is located in Unity's persistent datapath. To create the tables and columns for the models, the group has used the ModelBuilder API with overriding the OnModelCreating method and using EntityTypeConfigurations (Microsoft, 2021). This way of doing it makes it easier to add and/or manipulate tables and columns in the future if there needs to be made any changes. This class is also responsible for establishing relations between tables. The main way to access the database is through an object of this class, and one can use the DbSet sets to add, retrieve, change, and delete objects in the database, where there is one set for the main models, like a set for all the users, eye data, the sessions and more. An object of type GameContext should be short lived and used for only a few operations before disposing of the object, so using a C# using block is a good option for accessing the database (Microsoft, 2021).

4.2.3 Game Saver

GameSaver is a class which saves start and stop time in a PlayedGame object, game information in a GameData object, and replay information in a ReplayData object. Firstly, for the saving to be done there needs to be made a ScriptableObject in every game controller if there should not be a replay. Secondly, the object needs to be initiated which starts the saving. The group encountered some syncing problems when doing a replay, as the eye data did not synchronize properly with the saved in-game events, which made the group make a SavedSkew method which solves this problem by measuring a small delay and adding this in replay. After the game is done, in the method OnDestroy, the game stops saving with the method StopGameSavingAndCommitToDatabase.

4.3 User Interfaces

4.3.1 UI Toolkit

As previously explained Unity's new UI Toolkit system was chosen as the way to make the new user interfaces for the program. UI Toolkit is still in preview so it is missing features and might not be fully stable yet. It uses a file type of UXML for specifying and designing the structure of the UI document. UXML is built on top of normal XML. USS is used for styling the documents as their own files or directly as attributes of items in the UXML document. USS is mostly a subset of CSS focused on the flex layout.

This new system works better for creating regular user interfaces similar to those found in web pages and desktop applications. Especially useful features for the group were easy to apply components like adding lists and binding objects from code to the lists and the event system.

Every UI page is a new scene with a UXML document and a C# code-behind file for that document. On the start of the code-behind object it will query the UI items, set up the document with event handlers, populating lists and more. Most of the new UI pages and scenes uses UI Toolkit, but the welcome screen and graph page does not use it and uses the normal Unity UI system. The game menus and games also does not use UI Toolkit. All the UXML documents are using a set resolution of 1024x576 since it makes designing a page easier and it will scale with the screen size for larger resolutions.

4.3.2 Welcome screen

A welcome screen was added to the program as shown in figure 4.3. It mostly serves as a good place to do start-up functionality, where it will create the database file if it does not exist using EF Core's database EnsureCreated method on a GameContext object. The screen has a dynamic text that updates when the program is ready for any mouse or keyboard input from the user.

Figure 4.3 – The welcome screen page.

4.3.3 Select User

The "select user page" comes after the welcome screen and can be seen in figure 4.4. On load it will retrieve all users from the database and show the usernames in a list, if there are any. There are several buttons on this page, so this acts as a hub to get to the different parts of the program. The "Avslutt" button will quit the Unity program to desktop. The "Avanserte innstillingar" button will move the user to the advanced settings for the program, see chapter 4.3.7. The "Lag ny brukar" button is for creating a new user if there are none or a new user will play games, see chapter 4.3.4. The "Vis resultat" button is for showing all previous sessions and results on the selected user in the list, see chapter 4.3.5. The "Spel" button is for playing the serious games in the program, see chapter 4.5.

A list event is captured in the code when the user selects a username in the list. Then the program will set the username in the PlayerPrefs for the program for use in other scenes. The "Vis resultat" and "Spel" buttons will also be activated so the user can press those. On return to this page from another scene or after quitting and starting the program again, the user that was selected will be automatically selected using the PlayerPrefs. The user needs to select a user so that the program knows on what user to save the play session info from the games.

Figure 4.4 – The select user page. Here a user can add a new user, select a username, and see earlier results or play games. A user has been selected in the figure.

4.3.4 New User

The "new user page" is for creating a new user in the database that can be used to play games as. An example is shown in figure 4.5. Only a username is required, but optionally the user can type in their first and last names and their age. These optional fields are not used for anything other than to store in the database. Checks will be done on the username after stopping to type ("lost focus" of the field), and if the username is something that does not exist in the database the "Opprett brukar" button will be enabled. Age will also be checked if it is a number before creating a user. When pressing the "Opprett brukar" button the program creates a user and saves it to the database and returns to the select user scene. In this scene there is also a small text that informs the user that the program will save the eye data for the user locally on the machine, in compliance of the Tobii Eye Tracking Data Transparency (Tobii, 2020). This might be missing additional info on, for example what the data will be used to and why the data needs to be stored and used to be fully compliant.

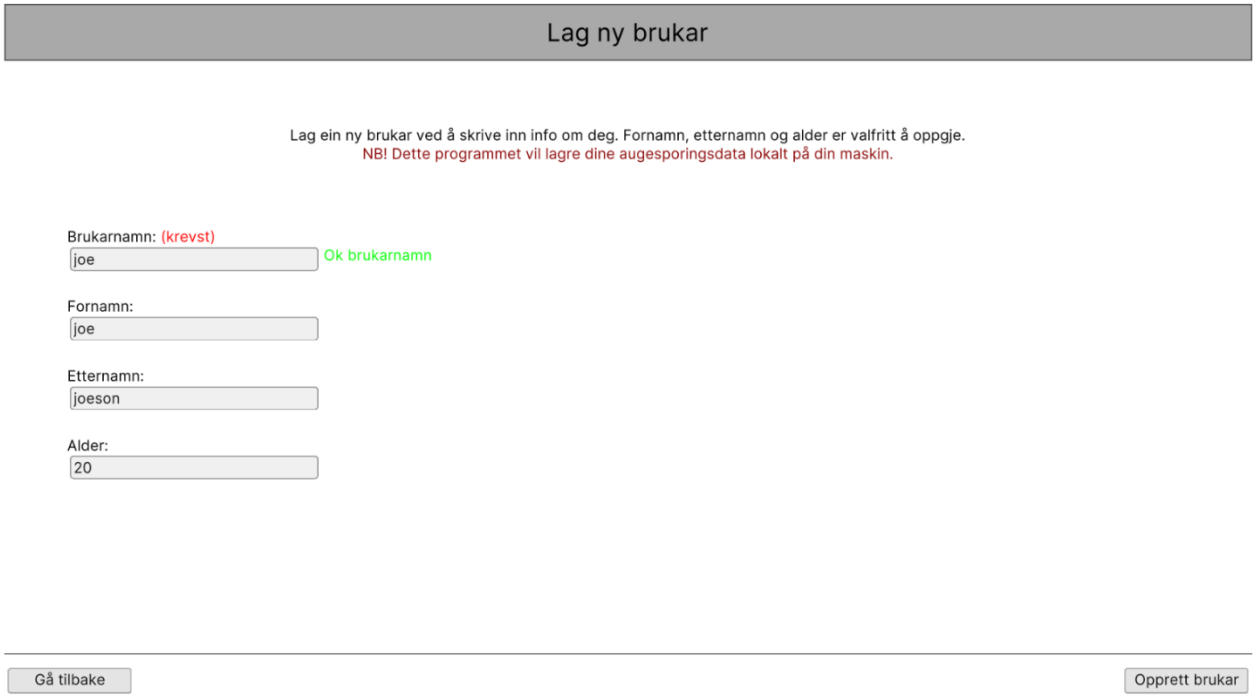


Figure 4.5 – The create new user page with information about a user filled out.

4.3.5 Show Results

The “show results page” will show two lists when navigated to, one for all the different dates a user has played a game, and a second list for the different sessions on that day. When a date is selected in the list to the left, the list to the right is populated with short info about the different sessions that was played that day; what game it was, what the time was at start and total time elapsed.

When a session is selected in the list, the right side of the page will be updated with information about that session which can be seen in figure 4.6 and 4.7. First the general info about the game is displayed; what game it was, the ID in the database, start and end times and total duration of the game. Below that comes information about the different scores, settings and difficulties that was used for that session, and a short explanation of those. Last is some information about what measurements and results from the analysis of the eye data from that session were, like the total distance travelled in horizontal and vertical direction for both eyes, and how many fixations there were. See chapter 4.6.2 for more info on the analysis.

By using a slider on the top of the page the user can choose how much information will be shown at the right side of the page, where it has three options and will first show only scores, then measurement info is added, and lastly more advanced items. The only advanced item as of now is a button which converts and saves most of the information stored under that session as a comma separated values (CSV) file so one can use the information to do analysis and measurements with better tools and algorithms than what was used in the program. The folder and path where the file is saved is also displayed after the file was written to disk, and will be copied to the clipboard on a mouse press of the path text.

On the bottom right of the page the user has two buttons; one button for showing a graph of the eye data for that session, see chapter 4.3.6, and a button for replaying the game, see chapter 4.4.

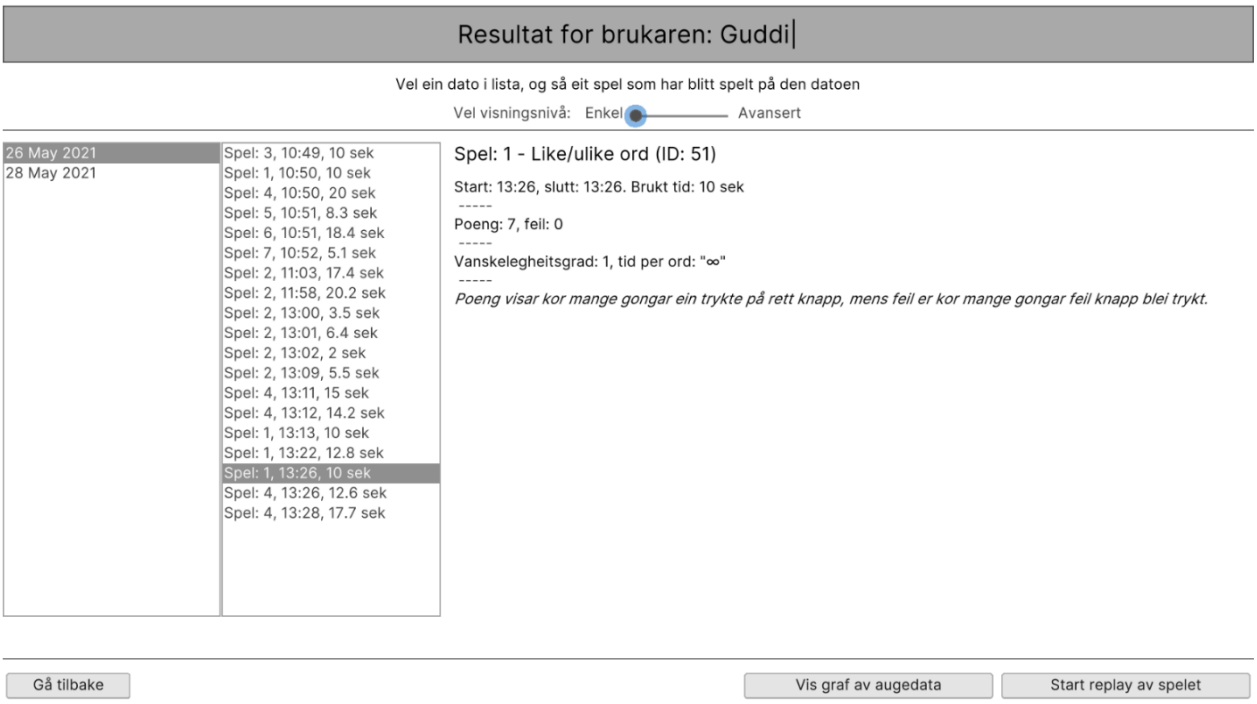


Figure 4.6 – The results page with simple information about the games played and time accomplishments.

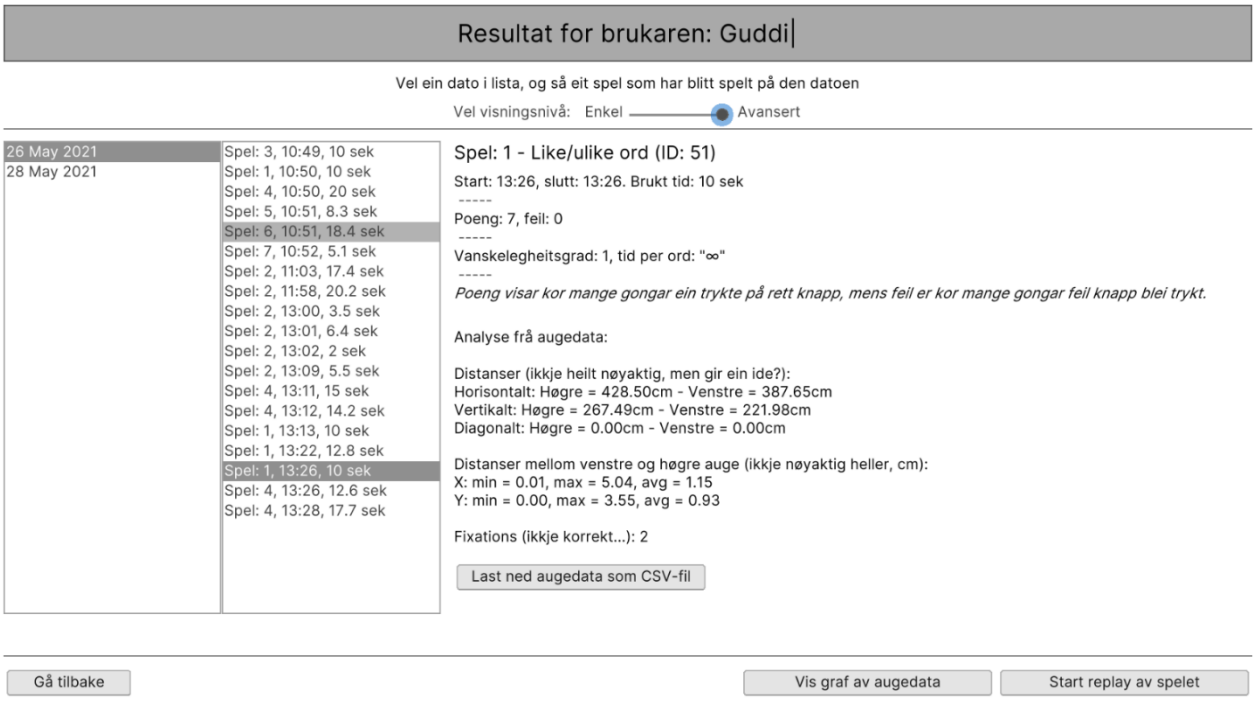


Figure 4.7 – The result page with advanced information about the eye data analysis from a game.

4.3.6 Graphs

The “graph page” uses the XCharts library to show line graphs of the eye data that was collected and stored during the play-session of a game. The line graphs are a component of XCharts that was added to the scene and then configured through a script for that page. There are two graphs, one for showing horizontal movement at the top, and another for showing vertical movement on the bottom. Both graphs show the line for both eyes, with the line marked as what belongs to what eye and can be hidden and shown by pressing the names for the lines. Fixations will be shown as a red transparent rectangle in the graph as well, as can be seen in figure 4.8.

There are several issues that the group discovered with this library that might make it difficult to continue to use it in later iterations of the program. One is that there is a limit of 65000 vertices in a mesh in Unity which this library uses, which limits the data points in the dataset for the graph before it will throw an exception. For datasets larger than this (about 5000 eye data) there is a built-in function called sampling distance where two points closer than the specified distance will be replaced by one single point by averaging the two points. This sampling will lose information, however.

Another issue problem can be that there is no way to zoom in closer on an area of the graph to examine it, which would especially be useful for larger datasets of eye data, so as the graphs are set up now, they do not work especially well for sessions that are longer than one minute.

