UNIVERSIDADE D
COIMBRA

João Dinis Duarte Soares de Sousa

Fun2HelpElderly

Cognitive stimulation and rehabilitation of hand-eye coordination in the elderly through serious games

Dissertação no âmbito do Mestrado Integrado em Engenharia Eletrotécnica e de Computadores, no ramo de especialização em Computadores, orientada pelo Professor Rui Paulo Pinto da Rocha, apresentada ao Departamento de Engenharia Eletrotécnica e de Computadores da Faculdade de Ciências e Tecnologia da Universidade de Coimbra.

Fevereiro de 2022

This page intentionally left blank.

1 2  9 0 UNIVERSIDADE D COIMBRA

Fun2HelpElderly - Cognitive Stimulation and Rehabilitation of Hand-Eye Coordination in the Elderly Through Serious Games

João Dinis Duarte Soares de Sousa

Dissertation submitted to the Electrical and Computer Engineering Department of the Faculty of Science and Technology of the University of Coimbra in partial fulfillment of the requirements for the Degree of Master of Science in Electrical and Computer Engineering

Supervisor: Prof. Rui Paulo Pinto da Rocha

Jury

President: Prof. Dr. Jorge Miguel Sá Silva
Vogals: Prof. Dr. Paulo Jorge Carvalho Menezes
Prof. Rui Paulo Pinto da Rocha

February 2022

This page intentionally left blank.

Acknowledgments

Em primeiro lugar quero agradecer ao meu orientador Prof. Rui Paulo Pinto da Rocha, por todo o apoio, disponibilidade, e orientação nesta fase final do meu percurso académico.

Gostaria também de agradecer ao Instituto de Sistemas e Robótica, por me ter disponibilizado todos os meios e condições necessários para a elaboração da parte prática deste trabalho. Deixo também um agradecimento a toda a comunidade do DEEC, colegas, professores e funcionários que se cruzaram comigo e que de alguma maneira contribuíram para que este percurso tenha acabado da forma desejada.

Não menos importante, quero também realçar a importância do apoio incondicional dado pela minha família e amigos. Guardarei com carinho todos os momentos partilhados no decorrer deste trajecto.

Por último, um agradecimento especial à Cáritas Diocesana de Coimbra, assim como a todos os idosos que disponibilizaram um pouco do seu tempo para me auxiliar nas etapas finais deste trabalho. Toda a ajuda prestada foi fulcral para que esta dissertação pudesse ser terminada.

Os vossos ensinamentos e conselhos foram fundamentais para perpetuar este percurso.

Um sincero obrigado,

João Dinis Sousa

This dissertation was supported by the EuroAGE project, funded by Interreg V A Spain - Portugal program under Grant No. POCTEP-0043_EUROAGE_4_E. It also had the contribution of the Institute of Systems and Robotics(ISR) – University of Coimbra, funded by "Fundação para a Ciência e a Tecnologia" (FCT) under GrantNo. UIDB/00048/2020

A todos,
Muito Obrigado!

Abstract

Technology has been exponentially evolving through the last decades, and as a result of that, our lives have drastically changed due to its undeniable impact in every dimension and field of our lives, including our daily habits and routines. Thus, it is easily acknowledged that the mentioned technological evolution provides innovative and creative ways to solve problems that have been around since the beginning of humanity through new techniques and gadgets. However, this information can only be helpful if used by someone who understands it and comprehends its true potential, hence the importance of familiarizing the maximum amount of possible people who currently do not have an average technology knowledge level. Due to this lack of familiarization, elderly people happen to represent a big portion of the latest group, and curiously, they are also the ones that could benefit the most from the evolution that has been witnessed over the past few years, considering their physical and cognitive debilities.

Adding to the previous idea, population average age is gradually increasing, which means each year that passes by, senior population numbers are becoming superior to the younger ones. So, with this group of people representing such a big percentage of the population, answering their needs should be a priority, considering that an improvement in this age group is going to affect a big portion of the world population, not only directly strengthening society health conditions, but also increasing their overall life quality.

Considering what was previously mentioned, this dissertation aims to take a leap regarding physical and cognitive rehabilitation of elder people, which is desired to be achieved through the development of an easy-to-play and understandable serious game, that aims to reach and engage the seniors through a virtual, fun and lighthearted experience, trying to make justice to the project name, Fun2HelpElderly. Knowing that luring elders into playing a virtual game can be a tough yet challenging task, due to their lack of technology familiarity, studies prove that it is indeed possible to turn a likely disengaging rehabilitation therapy, or certain physical activity experience into a fun regular habit, while requiring almost no professional assistance, as it will be seen forward in this dissertation. All in all, this serious games intends to fulfill an important health role, through the stimulation of different emotions and use of techniques that traditional

therapies may not provide and/or satisfy.