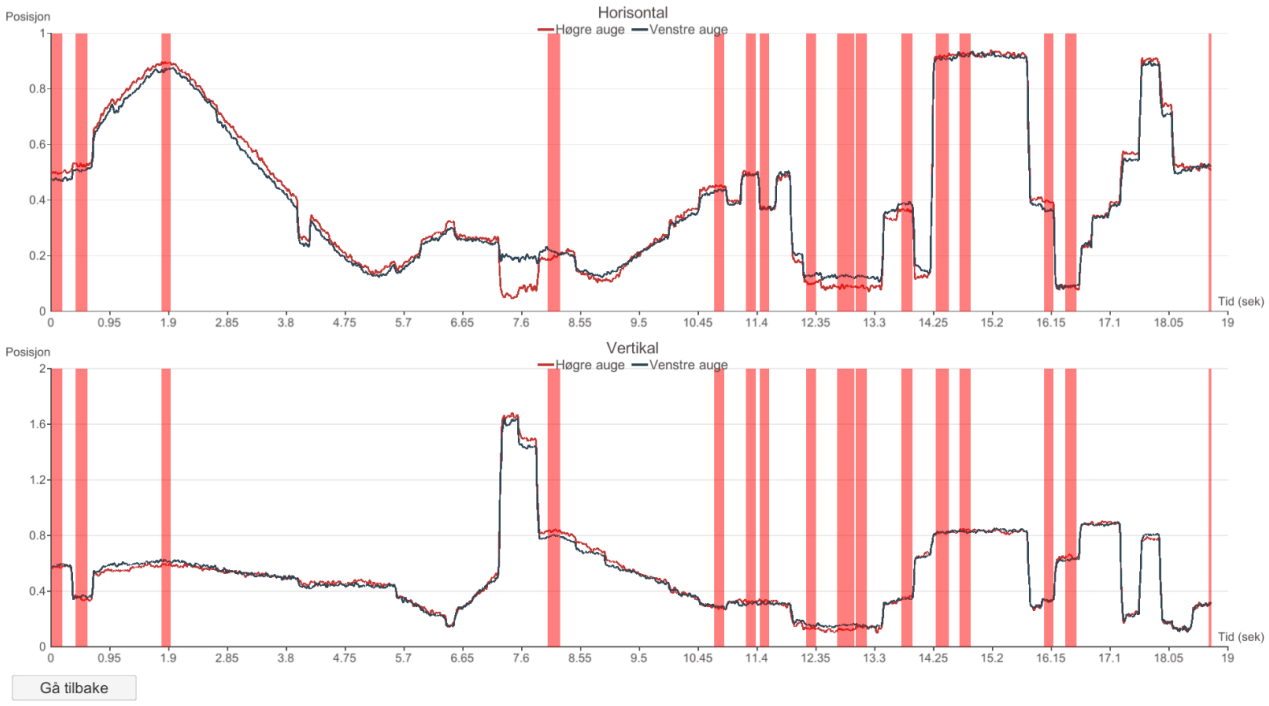


Figure 4.8 – Graphs of a session from game 5.

4.3.7 Advanced settings

The last new user interface page is for doing advanced tasks and is shown in figure 4.9. When a user navigates to this page a warning will be presented and the user must accept the warning to proceed, if it is not accepted the user will be redirected to the select user page again. The warning is there because this page can delete data stored in the database.

The first function of the page is to delete and create a new clean database. Further one can also do a backup of the database file to the same location where the CSV files are saved (this is also displayed as a text in the program). To restore a backup, one needs to rename the backup file as "gazeDatabase.db" and place it in the same folder where the old database file is located. Other functionality is to delete sessions older than some days (can specify how many days), delete a user with all information about their sessions, and lastly one can delete a session with the ID for it.

On the right side of the page there are buttons for testing the performance of the database by doing tests where many thousands eye data are randomly generated, analysed, and saved to the database and measures how long each step takes to do. See chapter 5.2 for the results of performance testing.

Figure 4.9 – The advanced settings page.

4.4 Replay: Superimpose the left and right eye position to see at the same time the visual information on the screen

For doing a replay functionality that would be universal in the program and could be used for any type of game and new games if the necessary information were saved in the database, the group created the class GameReplay. On the creating an object of the class, in the OnEnable method that runs when after the object is created, it will retrieve eye data, game information, and replay data from the database. To make the replay universal, the saved data needed in GameReplay, the game and replay data, are the type of the abstract model classes, which later needs to be cast to the specific type in the games. Since the data is the type of the abstract classes, any game data can be saved and used in the replay function. On creation, all data from the database will be sorted since order is not guaranteed on insert of lists in EF Core, and the eye data will have a short amount of time added (a time “skew”) so the replay will be more in sync with the rest of the stored data for the replay. This skew is needed since the group discovered that there is a delay between the game starts and connects to the Eye Tracker and the Eye Tracker sends data to the Unity program. The delay is measured to be around 20 milliseconds but can vary from session to session.

The solution is not a drop in component, and different code for implementing replay might be needed depending on the game. GameReplay has static methods for checking whether a replay should be played or not, and then different code should be executed depending on if it is a replay or not in the game controller. In the Start method of a game a GameReplay object needs to be created and retrieve necessary data for that game. Saving of eye and game data and connecting to an eye tracker should not be done here or in the OnDestroy method since it is a replay. The Update method will vary the most from game to game; sometime the same logic for the game can be used with the saved eye data, but other times using mostly replay data is necessary. Often a combination is required.

For drawing the eye positions over the game and retrieving eye data for that frame, the method DrawEyePostitions can be used. For finding the eye data that should be shown that frame the group first tried converting the data so that the eye data matches the refresh rate in the game, from 90Hz to 60Hz in the groups case, but this would not work if the frame time would deviate from about 16,67 milliseconds (for 60Hz), like if the program runs faster or slower. The method that was chosen in the end was to use a timer that would update each frame and then step through the list of eye data a few steps each frame to get the eye data right before the point in time the timer is, which is similar to how it works for normal playing. This was also easy computationally to do since there would only be a few steps in the list each frame, max 2 increments for 90Hz eye data on a 60Hz screen. Interpolating the two eye data closest to the timer might also be more accurate but was not done.

Doing a video recording was shortly discussed, but not done because it would take too much space, and one would also need to edit the video afterwards with eye positions added on top of it. A screenshot of game 1 being replayed is shown in figure 4.10.

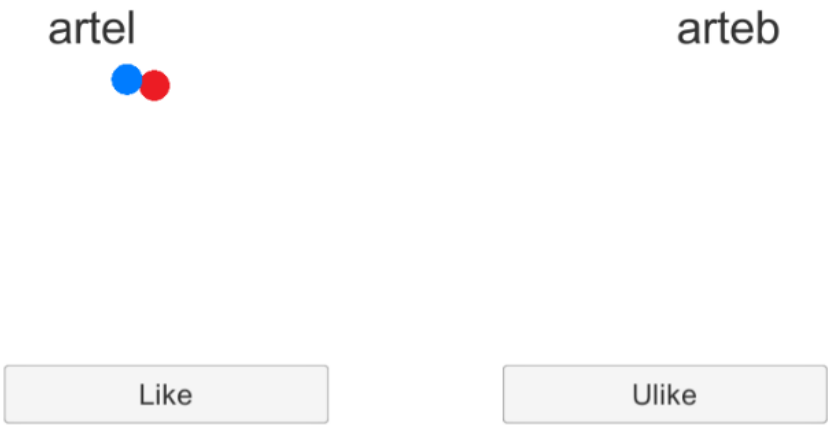


Figure 4.10 – Screenshot from Game 1 being replayed with superimposed eye positions.

4.5 Improving the games

To improve each game, the group chose to add menu scenes for each game and use similar design in all menu scenes. The games that did not have a menu scene were games 2, 5 and 6. Each menu scene had a header with the title of the game, a sub-header with a description of the game and a start button to start the game. Each game also had additional sliders and dropdown menus where a user could tweak the settings of the game. These settings can be the difficulty of the game, how long a game should last, or others varying on the game. Every game has its unique settings as well. For example, in game 6 (Choose the missing object), the player can now choose from 4 up to 10 objects shown in the game. Additionally, the group also added a button to take the user back to the scene from the selected games' menu to make the user experience better and navigation in the program easier. Most of these suggestions were suggested by the group's supervisor, Ilona. The idea behind these additional settings for each game were to create more freedom to tweak and set up each game tailored for each patient in training. People with different vision impairments may require different training possibilities.

The games and improvements:

4.5.1 Game 1

For the first game the group chose to add the functionality for the user to choose how long the game should last. The user can now play the game from 10 seconds up to 2 minutes. The group also added the option to have a time limit on each word comparison. The player can then have

no time limit on deciding if the words are alike or not, but they also can choose to have anything from 1 second to 10 seconds to decide. If the time runs out the words will change.

Figure 4.11 shows game 1 being played with 67 seconds remaining of the game, 2 points gathered, and 6,3 seconds left to determine if the words are alike or not. Figure 4.12 shows the menu for game 1 where the difficulty is chosen as 1 (easiest), no time limit on determining if words are alike and 10 seconds for the duration of the session.

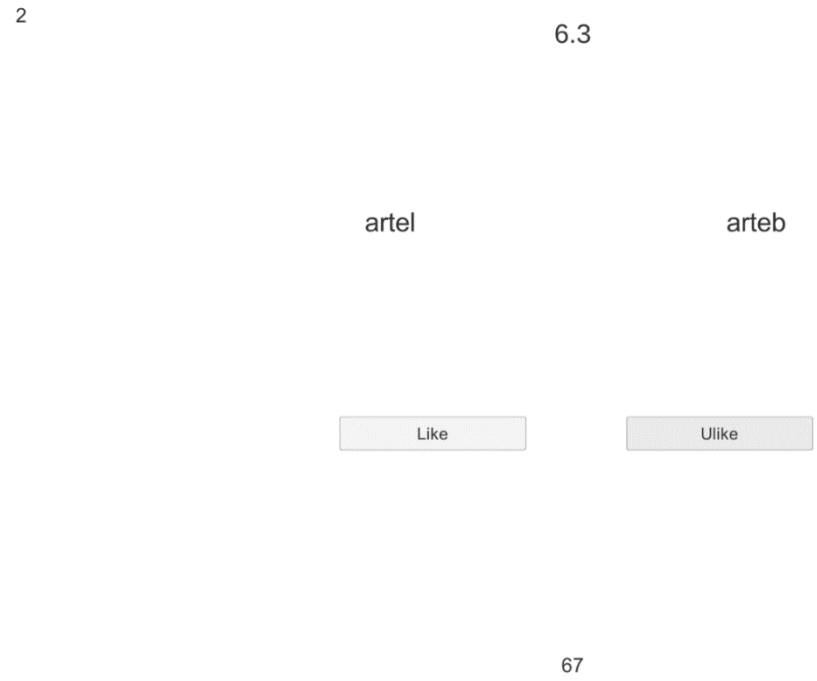


Figure 4.11 – Screenshot from Game 1 being played.

Figure 4.12 – Screenshot from Game 1 menu, where difficulty level, word length, response time can be chosen.

4.5.2 Game 2

For game 2 the group implemented two sliders in the menu. The first slider gives the user the option to choose between 3 different labyrinths to play in the game. The group added new labyrinths by creating an empty game object that contained some sprites with a hitbox and set up which empty game object with the selected labyrinth would be active. In addition to the labyrinths the second slider gives the user the option to choose how many points allowing to win the game. An option for 2, 4 or 6 points was added.

Figure 4.13 shows game 2 being played where the eyes are represented by the red and blue dot. The player has reached 2 points so far. In figure 4.14 one can see the menu for game 2 where labyrinth number 1 has been chosen as well as 2 points being the number of points to be reached to win the game.

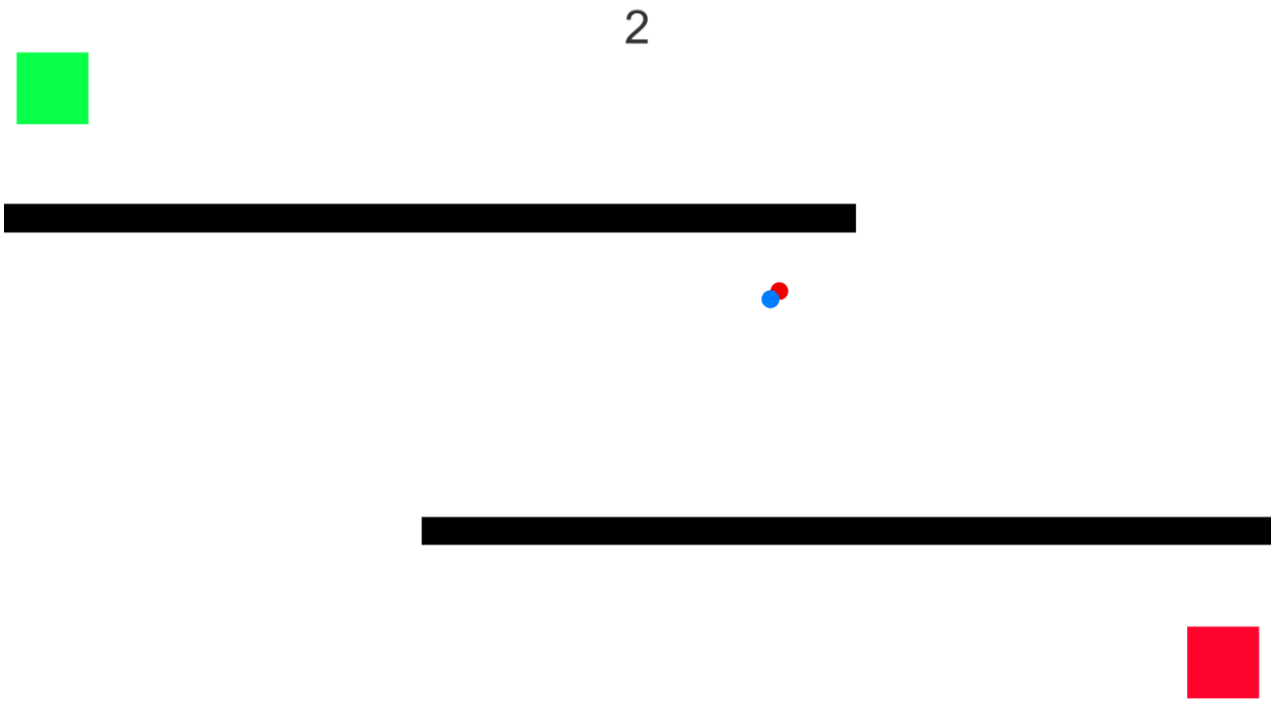


Figure 4.13 – Screenshot from Game 2 being played with superimposed eye position.

Figure 4.14 – Screenshot from the Game 2 menu, allowing to adjust the difficulty (points needed to finish) and which labyrinth to play.

4.5.3 Game 3

Game 3 already had 2 sliders the gives the user the option to determine frequency and speed of the comets going towards the rocket. The group added a third slider in the menu that sets the game to last between 10 to 2 minutes.

Figure 4.15 shows game 3 being played where the player has acquired 4 points and has 27.9 seconds left of the current session. Figure 4.16 shows the menu for game 3 where the speed and frequency of the asteroids are 1 (easiest) and the time of the session to last for 10 seconds.

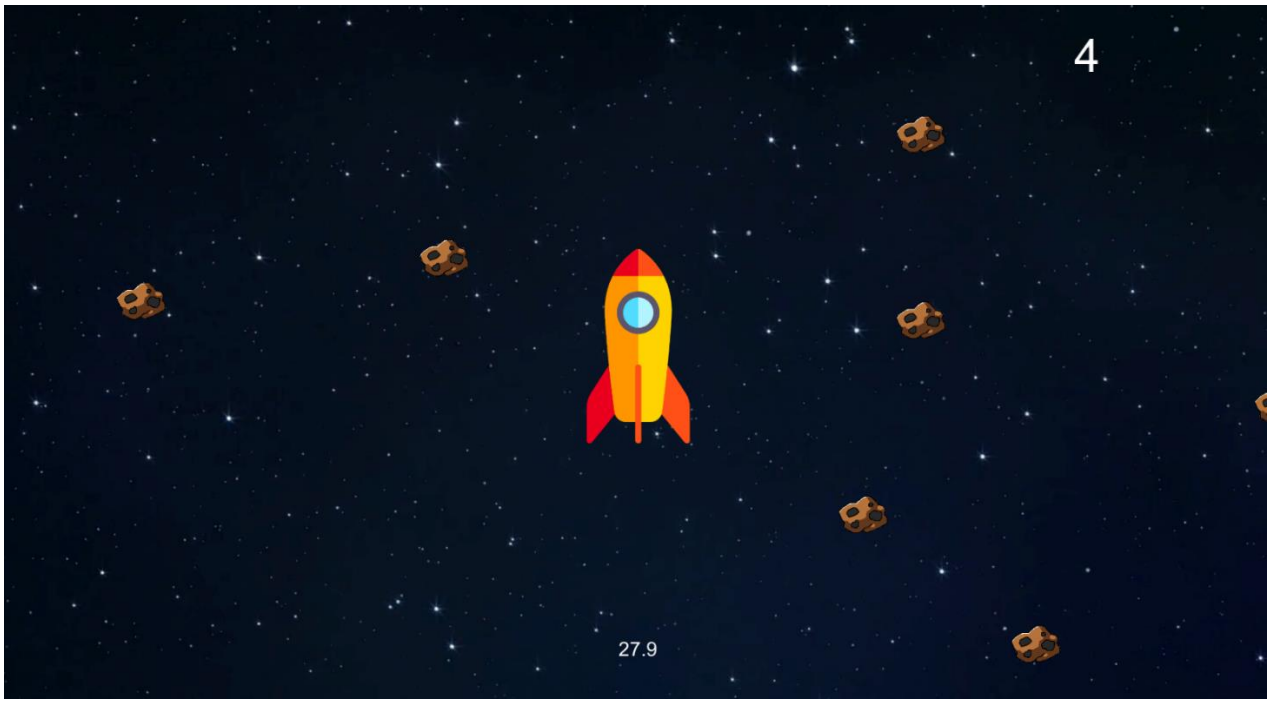


Figure 4.15 – Screenshot from Game 3 being played. The user needs to focus on the meteors to save the rocket.

Figure 4.16 – Screenshot from the Game 3 menu. A user can adjust the speed and frequency of the meteors and time to play.

4.5.4 Game 4

In game 4 the group added a slider to choose how long the focus time should be on the correct figure. The user can now choose to have the player focus on the correct figure from 1 to 10 seconds. Additionally, in the game the group implemented the figure to slowly increase in size as

the figure is being focused on instead of popping up in size for a split second. If the player loses focus, or blinks, the focus timer resets.

Figure 4.17 shows game 4 being played where 2 points have been acquired and the player is focusing on the woman object that is currently increasing in size. In figure 4.18one can see the menu where the player has chosen men and women as the objects, difficulty level 1 (easiest) and the focus time on the target object to be 1 second.

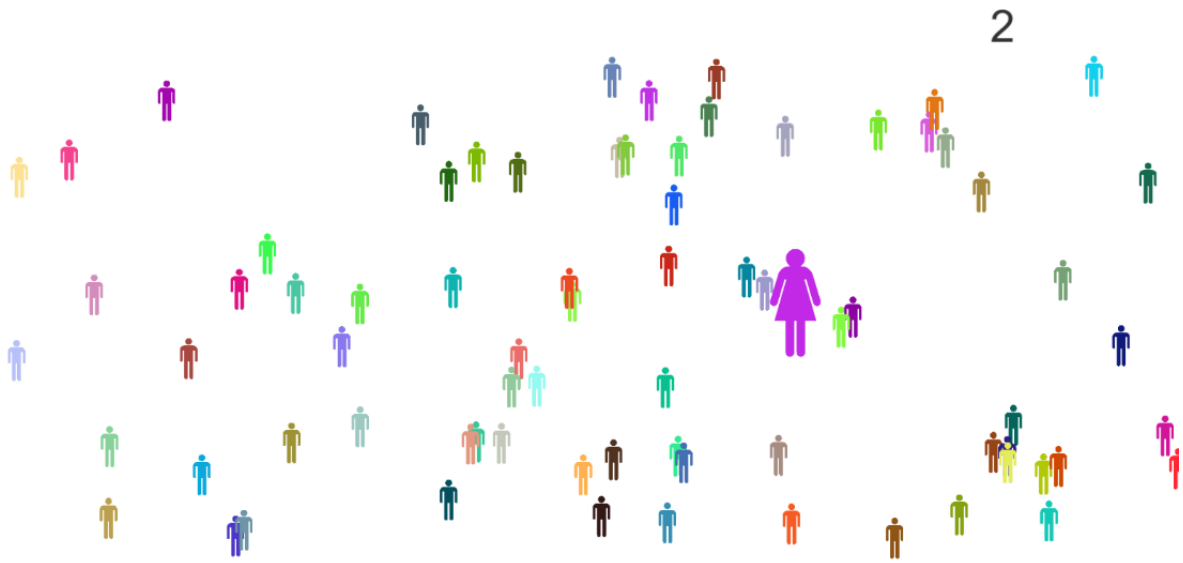


Figure 4.17 – Screenshot from Game 4 being played.

Figure 4.18 – Screenshot from Game 4 menu, illustrating that the user can choose the type of figures, difficulty (the number of figures on the screen) and time to focus to find the target figure.

4.5.5 Game 5

In game 5 the group only added the menu to the game as mentioned earlier. The menu was added without any setting to set up for the game, but this will make it easier to add settings later. With a header and a sub-header explaining the game before starting will help prepare a patient who has not played the game before.

Figure 4.19 shows game 5 being played and that the player has acquired 337 points so far and the distraction has popped up in the left corner. Figure 4.20 shows the menu for game 5 with a description of the game.

Figure 4.19 – Screenshot from Game 5 being played.

Figure 4.20 – Screenshot from Game 5 menu, illustrating instructions to play.

4.5.6 Game 6

For the last existing game, the group added two new settings to the menu. The user can now choose how many objects are shown to the player during the game. The user can choose from having 4 up to 10 objects. If one chose 5 objects, the first picture shows 4 figures clustered in random positions on the screen. In the next picture the player is presented with the same 4 figures plus an additional fifth figure which the player must put in the box/goal to win the game. The second setting gives the user the option to choose how long the player can look at the first picture with the objects clustered. One can choose the time to be between 5 to 60 seconds.

In figure 4.21 one can see 4 objects scattered around on the screen and the player has 28 seconds left to remember these objects for the next screen. In the next figure (4.22) the player has been presented with the same number of objects as well as an additional object (the glass with water). The player has picked this object up and put it in the box and has currently one the game. In the picture of the menu for game 6 in figure 4.23 the player has chosen the time to remember the objects on the first screen to be 5 seconds and the number of objects to be 4.

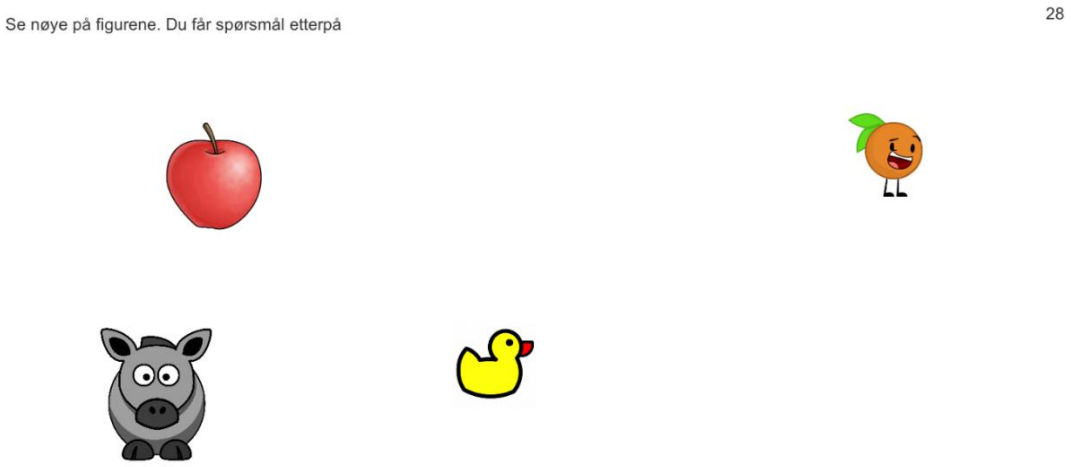


Figure 4.21 – Screenshot from Game 6 being played for screen 1.

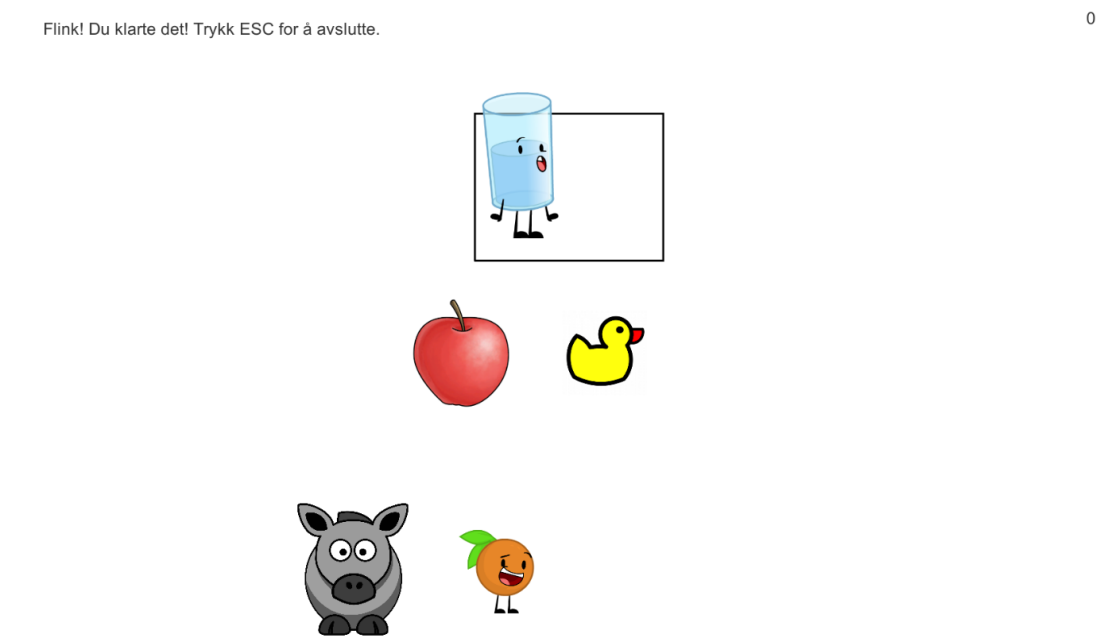


Figure 4.22 – Screenshot from Game 6 being played for screen 2. The correct object has been chosen and the game is finished.

Figure 4.23 – Screenshot from Game 6 menu showing the ability to choose the time to remember objects on the first screen and number of objects to be shown.

4.5.7 Game 7 (new game)

A new game was created by the group. The new game has the player focus on points on the screen in the form of numbers. The player must then focus on each number in order and then a red line is drawn between the points. The last line is drawn from the highest number to the

lowest (starting) number. As of now one can choose between two shapes: a hexagon and a simple house.

In figure 4.24 the player has almost finished the game by drawing a simple house. The only thing left to do is focus on point number 9 and the focus back to point number 1. Figure 4.25 shows the menu for game 7 where figure 1 has been chosen (the hexagon shape) to be drawn.

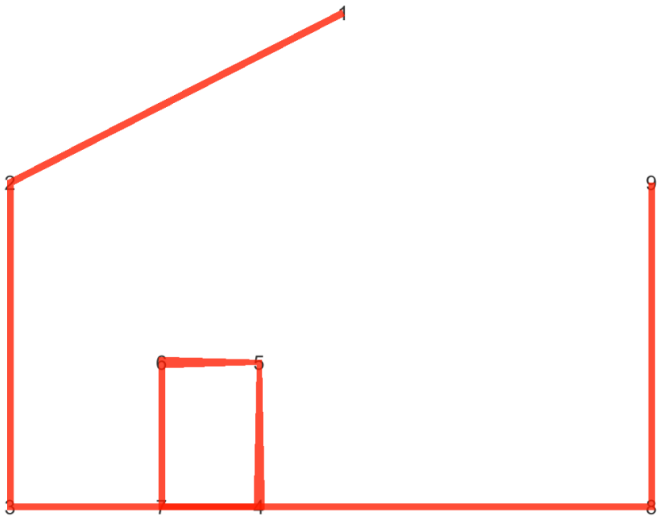


Figure 4.24 – Screenshot from Game 7 being played.

Figure 4.25 – Screenshot from Game 7 menu illustrating that one can chose what figure to draw in the game.

4.6 Other components

4.6.1 Eye tracking

As the group continued working on the already existing software, the group analysed and looked at how Eye Tracking was implemented in the program they were working on. The group found a class in the software called EyeTracking which utilized the Tobii Pro SDK library. With use of the SDK, one could use this library file to communicate with the Eye Tracker and track each eye. The class is used in every game. The way it was used was by creating EyeData objects with data from the SDK and adding them in a list with several of those objects. The EyeData object is part of the database model that the group developed. This way one can save the information gathered. Afterwards when the game ends one ends up with a very long list of eye information throughout the course of a played game session. Afterwards the information is written to the database.

4.6.2 Measurement / analysis of eye data

At the end of the project, measurement and analysis of eye data after playing was implemented. The group does not have proper knowledge about this field so the algorithms are not properly implemented and might be incorrect in relation to thresholds applied for different vision problems. The group implemented two methods in the Measurement model class called AnalyzeEyeData and CalculateTotalDistanceAndTimeMoved for measuring and calculating different data. A FixationAnalyzer class was created for finding fixations along with a Fixation model class.

4.6.2.1 Horizontal, vertical, and diagonal eye movement

For horizontal and vertical eye movement the total distance travelled with both eyes in those directions are measured. This is measured by adding up all the distances between two and two neighbouring eye points. Since the eye points are in Tobii's "Active Display Coordinate System" (ADCS) and each point is from (0,0) to (1,1) the points needed to be converted to pixels or centimetres for a real-world system of distance (Tobii Pro, 2021). Through Unity's Screen class the group got the height and width of the program window in pixels, and the DPI (dots per inch) value and used these values to convert the points to centimetres (Unity, 2021). The DPI value does not seem correct after comparing the value to the specification of the screen, but it is what the group managed to find, and is close to the real value.

Diagonal distance travelled is measured by comparing the vector between two neighbouring eye points to a diagonal vector, like the vector (1,1), and then seeing if the two vectors are close to being parallel by some degrees and then adding that distance. Time measurement for each direction was tried but does not work and needs a better way to do this by excluding "noisy" data when the eye is "standing still" and focusing on the screen. The problem with noisy data will also apply to the distances travelled and a way to exclude this type of data was not found for this project in time.

4.6.2.2 Fixations

For finding the fixations in the eye data recorded while playing, Qasim were consulted, and the group was given several algorithms in other programming languages. An implementation of the Identification by Dispersion-Threshold (IDT) algorithm by GitHub user “ecekt” written in Python was chosen based on its simplicity and the groups familiarity with the language (Takmaz, 2017), and then implemented in C# in the FixationAnalyzer class.

To get the algorithm working, two parameters need to be adjusted: the distance threshold, the “area” where the eye points need to be within to count as a fixation, and the duration threshold, how long to focus before it is counted as a fixation. Duration threshold is now set to 150 ms and can vary between 100-400 ms but no “correct and accurate” value exists. The distance threshold should be the distance on screen that is between 0,5 to 1 degree off from the vector from the eye position to the screen. To calculate this the group retrieved the eye origin position along with the gaze position on the screen in the Tobii User Coordinate System (UCS) where the points are given in millimetres and calculated the distance between the two positions and then converted it back to a normalized coordinate system used by the eye data (Tobii Pro, 2021).

When the algorithms find a fixation, the start time, end time, and the centroid positions in x and y coordinates are saved in the database.

4.6.3 Loading screens

The group noticed that in certain situations the screen would appear as being “stuck” while it was doing some work like saving or doing calculations, and then implemented loading screens in these situations to inform the user that something is happening in the background. A loading screen was added when loading from a game’s menu to the actual game since this could take some time in setting up the scene and loading in game objects. When returning to the game selection menu after finishing a game another loading screen was added, since here the program will analyse the eye data and then store all information recorded during playing which might take some time. Lastly a loading screen when loading a replay of a previous session was added since the program needs to fetch all the data from the database and set up the game scene.

The loading screens are implemented using small scenes with only a camera, some text on a canvas and a short C# controller for the scene. Changing scene to a loading screen is very fast, and after the loading screen is loaded, it will immediately begin to load the actual next scene asynchronously while updating the text with a progress info and other text to make the loading screen seem like it is doing something. For knowing what scene to load next from the loading scene, a static class is used for sharing what scene to load next. A guide from the website Game Dev Beginner was used for creating loading screens (French, 2020).

4.7 Use Case / Scenario

The games' use case would be that it is introduced to a person with some form of OMD by a vision teacher. The vision teacher can help the user with calibrating the Eye Tracker and set up a user account or choose an existing one for the user using the program with the games. Then the user is free to play the games and also check out the replays if they desire to see where they looked during a game. The vision teacher can also analyse the replay with the user and give valuable feedback to the user. The graphs that are displayed for each game played is probably more useful for the vision teacher. A schematic representation about the use case can be seen in Figure 4.26.