Keywords

Serious Games, Physical Rehabilitation, Human-Machine Interaction, Cognitive Stimulation

Resumo

A tecnologia tem evoluído exponencialmente ao longo das últimas décadas, e como resultado disso, as nossas vidas mudaram drasticamente devido ao seu impacto inegável em todas as dimensões, nas quais estão incluídos os nossos hábitos e rotinas diárias. Assim, é facilmente reconhecível que a referida evolução tecnológica proporciona formas criativas e inovadoras de resolver problemas existentes desde o início da humanidade, através da utilização de novas técnicas e dispositivos tecnológicos. Contudo, esta informação apenas será útil caso seja utilizada por alguém que a compreenda, salientando assim a importância de familiarizar o maior número possível de pessoas que atualmente não possua um nível médio de conhecimento tecnológico. Os idosos representam uma grande parte do grupo anterior, e curiosamente, são também eles quem mais poderia tirar proveito da evolução que temos vindo a testemunhar nos últimos anos, tendo em conta as suas debilidades físicas e cognitivas.

A idade média da população está a aumentar gradualmente, o que significa que a cada ano que passa, os números da população sénior tornam-se mais elevados. Assim, com esta faixa etária a representar uma percentagem tão elevada da população, responder às suas necessidades deve ser uma prioridade, considerando que uma melhoria neste grupo etário vai afetar uma grande parte da população mundial, não só reforçando directamente as condições de saúde da sociedade, mas também melhorando a sua qualidade de vida no geral.

Dado o exposto, esta dissertação visa ter um impacto positivo na reabilitação física e cognitiva dos idosos, através do desenvolvimento de um jogo sério, compreensível e fácil de jogar, que pretende cativar e entreter os idosos através de uma divertida experiência virtual que faça justiça ao nome do projecto, Fun2HelpElderly. É sabido que convencer os mais velhos a jogar um jogo virtual pode ser uma tarefa difícil e desafiante, devido à sua falta de familiaridade tecnológica. Contudo, estudos comprovam que é de facto possível transformar métodos de terapia e reabilitação desinteressantes num hábito regular e divertido, não requerendo qualquer assistência profissional, tal como se será explicado no decorrer desta dissertação. Em suma, este jogo sério pretende desempenhar um papel relevante na saúde da população, estimulando diferentes emoções e usando técnicas alternativas aquelas utilizadas nas terapias tradicionais já

conhecidas.

Palavras Chave

Estimulação Cognitiva, Interação Homem-Máquina, Jogo Sério, Reabilitação Física

Contents

This page intentionally left blank.

List of Figures

This page intentionally left blank.

List of Tables

This page intentionally left blank.

1

Manual de Utilizador

Contents

This chapter summarizes all the motivations and goals of this dissertation, while also providing a reading guide for contextualization purposes.

1.1 Motivation and context

Technology's influence is undeniable and does not go unnoticed on our everyday tasks, as it slowly gets more integrated into our gadgets, routines, and overall lifestyle. Therefore, it is quickly concluded that its impact and benefits on numerous areas is notorious, aiming to simplify, accelerate, and improve our daily lives.

It seems obvious how technology could take place in fulfilling an essential role in healthcare. As seen in [8], areas such as patient reach out, treatment facilities, decision making, disease diagnosis, rehabilitation, motoring, among others, already rely on technological support to improve its intrinsic value and efficiency. Due to the technology available in the rehabilitation area, it is possible to turn therapeutic exercises and tasks into appealing and engaging activities. Nevertheless, technology itself is not solving all existing problems. It should be used as an extension tool to apply scientific and logical thoughts towards a specific goal.

The scientific and medical communities can explore the previous idea in several forms, such as resorting to specialized robots [9], promoting various physical and cognitive stimulations [10], motion tracking & analysis tools through virtual reality resources [11], among many other ways. This dissertation will focus on the latter, more precisely, by creating an interactive virtual reality environment inside a developed serious game, where the player has to perform certain determined motions and reasonings to tackle the requested quests.

As a result of the previous analysis, this dissertation is seen as an opportunity window to serve real people's needs while utilizing academic knowledge. The developed serious game mainly focuses on elderly users, but could also be played by anyone who benefits from it. It aims to promote cognitive stimulation, cognitive rehabilitation, prevent impairments and improve the recovering progress of body members or other related handicaps. The motivational aspect of this dissertation resides in this serious game potential to become a step towards success inside an ageing society environment. It will allow older people to improve their health and familiarize themselves with technology, while circumventing boredom.