An individual can also use these games at home to play the games granted that they have an Eye Tracker that supports the Tobii Pro SDK like for instance the EyeX or the 4C model. The user experience might also depend on the age of the person. A child will more likely figure out how to play the games much faster than how an older person would.

The use case diagram in figure 4.26 shows a potential user and what options the user has, to navigate in the program. The user can be both a normal person and a vision expert.

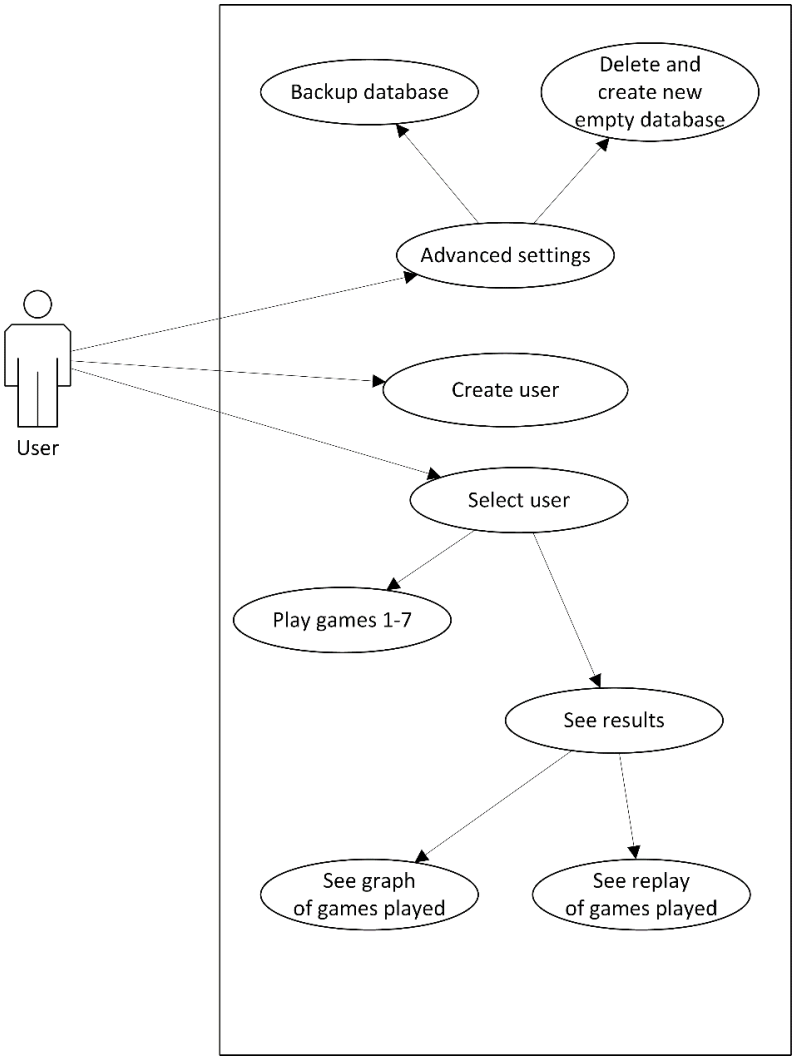


Figure 4.26 – The use case diagram.

4.8 Build

Building the program from Unity to a playable executable file is not hard to do, but the group had to make sure that all files used in the games come with the build. For example, in game 1, one must compare words and the words are fetched from a text file which must be placed in the correct folder for it to be included. Also, the settings must be correct when building the game. For example, the game is meant to be played on a screen with an aspect ratio of 16:9. The game should scale with the screen's resolution, and screens with a different aspect ratio should have bars at the sides or top and bottom. The build is set to play in full screen and in an aspect ratio of 16:9.

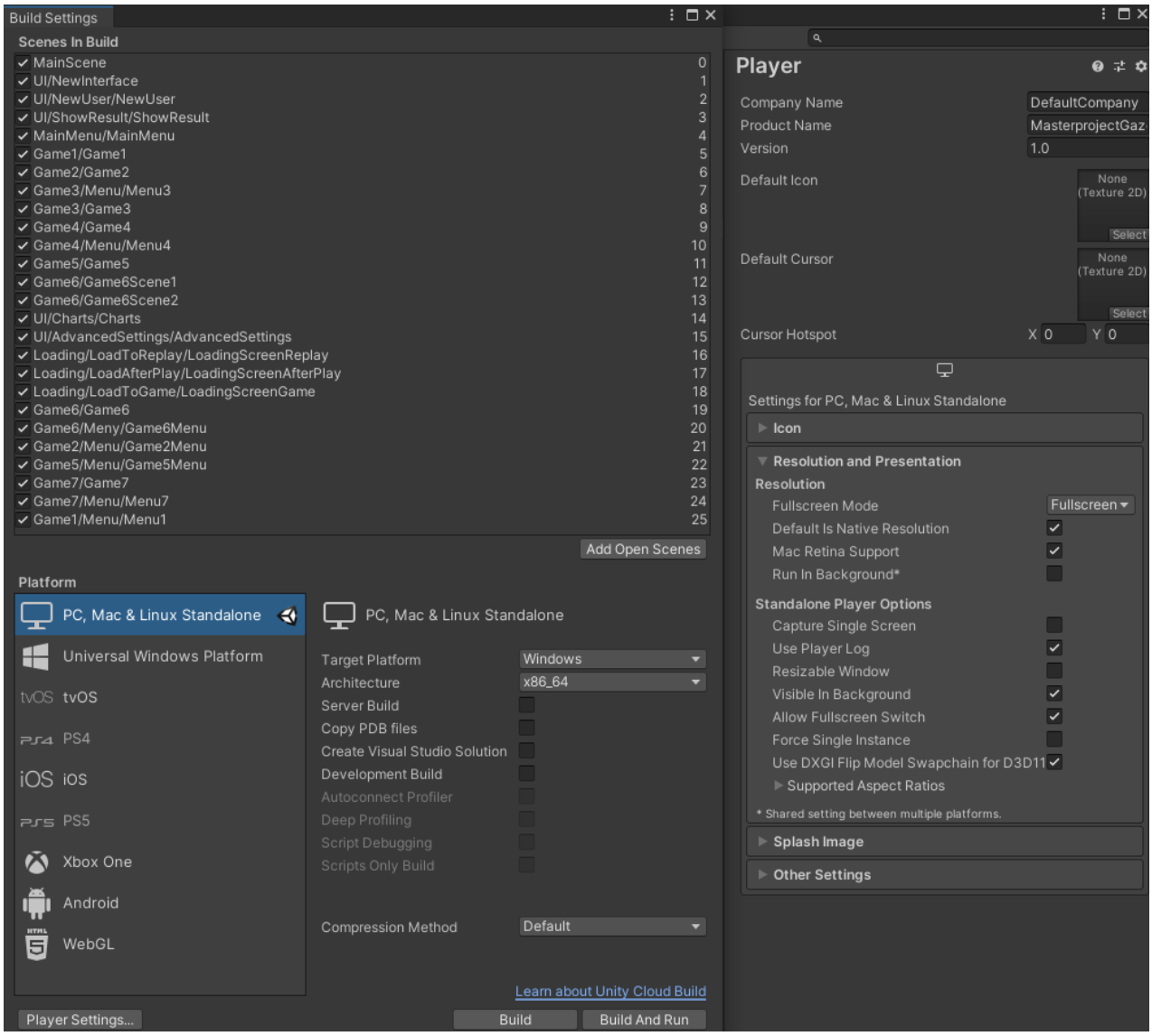


Figure 4.27 – Build settings in Unity.

5 EVALUATIONS AND RESULTS

5.1 Evaluation methods

The feasibility of the application was evaluated by testing the program continuously and with user experience evaluations.

5.1.1 Program evaluations

For testing the functionality of the program and the correctness of the code the group has done several tests throughout the work on the program, to see what needs to be changed or implemented before committing to the GitHub repository. Debug mode in the Unity editor and logging to the Unity debug output were used when testing.

Testing of replay was mostly done with manual tests and seeing if it seems “normal” and “good”, but a video recording of the playing session was also used to check if the replay were fitting the video. The graph page was tested by using several different sizes of data sets of actual and randomly generated eye data to see if any errors would be shown. For testing the user interfaces, the group tried to use different buttons and see how they work, type into inputs field, and load up the pages with different amounts of data, like 0 users, a string in a number input field, and a user with many different days of sessions and more.

As chapter 4.3.7 mentions, menus and buttons for testing performance of saving, analysing, and retrieving eye data were added, and has been used for testing if the choice of database would be enough for the program’s needs. There are three performance tests, one will save, analyse, and retrieve larger and larger eye data sets, in 30 seconds intervals, from 30 seconds to 600 seconds of 90 hertz eye data. The second test will do the same but will do saving to the database asynchronously and will analyse the eye data alongside saving to check if it can give a performance uplift. Finally, a test will use smaller data sets, 30 seconds to 120 seconds in 30 seconds intervals and do this 15 times to check if performance degrades when the database becomes larger. Times are measured using C#’s stopwatch (Microsoft, 2021), and are saved to file as a CSV-file, and the database will be empty when starting the tests. All the tests will be performed three times each. Performance testing will be done on a Surface Book 2 15 inch with a 4 core Intel Core i7 CPU, 16GB RAM, and a 256 GB PCIe SSD set to full performance mode in Windows and connected to a charger, and a desktop PC with a 16 core AMD Ryzen CPU, 32GB RAM, and a 240 GB SATA SSD.

5.1.2 User experience (UX) evaluations

The way the testing was conducted was to have the test person sit down in front of the computer and help them calibrate their eyes for the Tobii Eye Tracker and then let them try the program. After giving information about the application, the group also assisted the users by telling them what to do if they were confused and did not know what to do.

The group started by telling the users to define a new user and then use this user to test all the games. The users were also invited to have a look at the replays and see how they did. Each test person tried each game at least once. The group also asked each test person about their overall experiences, and notes were taken on their feedback.

The user tests were performed 26th of May 2021 and tested 3 different people. Each testing lasted approximatively one hour, except the test with the vision teacher that lasted 2 hours, and were performed in Norwegian.

The feasibility of the last prototype was evaluated by 2 potential end-users and one vision teacher via a qualitative evaluation battery (Heldal et al., 2021). This included information about the study, consent for participation, a background questionnaire, a UX questionnaire (Laugwitz, et al., 2008), observations during the test, I.e., how they played the games and what they struggled with and was followed by a semi-structured interview.

The first test person is a vision teacher working with vision screening and training in her everyday work. She is female in her 50's. She is knowledgeable in the field of helping both children and adults with visual problems as well as OMD. It is a very broad and resource demanding field to screen and train people with these problems. Her feedback can be considered very valuable for this project, both regarding her competencies as a vision teacher, but also knowing the requirements and capabilities of different patient groups.

The second test person is male and is in his 50's. He has progressive glasses that might be good to test to see if there are different results. His user experience can also be very valuable as an individual with normal vision and glasses.

The final test person is male and is also in his 50's. He has a form of OMD where his eyes are not well aligned or synchronised. His experience with playing the games will be very valuable to the group as to see how his vision handles the games and to see what works and what does not.

5.2 Evaluation results

5.2.1 Results from the program evaluations

For testing the program, especially parts of the program that needs an Eye Tracker to work properly, the group had some difficulties to coordinate work since only one of the group members had a compatible Eye Tracker. To test the code that needed an Eye Tracker, is required, and most of the work was done via distributed collaboration and it was difficult to plan some of the period's meetings on the place. The group found a solution of sorts where sending a copy of the database file with several sessions already played would allow for the other members to test components like the replay and graph page.

After doing extensive testing of the program, the group found that the new interfaces are stable and work well and does not throw unexpected issues when faced with different situations. There

are however some design issues for some pages, especially the result page with showing results for games with many settings, scores and difficulties and a lengthy explanation, as that would go out of the designed area of where the text should be located.

Replay works well after testing and comparing video recordings of playing and replay, but there are still some small differences and should need more work. Especially in games that use the original logic in the games for the replay and uses the eye data retrieved from GameReplay to check if the eye position is inside a game object on the screen, where the replay and the actual game might get out of sync if the wrong eye data were used. As mentioned earlier, interpolating the eye data might help, but there should be done some more work to get the synchronization of the eye data for the replay more accurate as well.

The performance results for the Surface Book 2 laptop shows some issues with the database. The first test (small to large data set, see figure 5.1) shows that doing analysis is short (under two seconds up to about 5 minutes of eye data), and retrieving eye data is negligible (a few milliseconds), but the group suspects that this might be a fault in the groups testing, and retrieving data should be tested in isolation to confirm. Saving data however takes some time, and takes around 33 seconds for 5 minutes, with up to 2 minutes for 10 minutes of data, and the scaling is not linear it seems like as the database fills up. For game time under 3 minutes the saving time is acceptable. The second test, see figure 5.2, shows that using async and saving and doing analysis “at the same time” does not give any performance improvement, and in some cases is slower. In the final test (many small data sets multiple times, see figure 5.3) one can see that as the database filles up, it becomes slower to save to the database. 30 seconds of eye data goes from only taking about 1 second to save up to around 11 seconds after 15 repeats. All the graphs use milliseconds for the y-axis.

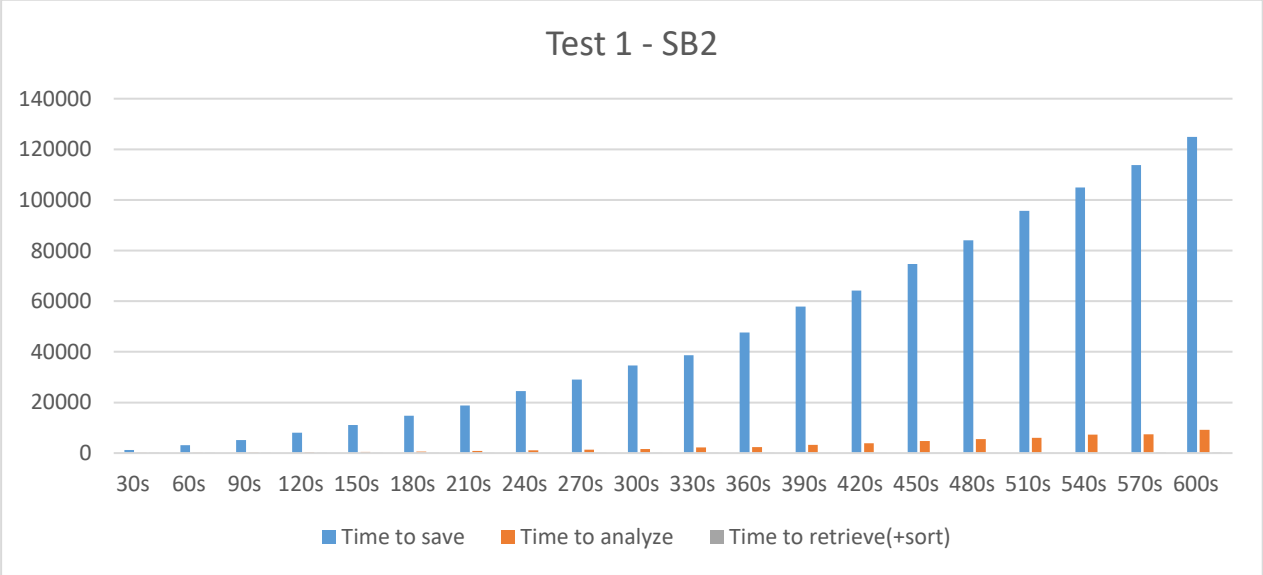


Figure 5.1 – Graph from performance test 1 on the Surface Book 2.

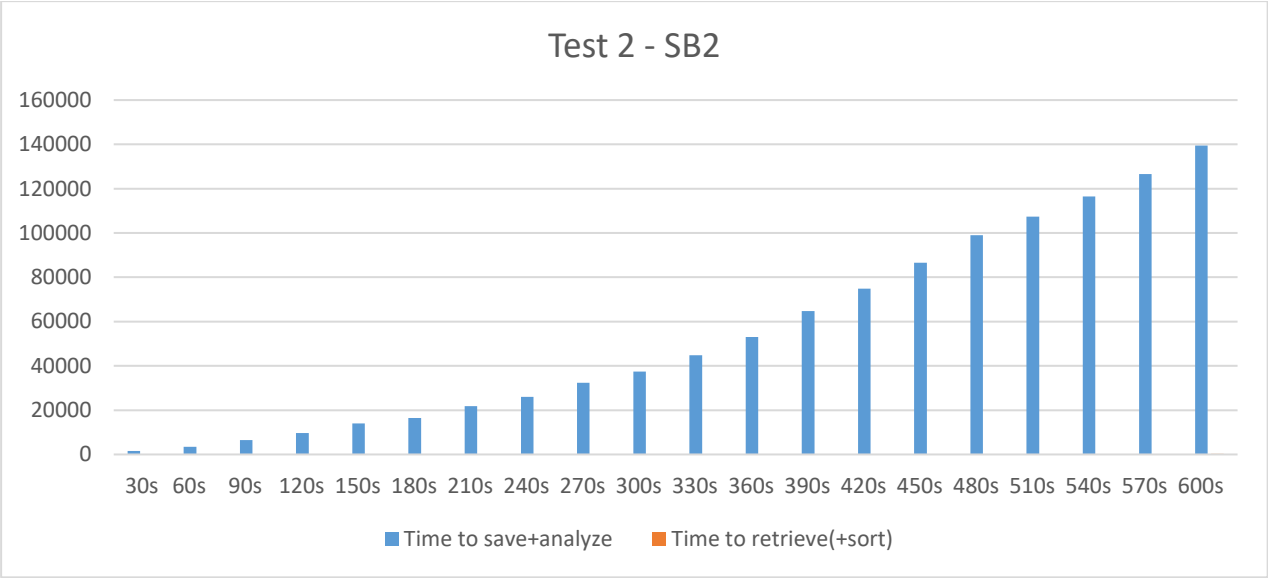


Figure 5.2 – Graph from performance test 2 on the Surface Book 2.

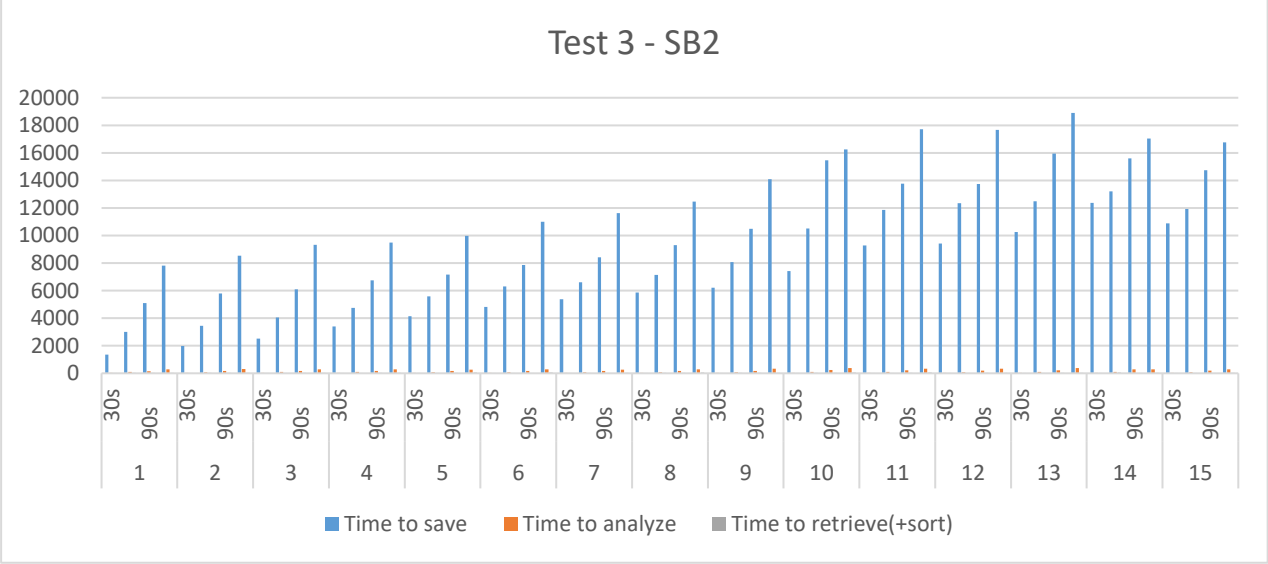


Figure 5.3 – Graph from performance test 3 on the Surface Book 2

The performance on the desktop is on average about 2 to 3 times slower as the Surface Book 2 because of the slower SATA SSD versus the much faster PCIe SSD. This makes it take a lot longer to save, for example 18 seconds for saving 2 minutes, where the Surface Book 2 only took 8 seconds. The time to analyse is also a lot higher, up to 4-6 times as high, for some unknown reason. But otherwise, the desktop performance shows the same as the Surface Book 2. See appendix 10.3 for the full performance numbers and graphs from the desktop PC.

After doing performance testing the group can see that SQLite as the database provider is not an ideal choice, but that it could be a combination with SQLite and EF Core, and changing to another object-relational mapper (ORM) might help performance as well at the cost of increased complexity in the program. The tests also show that a normal SATA SSD might be too slow for the implementation, and that using a magnetic hard drive is probably very slow and would not be a good user experience. This testing should really have been done before development started

properly, so the group could have tested with different database providers to check the performance. Also, the retrieval of eye data might not be accurate and should be done in isolation from saving to rule out the possibility that the program already has the objects in memory.

5.2.2 Results from the user experience (UX) evaluation.

After having a functionally usable product, the user experience (UX) evaluations were performed. Data was collected via questionnaires, observations, and interviews from the three participants. This gave the group a broader picture of how potential users can interact with the program and what might need to be changed to give a better user experience. Overall, the group received positive feedback, appreciating the product. For further development, problems, both with the equipment, design, and execution of the program were defined.

One of the early observations was that calibration of the Eye Tracker have some problems, almost for everyone. The group suspects this could have something to do with the resolution of the computer, which is higher than 1920x1080, but there was done an attempt to downscale to 1920x1080, since it works better at resolutions with an aspect ratio of 16:9.

The first participant, a male, the group had some technical difficulties with the build not working correctly and some calibration issues, but after some time the participant was able to test the games, as seen in figure 5.4. The tester had a good time and found most of the games interesting and fun. There was one question asked by this participant, which was why there needed to be made two profiles/users. This is because for now one must make a user with the Tobii Windows 10 software to calibrate the Eye Tracker with the PC, and then make a user in the program. This participant uses progressive glasses, which the group thought might be a problem, but in the end, it did not have any influence on the Eye Tracker nor his experience. He experienced some problems playing the game 7, but the group suspects this might have been due to the calibration.

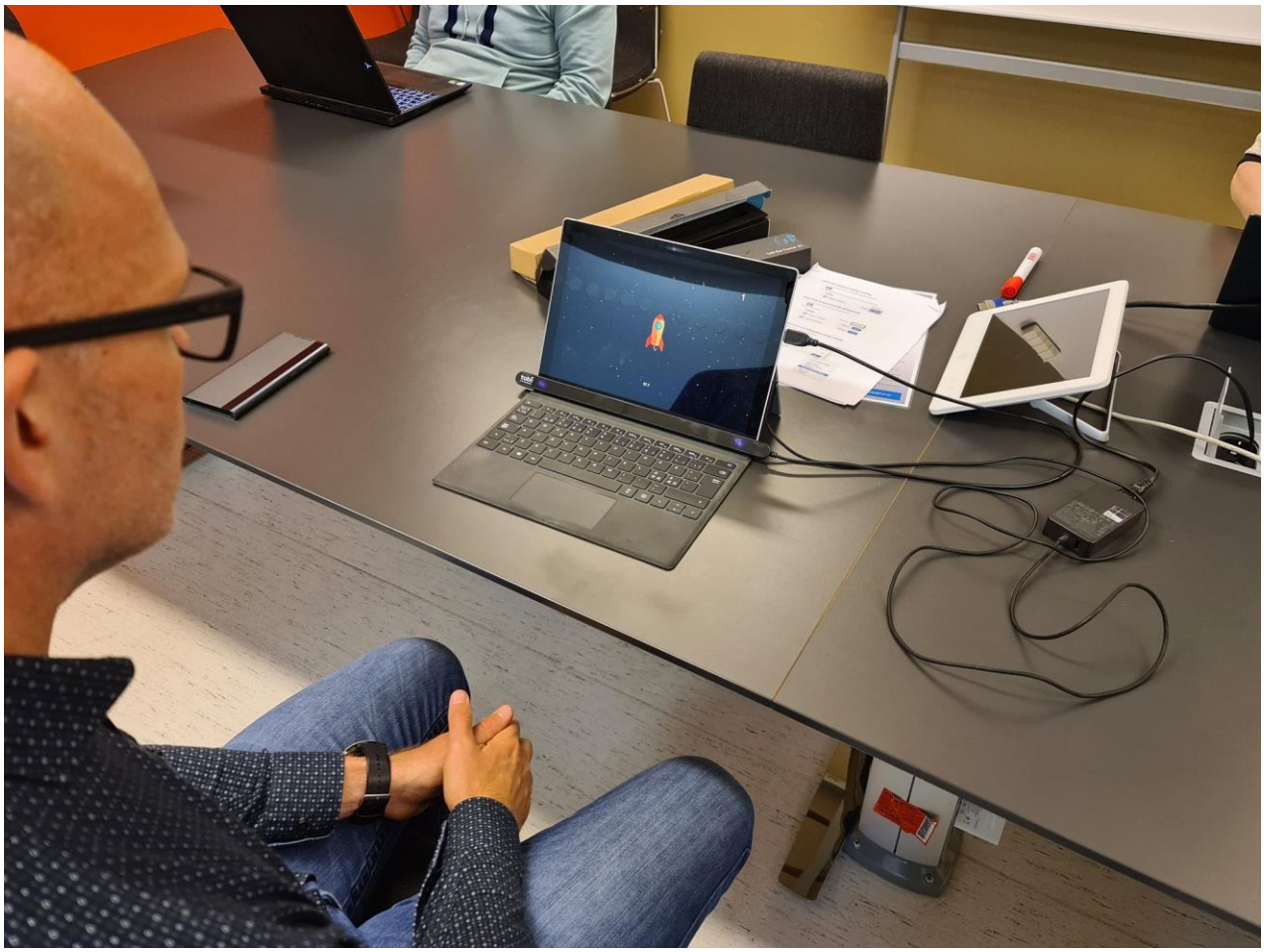


Figure 5.4 - One of the participants playing game 3 – "save the rocket".

The next participant, a male, had a difficult time playing game 2 (the labyrinth, as seen in figure 5.5) because his eyes were very far apart, he could not follow the path smoothly and hit the wall several times before leading the eyes to the goal. There was a possibility to "cheat" and close one of the eyes and only play with one eye, which seemed to work for him. He experienced the same issue in game 7, where he could not draw with both of his eyes, since these were not coordinated, but with closing one of his eyes and drawing only with one he managed to solve the game. This participant really enjoyed game 3, and despite of having OMD problems, it worked well.

Our last participant, female and vision educator, gave several, important feedback for us. For game 4 it was suggested that adding a trace in the replay of how the eye movement scanned for the object to see if the scanning was systematic or chaotic. The vision educator feels that game 7 would be good for children since it is about drawing and that is a fun thing for children. Game 1 and 6 might be harder for older people, as it uses either keyboard or the mouse in conjunction with playing the games with your eyes, but it was also mentioned that this would probably not be an issue for younger people as they tend to grasp mechanics like this easier and faster.

Adding more games to fit a broader aspect of the training, and variety of the experience, would be a good thing for the future of this program. It was stated that there is a belief that the most important thing with this type of training is that it should not become stale or boring after some

time as this makes the patient lose interest and makes the training less effective, or not effective at all.

In general, the vision educator said that implementing more feedback to the player would make the games more fun, and less stale over time. It was mentioned showing an indication that the player is getting points as well as having sound effects, applause, cheers, and more. This would encourage patients to play more and feel a sense of accomplishment, which makes the patients want to play and train more. Breaks in-between games to relax the eye muscles would be a great addition. Overall, the vision educator was happy with the games, and she also thinks there is a future for this in OMD training.

The group experienced that the description of the games were not particularly good, especially for the more complex games. The interface seemed to be a good user experience for new people to the program as the navigation from scene to scene was obvious and were effortless.

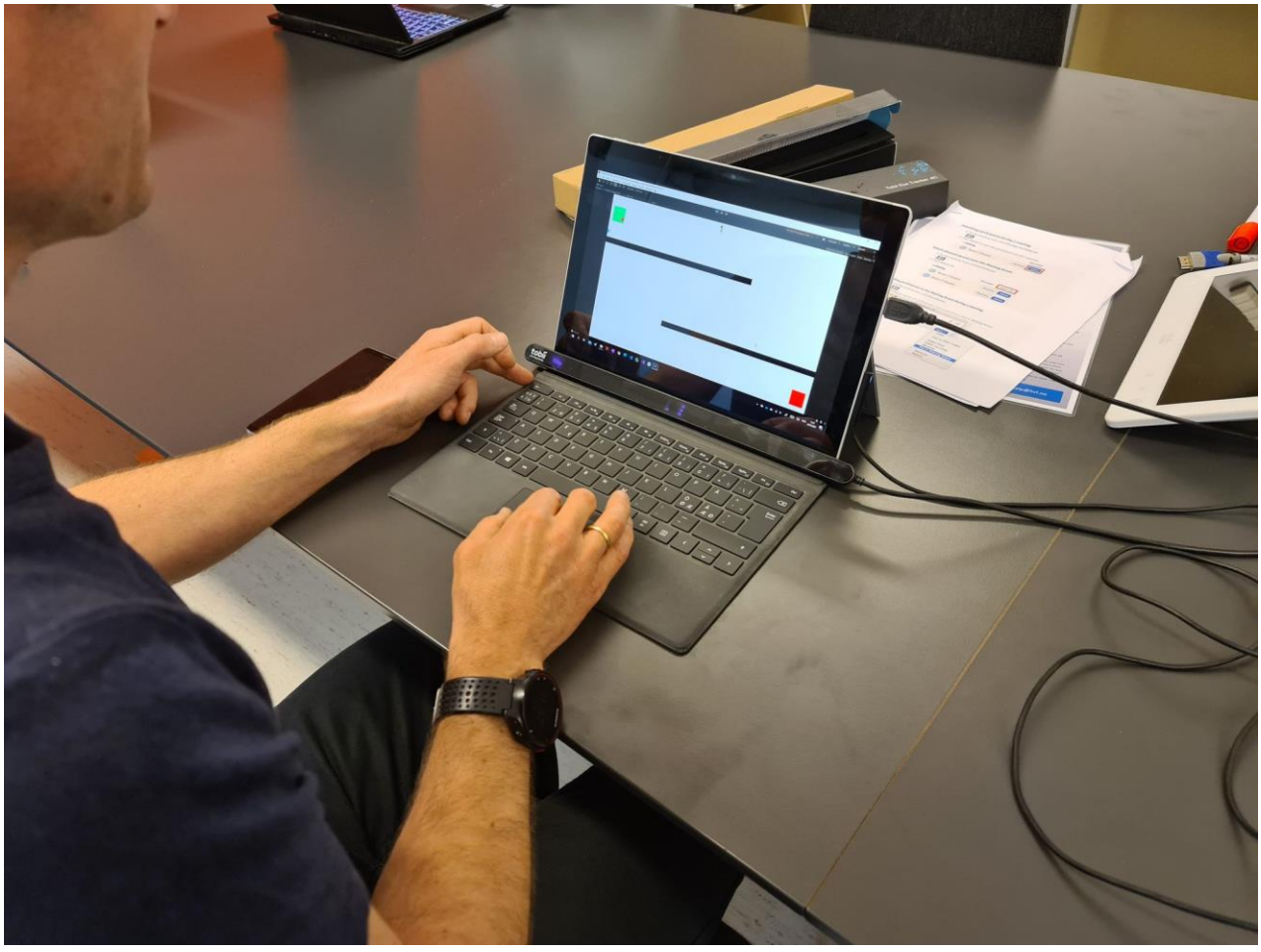


Figure 5.5 – Another participant playing game 2 – "the labyrinth".

6 DISCUSSION

At the end of the project the group agrees that the main goals with the project were reached. To implement a solution that can be used by professionals in the Healthcare sector and eventually by other non-professionals, this cannot be a short-term project and the product can be used by the intended users. This project reaches its goals by developing a program that can support training of eyes and demonstrating its feasibility.

While the target population of this product are school-aged children, the group can only have the hypotheses that the product will suit the children at this moment. Children with vision problems must be advised to use training by professionals at this moment. Therefore, their opinions on the usefulness of the games are more important at this stage. They need to be able to train with this product. Since the product is games, satisfying guidelines for training the eyes (Scheiman, 2011) and considering aspects for gamification, the intention is that in the next step the games would be tested by children as well.

Due to the Covid-19 situation and intensive programming work required for this project the group planned the user experience evaluations to test the software very late. Therefore, the feedback on improvements of the program must be mentioned in the further work chapter. An additional limitation that this group had to deal with was the limited access to experts with competence within vision impairments and possibilities to test with additional vision experts. Analysing the eye data was an extra step the group tried to accomplish but it was not accurate enough, even though fixation has been implemented and is shown in the graphs. To measure and examine other eye-movement data is suggested to be done to complement this work as a first step after this project in the future.

The group started early on the project, and this helped getting familiar with the games and using Unity. The goals were very “blurry”, and the group set up many meetings with Ilona and Qasim in the beginning to help make the goals clearer and more concise. Starting early on development meant that the group had time to figure out the best solutions for saving data in a database and working with this data and so on. For example, using EF Core is very useful as it is easier to change to other database providers and it is also easy to add new tables and columns in the model that was created for the program. However, more work should have been done on testing the different databases and for using it in the program.

The group also might have spent too little time on finding a good UI system for Unity, as there are more options to use other then only Unity’s own systems and tools, and the one used now is still in preview so it might be unstable and is missing options that would be nice to create better interfaces.

Another thing the group could have done more was to research the C&Look program more than what the group ended up doing, which was very little. The program is very

large and complicated and there could be several components and methods that might have been possible to use in this project.

Unity's unit tests and automatic tests were not used for testing the correctness of the code because of time limits and not enough knowledge by the group.

In addition, the group had no previous knowledge in this field, both about OMD and using Unity. This made trying to accomplish some of the goals, like analysis and improvements of the games, at the very end of the project hard and difficult to do.

There could have been spent more time to have made the games even better, and especially the analysis, which still needs improvement. Improvement of games would be more useful for the vision teachers and patients that could be using the program.

7 CONCLUSIONS AND FURTHER WORK

7.1 Conclusions

The overall goal of this project was to design and develop an Eye Tracking based program including serious games that can supplement vision experts to help them to plan training of school aged children, a vulnerable group where a large percentage has vision problems and need help. The project had the following subgoals, and reached them in the following way:

1. Create a database. The first subgoal is to expand the program to save information and data from the games using an Eye Tracker in a database, the basic information needed to register, handle, and examine training data from the different users. A database was created through EF Core and SQLite, which handles the saving and storing of data and makes it possible for retrieving data for later use. A component was also created for saving data from the games into the database.
2. Develop a user interface. A new interface is needed for vision experts and users who use the games for training their eyes. A user interface was created with Unity UI Toolkit, with interfaces that shows user-data and game session information from the games. The group also improved the interface to start and run the individual games.
3. Visualizing eye-activities during playing. Implementing a "replay function" that superimpose eye-movements on activities performed on the screen. A replay function was created through a universal component that is implemented and works in every game.
4. Improve existing games. Six existing games had to be further improved for better usability. The group has done improvements to the games, by making them easier to use, more challenging by including game elements, and better correlated to existing training by measuring important eye movements and is shown to possibly be used in vision treatment. A new game was also created. Fixations are measured, but issues were

observed in relation to registering and measuring it. In relation to measuring eye movements this will be suggested in future work.

5. Examine feasibility. The group tested that the program was in working condition with iterative development. The program was evaluated by 3 users that gave the group a lot of feedback on what needs to be done to further develop the program as well as usability.

By reaching these five goals the group believes that there has been made contributions to developing the games so that the program can be used for training vision impairments with focus on OMD.

7.2 Further work

One of the main benefits of this project is to define some concrete steps for future work.

While the group have done a lot and implemented many features, further work will be needed for this project. A better and more accurate method to analyse and measure concrete eye data from the Eye Tracker needs to be researched, developed, and implemented into the program. Measuring other types of data like saccades and smooth pursuit are needed as well. The problem with noisy eye data when measuring different distances and the total times travelled with the eyes, also needs further work.

The games need more improvement to make them more engaging and fun. Some examples are more varied levels in game 2 (the labyrinth), more figures in game 4 and 7, and that one can choose a frequency for distractions and the speed of the ball in game 5. The biggest feedback the group got from the evaluation was that the games should be more fun and interactive, through use of visual effects, sound effects, animations, and more, so that the game can keep the user engaged and be more willing to use the games for training their eyes over an extended period.

Other work that should be done is to change the database provider for EF Core from SQLite to a database that is not based on a database file but runs as a server on the local computer or in the cloud, like MySQL, PostgreSQL, or Microsoft SQL Server where all have packages for EF Core as per the Microsoft documentation (Microsoft, 2021). Alternatively, one could change the database or ORM library altogether and save information in a different way, like using JSON, XML or something else.

8 LITERATURE/REFERENCES

8.1 Literature

Ali, Q. et al. (2020) Using Eye-tracking Technologies in Vision Teachers' Work – a Norwegian Perspective, *International Conference on e-Health and Bioengineering (EHB),* pp. 1-5. Available at: https://ieeexplore.ieee.org/abstract/document/9280169

Eide, M.G. and Watanabe, R. (2017) *Detecting oculomotor problems in children using eye tracking.* MSc Thesis. Bergen: Western Norway University of Applied Sciences & University of Bergen. Available at: https://ieeexplore.ieee.org/document/8969956

Eide, M.G. et al. (2019a) Detecting oculomotor problems using eye tracking: Comparing EyeX and TX300, *10th IEEE International Conference on Cognitive Infocommunications (CogInfoCom)*, pp. 381-388. Available at: https://ieeexplore.ieee.org/abstract/document/9089895

Eide, M.G. et al. (2019b) Eye-tracking complementing manual vision screening for detecting oculomotor dysfunction, *E-Health and Bioengineering Conference (EHB),* pp. 1-5. Available at: https://ieeexplore.ieee.org/document/8969956

Falkenberg, H.K., Langaas, T. and Svarverud, E. (2019) Vision status of children aged 7–15 years referred from school vision screening in Norway during 2003–2013: a retrospective study, *BMC Ophthalmol*, 19(180). Available at: https://doi.org/10.1186/s12886-019-1178-y

Heldal, I. et al. (2021) Supporting School Aged Children to Train Their Vision by Using Serious Games, *Computers*, 10(4), pp. 53. Available at: http://dx.doi.org/10.3390/computers10040053

Laugwitz, B., Held, T. and Schrepp, M. (2008) Construction and Evaluation of a User Experience Questionnaire, in Holzinger, A. (ed.) *HCI and Usability for Education and Work*. Berlin, Heidelberg: Springer Berlin Heidelberg, pp. 63–76. Available at: https://link.springer.com/chapter/10.1007/978-3-540-89350-9_6

Pettersen, H.H. (2018) *Design and Development of Game-based Oculomotor Training using Eye Tracking*. MSc Thesis. Bergen: Western Norway University of Applied Sciences & University of Bergen. Available at: https://bora.uib.no/bora-xmlui/handle/1956/19623

Scheiman, M. (2011) *Understanding and Managing Vision Deficits: A Guide for Occupational Therapists*. 3rd ed. Thorofare: Slack Incorporated.

8.2 References

French, J. (2020) *How to load a new Scene in Unity (with a Loading Screen)*. Available at: https://gamedevbeginner.com/how-to-load-a-new-scene-in-unity-with-a-loading-screen/ (Accessed: 01. June 2021)

Haraldseth Software (2021) *About*. Available at: https://visionbuilder.com/about.html (Accessed: 18. May 2021)

Johnson, R. (2016) *Oculomotor Dysfunction*. Available at: https://www.advancedvisiontherapycenter.com/about/blog/e_930/Signs-of-a-Vision-Problem/2016/7/Oculomotor-Dysfunction (Accessed: 29. May 2021)

Lexplore (2021) *Lexplore – New AI Reading Analytics*. Available at: https://www.lexplore.com/ (Accessed: 18. May 2021)

marker software (n.d.) *Info*. Available at: http://www.markersoftware.com/USA/frames.htm (Accessed: 29. May 2021)

MedlinePlus (2020) *Vision Screening*. Available at: https://medlineplus.gov/lab-tests/vision-screening/ (Accessed: 2. June 2021)

Microsoft (2021) *.NET implementations supported by EF Core*. Available at: https://docs.microsoft.com/en-us/ef/core/miscellaneous/platforms (Accessed: 01. June 2021)

Microsoft (2021) *Creating and configuring a model*. Available at: https://docs.microsoft.com/en-us/ef/core/modeling/ (Accessed: 03. June 2021)

Microsoft (2021) *Database Providers*. Available at: https://docs.microsoft.com/en-us/ef/core/providers/?tabs=dotnet-core-cli (Accessed: 29. May 2021)

Microsoft (2021) *DbContext Lifetime, Configuration, and Initialization*. Available at: https://docs.microsoft.com/en-us/ef/core/dbcontext-configuration/ (Accessed: 01. June 2021)

Microsoft (2021) *Installing Entity Framework Core*. Available at: https://docs.microsoft.com/en-us/ef/core/get-started/overview/install (Accessed: 01. June 2021)

Microsoft (2021) *Stopwatch class*. Available at: https://docs.microsoft.com/en-us/dotnet/api/system.diagnostics.stopwatch?view=netstandard-2.0 (Accessed: 02. June 2021)

monitor1394 et al. (2021) *XCharts: A charting and data visualization library for Unity.* Available at: https://github.com/monitor1394/unity-ugui-XCharts (Accessed: 01. June 2021)

One Mind PsyberGuide (n.d.) *COGPACK*. Available at: https://onemindpsyberguide.org/apps/cogpack/ (Accessed: 29. May 2021)

RightEye (2021) *Revolutionary Eye-Tracking Technology*. Available at: https://righteye.com/ (Accessed: 29. May 2021)

RightEye (2021) *Vision Care*. Available at: https://righteye.com/markets/vision-care/ (Accessed: 10. April 2021)

Smart Optometry (2020) *Vision Therapy with AmblyoPlay – Learn about your path to better vision*. Available at: https://www.amblyoplay.com/about/ (Accessed: 29. May 2021)

Takmaz, Ece (2017) *eyegaze*. Available at: https://github.com/ecekt/eyegaze (Accessed: 01. June 2021)

Tobii (2020) *Tobii Eye Tracking Data Transparency Policy*. Available at: https://transparency.tobii.com/ (Accessed: 18. April 2021)

Tobii (2021) *What's the difference between Tobii Eye Tracker 4C and Tobii* EyeX. Available at: https://help.tobii.com/hc/en-us/articles/212814329-What-s-the-difference-between-Tobii-Eye-Tracker-4C-and-Tobii-EyeX (Accessed: 01. June 2021)

Tobii Pro (2018). *Tobii Pro Upgrade Key for Tobii Eye Tracker 4C.* Available at: https://www.tobiipro.com/siteassets/tobii-pro/product-descriptions/tobii-pro-upgrade-key-product-description.pdf/?v=1.5 (Accessed: 18. April 2021)

Tobii Pro (2021) *.NET – Getting started*. Available at: http://developer.tobiipro.com/NET/dotnet-getting-started.html (Accessed: 01. June 2021)

Tobii Pro (2021) *Coordinate systems*. Available at: http://developer.tobiipro.com/commonconcepts/coordinatesystems.html (Accessed: 01. June 2021)

Tobii Pro (2021) *Products.* Available at: https://www.tobiipro.com/product-listing/ (Accessed: 10. April 2021)

Unity (2021) *Comparison of UI systems in Unity.* Available at: https://docs.unity3d.com/Manual/UI-system-compare.html (Accessed: 18. April 2021)

Unity (2021) *PlayerPrefs*. Available at: https://docs.unity3d.com/ScriptReference/PlayerPrefs.html (Accessed: 01. June 2021)

Unity (2021) *Screen*. Available at: https://docs.unity3d.com/ScriptReference/Screen.html (Accessed: 01. June 2021)

Vivid Vision (2021) *Vivid Vision for lazy eye, crossed eye, and convergence insufficiency*. Available at: https://www.seevividly.com/ (Accessed: 29. May 2021)

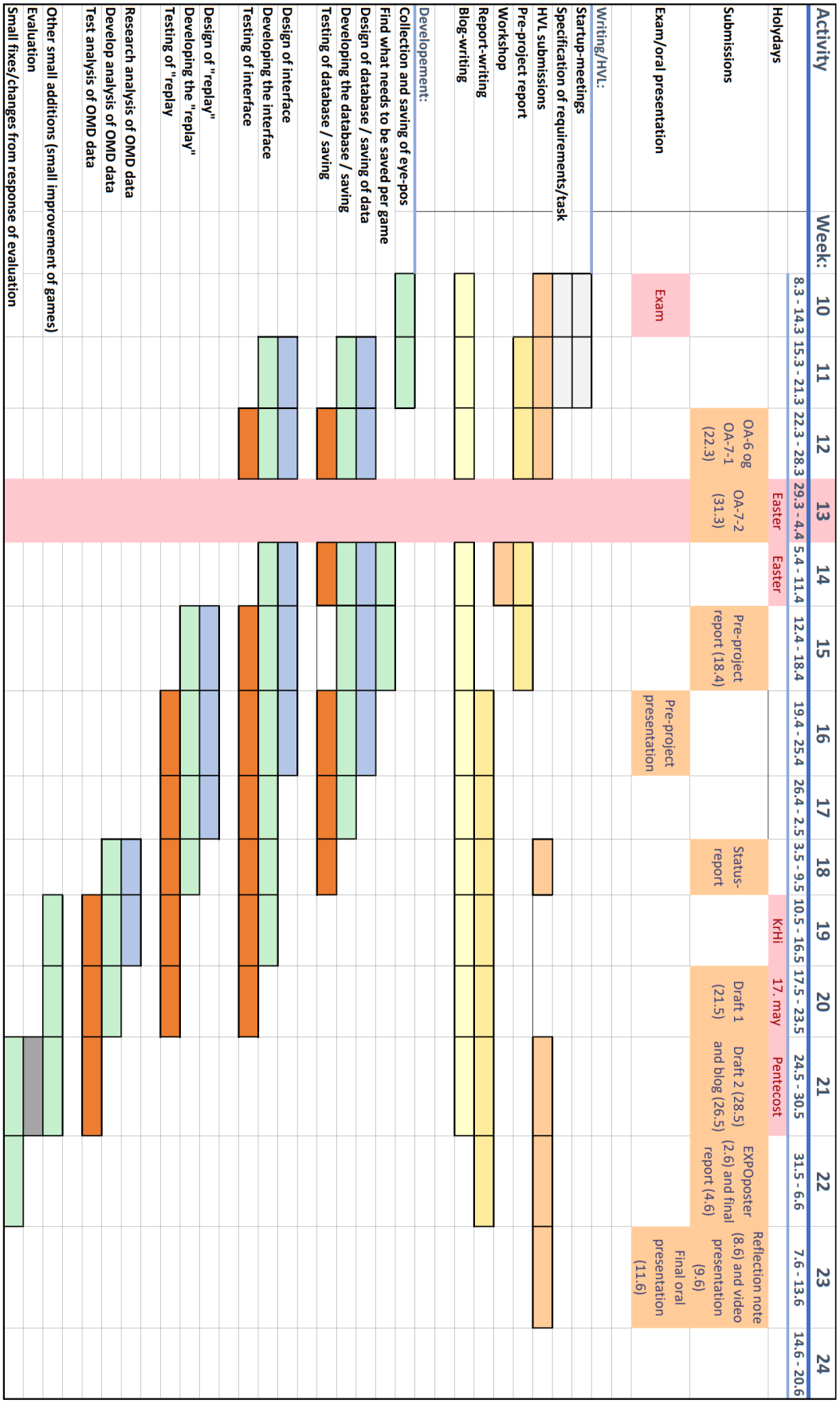
9 APPENDIX

9.1 Risk list

Risk	Description	L	C	RF	Preventive measures	If it happens
Application is not done.	The goals and requirements for the application are not finished.	2	4	8	Work together continuously and set goals for each week.	Write extensive about why and what in the report.
Misunderstandings	The team misunderstands what task to do in the project and misunderstand what the project is supposed to do.	2	5	10	Have daily meetings with the team and have weekly meetings with the supervisor.	Make sure to figure out what the misunderstanding is and solve it as soon as possible.
Report not finished in time.	The team do not finish the report in time.	1	5	5	Knowing deadlines, respecting them, and start working on the report in reasonable time in advance.	It must be finished in time.
Corona	Reduced testing on other people.	5	2	10	Follow national guidelines. Online test meetings.	Do the best we can out of the situation presented right in front of us.
Eye Tracker troubles	Eye Tracker stops working, other hardware or software issues.	2	3	6	Take good care of the device, keep drivers up to date.	Try to get new equipment from the supervisor.
Difficulties implementing parts of the program.	Issues with database and saving of data, replay of games, and more.	2	4	8	Do good research of the necessary parts and work continuously.	Ask other people at HVL for assistance, like Qasim.
Git problems	Using unnecessary time	4	1	4	Work together and not do	Clone project again, keep

	to fix Git push/pull/merge problems.				work in the same files. Keep multiple backups.	incoming changes and take backup of work done.
Measuring OMD data	Trouble with measuring fixation, saccades and more.	4	4	16	Look into what C&Look has done and if there are other tools to measure it.	Ask other people for assistance.
Unity is unknown	Trouble with using Unity and its functions.	4	2	8	Use online guides and the Unity documentation.	Ask for assistance.
Replay problems	Issues with sync of replay and eyedata. Issue with getting replay as close as possible to the games.	4	4	16	Test after doing a change with the replay. Try to use the game code for replay as well.	Ask for assistance.

9.2 GANTT diagram



9.3 Performance test results

Test 1 (small -> large)

		Surface Book 2				Desktop PC			
Time	Count	Time to save	Time to analyze	Time to retrieve(+sort)	DB size increase	Time to save	Time to analyze	Time to retrieve(+sort)	DB size increase
30s	2700	1264	28	13	163KB	3252	129	18	163KB
60s	5400	3075	73	16	331KB	7404	479	29	331KB
90s	8100	5144	192	33	491KB	12513	1049	42	491KB
120s	10800	8000	321	33	663KB	18668	1847	55	663KB
150s	13500	11122	443	36	847KB	25791	2837	79	847KB
180s	16200	14722	608	60	1015KB	34033	4101	88	1015KB
210s	18900	18792	838	63	1187KB	43076	5514	108	1187KB
240s	21600	24474	1042	73	1351KB	52770	7189	114	1351KB
270s	24300	29069	1319	67	1536KB	63937	9087	131	1536KB
300s	27000	34651	1562	77	1691KB	75720	11195	142	1691KB
330s	29700	38719	2244	71	1863KB	88886	13510	147	1863KB
360s	32400	47590	2310	107	2035KB	102610	16065	184	2035KB
390s	35100	57879	3264	129	2207KB	117631	18929	178	2207KB
420s	37800	64252	3904	131	2367KB	133738	21849	193	2367KB
450s	40500	74644	4758	140	2547KB	150493	25079	270	2547KB
480s	43200	84043	5571	135	2711KB	168725	28550	231	2711KB
510s	45900	95745	6033	142	2883KB	187307	32304	254	2883KB
540s	48600	104946	7317	166	3051KB	207285	36276	255	3051KB
570s	51301	113776	7459	169	3219KB	227704	40187	272	3219KB
600s	54001	124867	9149	203	3391KB	249840	44737	279	3391KB

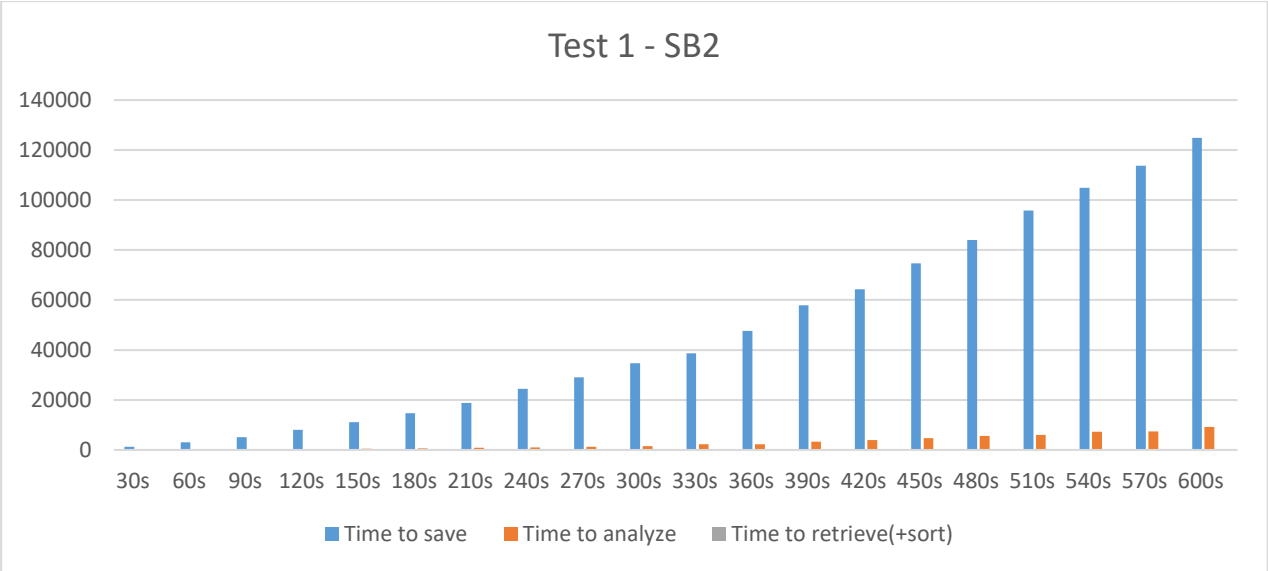
Test 2 (small -> large + async)

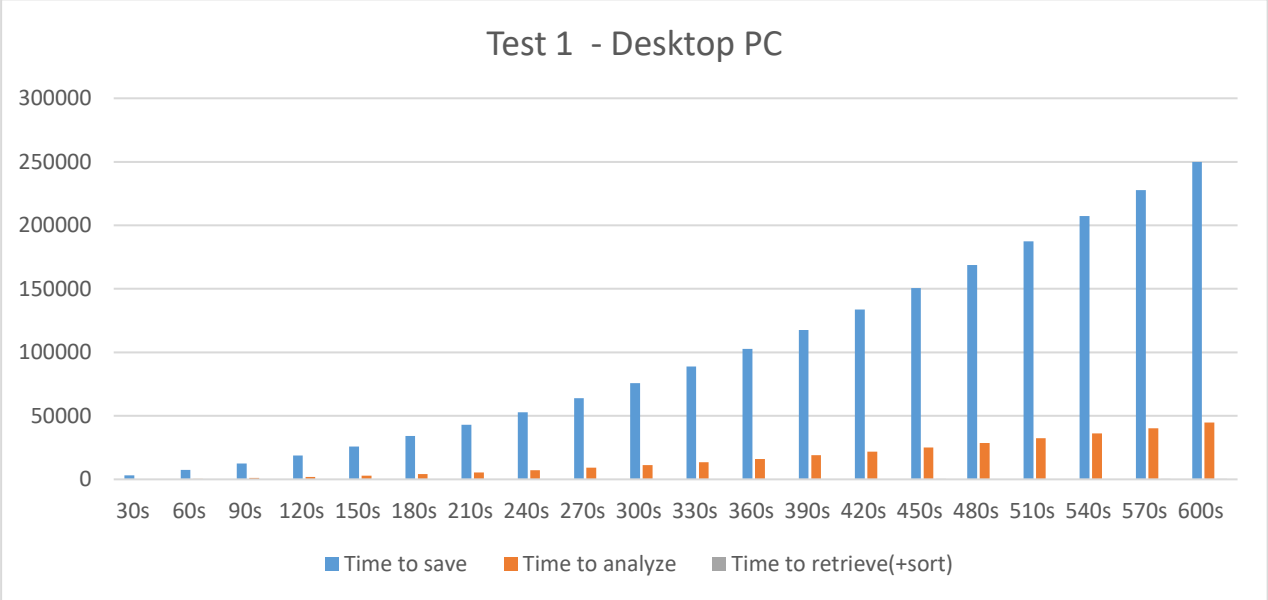
		Surface Book 2			Desktop PC		
Time	Count	Time to save+analyze	Time to retrieve(+sort)	DB size increase	Time to save+analyze	Time to retrieve(+sort)	DB size increase
30s	2700	1603	13	163KB	3636	19	163KB
60s	5400	3489	20	331KB	8564	31	331KB
90s	8100	6484	28	491KB	14978	42	491KB
120s	10800	9634	63	663KB	22637	57	663KB
150s	13500	13995	46	847KB	32035	72	847KB
180s	16200	16463	46	1015KB	42558	86	1015KB
210s	18900	21770	57	1187KB	54268	100	1187KB
240s	21600	26049	65	1351KB	67089	110	1351KB
270s	24300	32319	64	1536KB	81738	135	1536KB
300s	27000	37383	69	1691KB	98328	138	1691KB
330s	29700	44772	84	1863KB	116719	153	1863KB
360s	32400	52992	96	2035KB	135152	185	2035KB
390s	35100	64735	112	2207KB	154362	183	2207KB
420s	37800	74820	132	2367KB	176129	205	2367KB
450s	40500	86623	153	2547KB	198902	205	2547KB
480s	43200	99030	178	2711KB	223365	243	2711KB
510s	45900	107392	145	2883KB	248179	227	2883KB
540s	48600	116516	171	3051KB	276379	278	3051KB
570s	51301	126535	159	3219KB	303871	266	3219KB
600s	54001	139480	203	3391KB	334716	278	3391KB

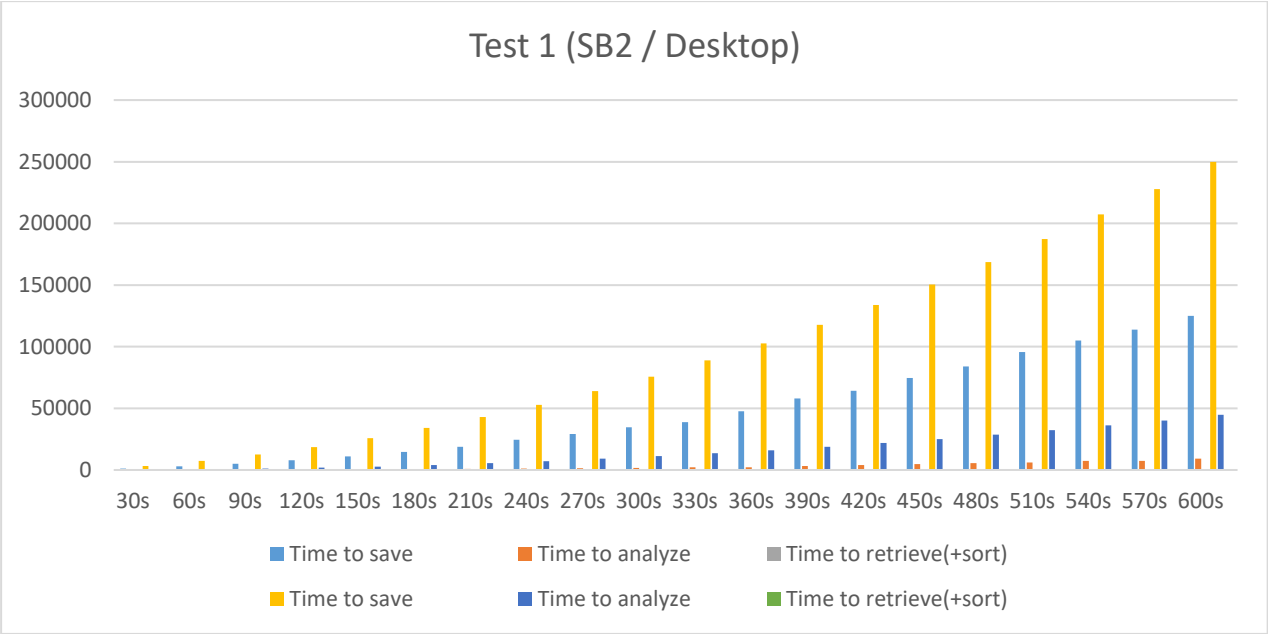
Test 3 (many small - 15 loops)

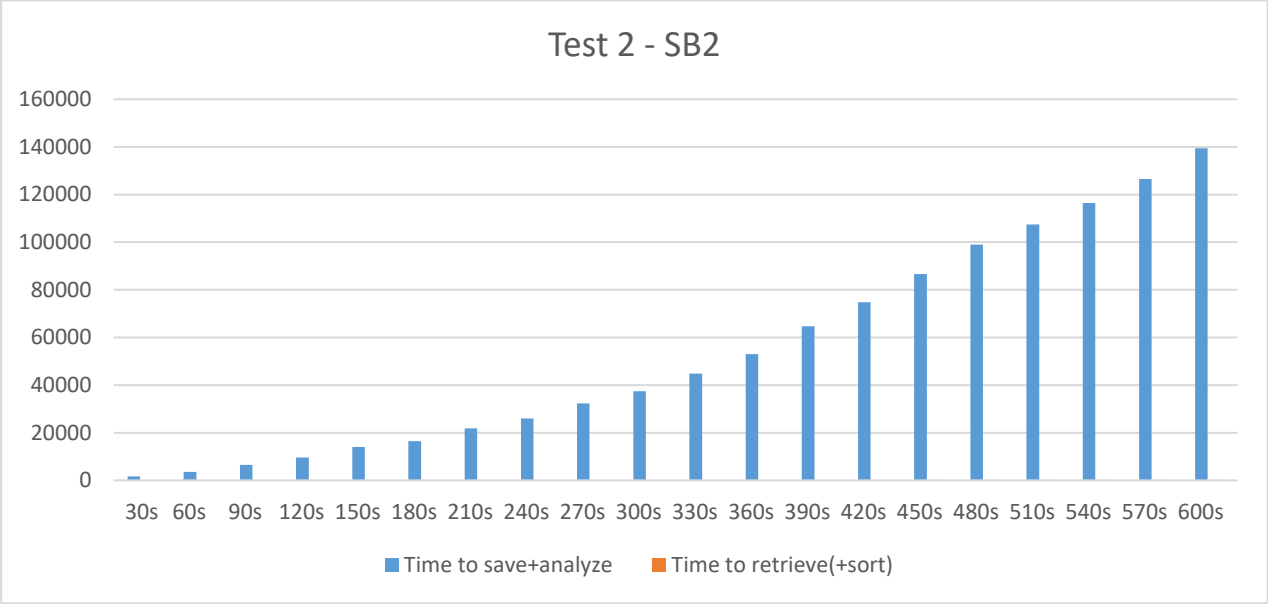
		Surface Book 2				Desktop PC			
Loop	Time	Time to save	Time to analyze	Time to retrieve(+sort)	DB size increase	Time to save	Time to analyze	Time to retrieve(+sort)	DB size increase
1	30s	1346	28	16	163KB	3233	128	17	163KB
	60s	2999	94	24	331KB	7418	476	29	331KB
	90s	5091	158	25	491KB	12483	1044	41	491KB
	120s	7803	287	28	663KB	18606	1827	61	663KB
2	30s	1990	26	13	163KB	5465	131	17	163KB
	60s	3445	84	26	339KB	9665	474	38	339KB
	90s	5794	164	25	512KB	14719	1040	46	512KB
	120s	8527	298	32	675KB	20904	1840	71	675KB
3	30s	2520	26	12	172KB	7790	128	20	172KB
	60s	4047	89	20	335KB	12033	472	31	335KB
	90s	6104	162	25	507KB	17049	1039	48	507KB
	120s	9320	279	39	679KB	23161	1820	68	679KB
4	30s	3393	37	9	167KB	10070	132	15	167KB
	60s	4746	92	23	335KB	14460	479	28	335KB
	90s	6754	159	25	507KB	19509	1056	48	507KB
	120s	9484	285	33	675KB	25451	1850	59	675KB
5	30s	4143	35	11	184KB	12389	129	15	184KB
	60s	5577	87	19	335KB	16514	483	32	335KB
	90s	7169	163	23	503KB	21634	1046	42	503KB
	120s	9966	272	53	679KB	27707	1839	55	679KB
6	30s	4814	26	10	167KB	14659	130	15	167KB
	60s	6302	88	15	339KB	18759	474	33	339KB
	90s	7869	164	21	507KB	23889	1045	41	507KB
	120s	10995	278	53	675KB	30048	1834	55	675KB
7	30s	5374	27	10	172KB	16903	128	15	172KB
	60s	6595	86	21	335KB	21070	476	36	335KB
	90s	8420	165	22	516KB	26138	1047	42	507KB
	120s	11621	272	30	675KB	32243	1834	55	684KB
8	30s	5854	26	16	180KB	19211	128	15	180KB
	60s	7144	85	23	348KB	23496	484	30	348KB
	90s	9306	165	23	528KB	28509	1058	42	528KB
	120s	12466	286	47	704KB	34662	1837	55	704KB
9	30s	6222	26	9	180KB	21506	129	15	180KB
	60s	8065	73	26	348KB	25694	477	29	348KB
	90s	10498	178	26	524KB	30750	1045	46	524KB
	120s	14084	325	32	708KB	36928	1834	59	708KB
10	30s	7426	29	21	172KB	23643	128	15	172KB
	60s	10508	94	28	352KB	27940	474	29	352KB
	90s	15466	242	35	528KB	33019	1043	65	528KB
	120s	16256	374	52	704KB	39143	1837	65	704KB
11	30s	9276	30	12	172KB	25922	131	15	172KB
	60s	11859	102	18	352KB	30077	472	33	352KB
	90s	13755	206	36	528KB	35246	1038	41	528KB
	120s	17719	335	59	700KB	41319	1830	55	700KB
12	30s	9420	32	10	180KB	28291	128	18	180KB
	60s	12348	84	18	352KB	32548	480	38	352KB

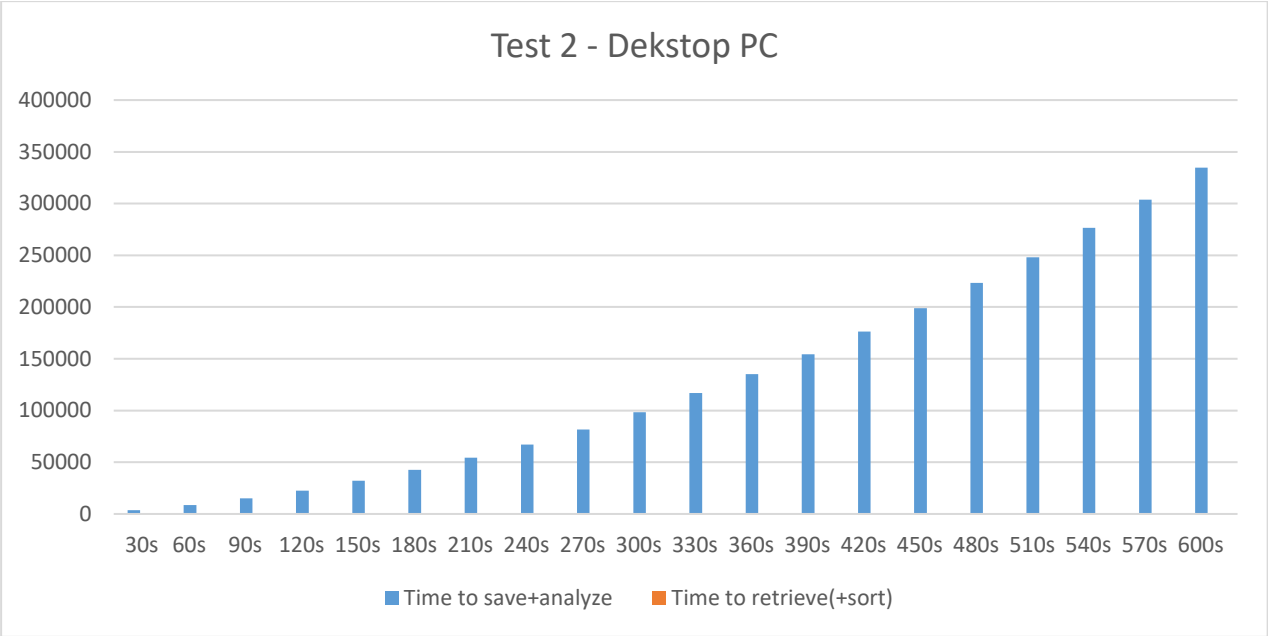
	90s	13731	193	24	524KB	37627	1059	50	524KB
	120s	17662	324	35	704KB	43649	1831	54	704KB
13	30s	10256	27	13	176KB	30449	129	20	176KB
	60s	12485	100	20	352KB	34769	476	30	352KB
	90s	15939	214	31	524KB	39803	1063	46	524KB
	120s	18900	369	33	700KB	45902	1837	55	700KB
14	30s	12364	28	14	184KB	32896	128	15	180KB
	60s	13214	91	45	348KB	36820	486	35	348KB
	90s	15590	275	23	528KB	42050	1040	53	532KB
	120s	17050	294	35	716KB	48102	1825	56	716KB
15	30s	10875	29	11	172KB	35098	132	15	172KB
	60s	11932	80	19	352KB	39236	475	30	352KB
	90s	14730	187	23	532KB	44177	1050	48	532KB
	120s	16762	280	39	700KB	50128	1820	72	700KB

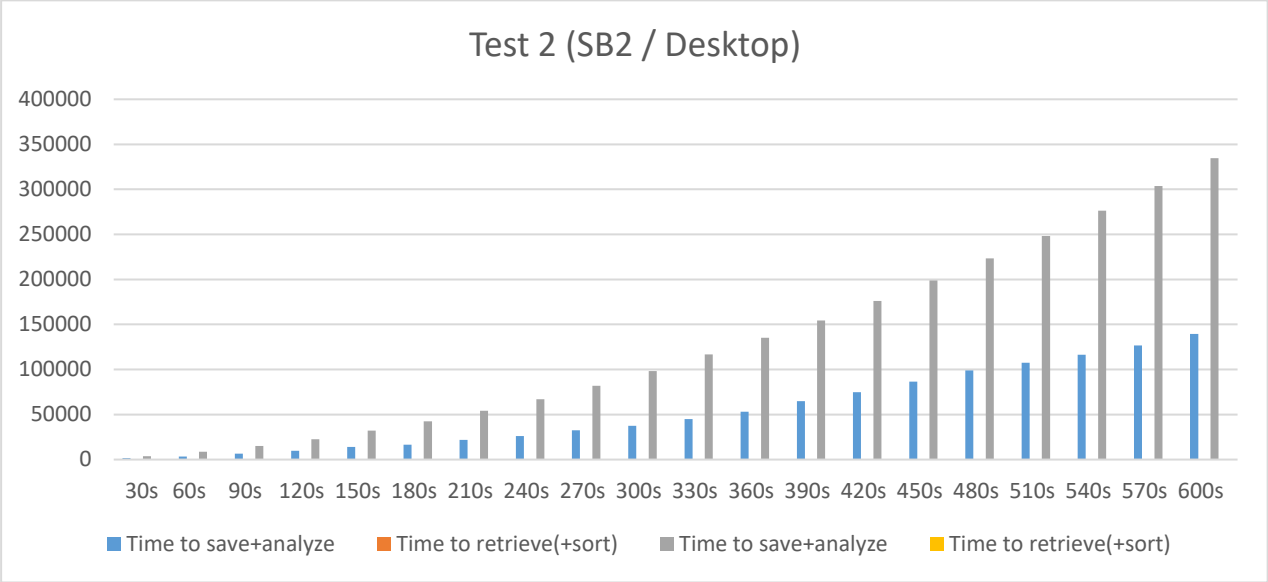


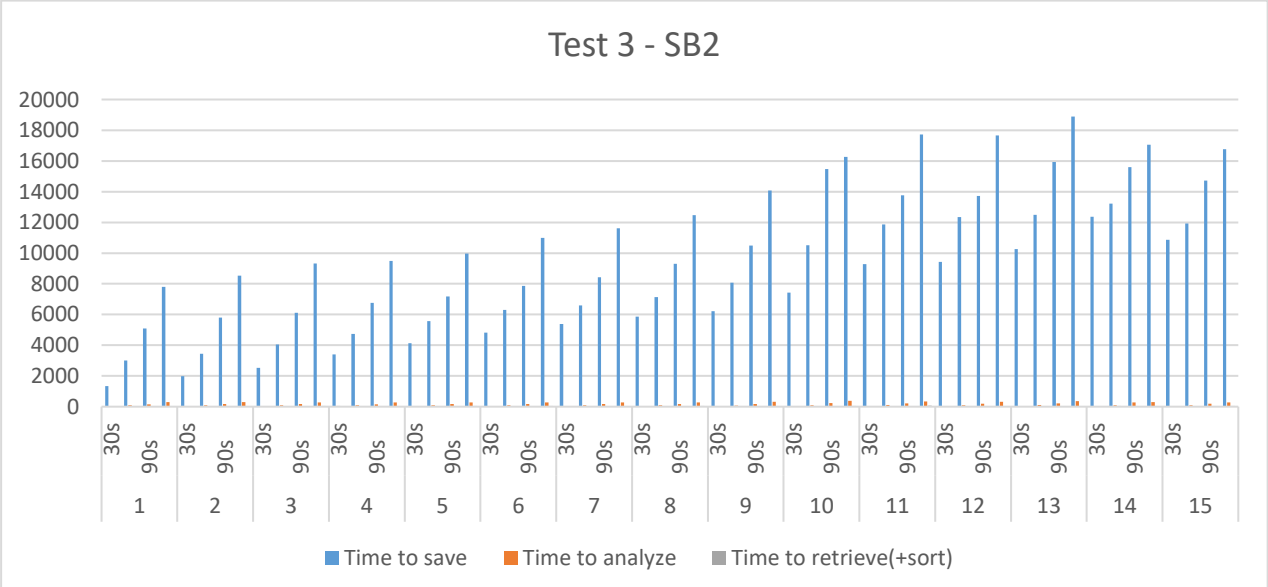


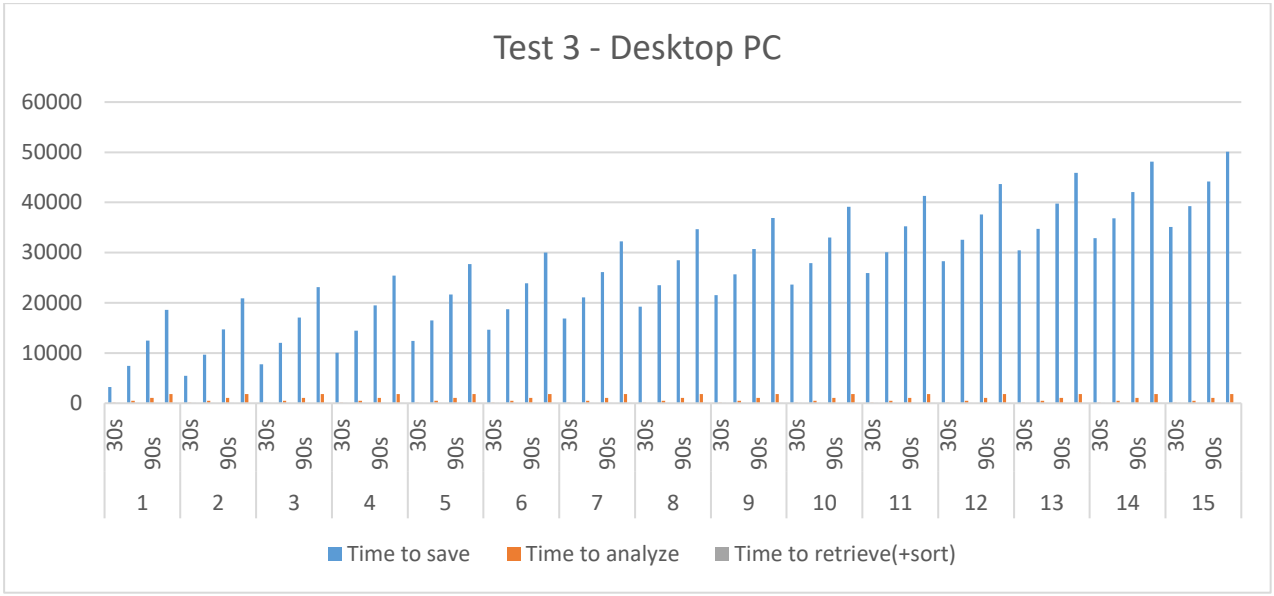

Test 1 (SB2 / Desktop)
300000
250000
200000
150000
100000
50000
0
30s 60s 90s 120s 150s 180s 210s 240s 270s 300s 330s 360s 390s 420s 450s 480s 510s 540s 570s 600s
Time to save
Time to analyze
Time to retrieve(+sort)
Time to save
Time to analyze
Time to retrieve(+sort)

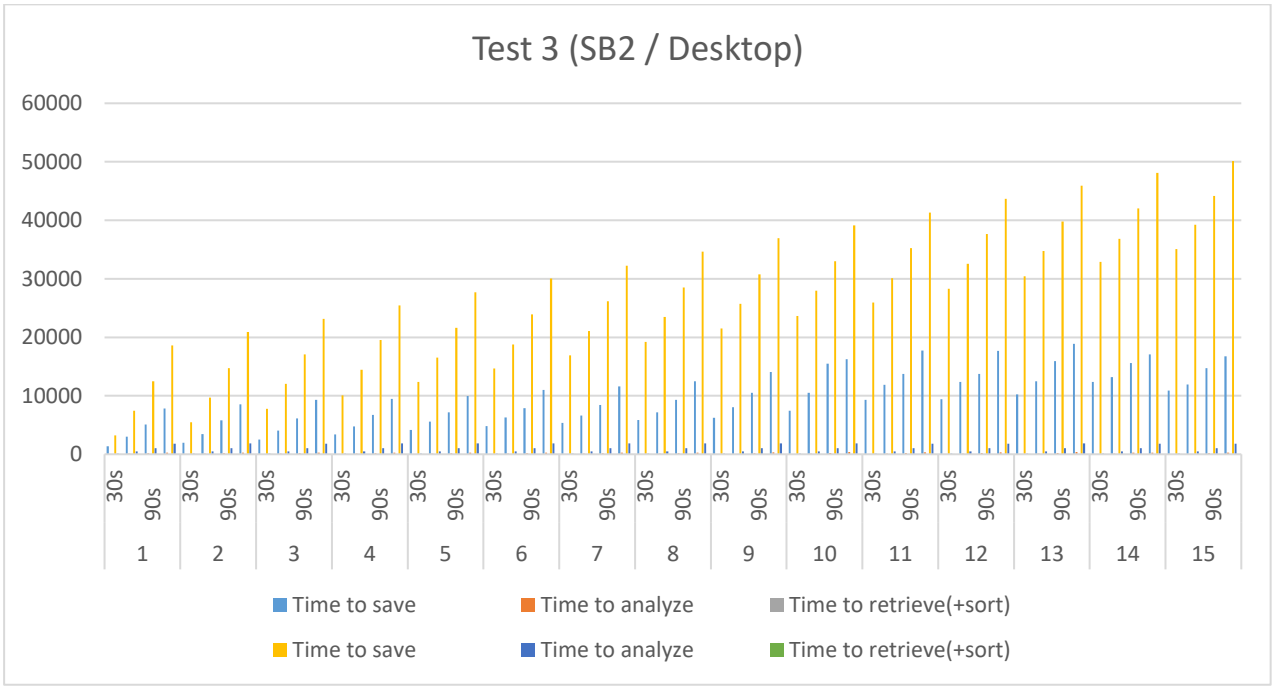

Test 2 - SB2
160000
140000
120000
100000
80000
60000
40000
20000
0
30s 60s 90s 120s 150s 180s 210s 240s 270s 300s 330s 360s 390s 420s 450s 480s 510s 540s 570s 600s
Time to save+analyze
Time to retrieve(+sort)


Test 2 - Dekstop PC
400000
350000
300000
250000
200000
150000
100000
50000
0
30s 60s 90s 120s 150s 180s 210s 240s 270s 300s 330s 360s 390s 420s 450s 480s 510s 540s 570s 600s
Time to save+analyze
Time to retrieve(+sort)


Test 2 (SB2 / Desktop)
400000
350000
300000
250000
200000
150000
100000
50000
0
30s 60s 90s 120s 150s 180s 210s 240s 270s 300s 330s 360s 390s 420s 450s 480s 510s 540s 570s 600s
Time to save+analyze
Time to retrieve(+sort)
Time to save+analyze
Time to retrieve(+sort)


Test 3 - SB2
20000
18000
16000
14000
12000
10000
8000
6000
4000
2000
0
30s 90s
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15
Time to save
Time to analyze
Time to retrieve(+sort)


Test 3 - Desktop PC
60000
50000
40000
30000
20000
10000
0
30s
90s
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
Time to save
Time to analyze
Time to retrieve(+sort)


Test 3 (SB2 / Desktop)
60000
50000
40000
30000
20000
10000
0
30s
90s
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
Time to save
Time to analyze
Time to retrieve(+sort)
Time to save
Time to analyze
Time to retrieve(+sort)