To complement what was previously mentioned, the following pie chart in Figure 1.1 – made with data obtained from [1] – shows how elder people practically represent one-quarter of the Portuguese population.

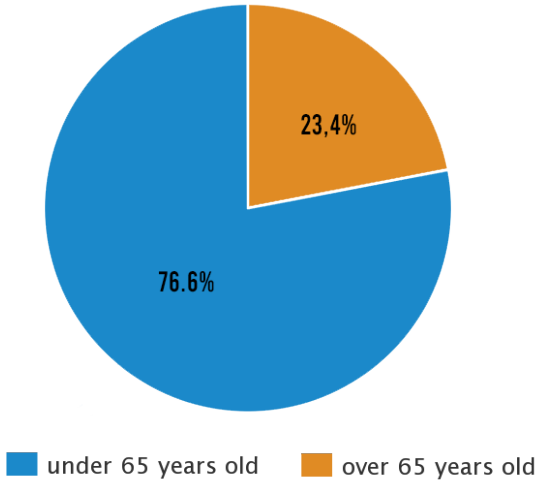


Figure 1.1: Percentage of population over and under 65 years old [1]

The information represented in the previous pie chart expresses how much of an impact technological advance directed to that specific age group could make.

1.2 Problem statement

Population average age is increasing gradually, and as a result, worsening the aged population numbers. According to [12] data, in about three decades, 47,1% of the Portuguese population will be over 55 years old – currently, those numbers sit at 35,1%. Therefore, to answer society's needs, and since elders already represent such a high population percentage, finding distinct ways of improving their healthcare and lifestyle is fundamental, and it will naturally influence and contribute to a community life quality overall improvement.

As described before, an ageing society is already a reality, becoming more pronounced every passing year. Even though older people represent such a significant percentage of the total population, this group is often the most neglected one. Unfortunately, the previously stated forgetfulness that older people are victim of, usually tends to result in an increasing disinterest towards healthcare and wellness chores. It is contradictory since, due to their natural limitations, elders should receive the most encouragement to try different types of stimulative activities that aim to improve their health and well-being. This disengagement is usually caused by a lack of monitoring, support, incentive, and information.

It is noteworthy how stagnated most elders get after retiring from their jobs. Excluding necessities, they suddenly find themselves without essential tasks to perform and quickly end up in a monotone and sedentary spiral of unhealthy habits, leading to minor to no cognitive and physical stimulation. Consequently, this set of harmful behaviors may anticipate possible impairments and handicaps, avoided otherwise had they engaged in a different lifestyle years earlier.

Moreover, serious games, a type of virtual entertainment with stimulation and pedagogical roles, would be expected to be more ingrained in the rehabilitation scene by now than they currently are. Inside the video game industry, only a few studies focus on how effective serious games can be when used as an auxiliary tool. Comparing to the whole leisure video games industry, serious games also represent a small percentage of the existing all the existing video games, which is unfortunate considering that they could play a more significant role in modern therapeutic methods, regarding cognitive and physical fields.

1.3 Objectives and contributions

This work pretends to combine healthcare and wellness with technology, taking advantage of the latter to improve and rehabilitate elders' physical and cognitive conditions by creating an entertaining, versatile and practical serious game, promoting a playful virtual experience for the end user.

Any player of any age group should be able to play this serious game. However, it is mainly focused on the elderly group, which have seen their cognitive abilities decline over time, or

that only want to prevent that from happening. The main movements required by the serious game consist on specific actions that need the upper body support, more specifically, arms, wrists, and fingers, aiming to enhance the mobility of the previously mentioned body members. However, this serious game also requires a basic level of upper body competencies on the end-user side to complete the demanded tasks by the interface. This game is not suited for people with advanced and severe low mobility cases since those will need more advanced and specific recovery methods, or programs with the proper specialized assistance. For that, simplicity is key – the serious game should be intuitive, straightforward, well-explained and appealing. It should be faced as a fun and engaging activity from the end user side, and it should be able to replace or help traditional recovery methods and exercises. Overall, this software strives to embody an objective and trackable improvement, in any aspect, to the already existing and used therapeutic strategies, in the particular interest of this work validation.

Since the major purpose of this activity is therapeutic rehabilitation, professional validation should be considered. Implementing the serious game to make it seen as a fun experience is mandatory. However, this does not guarantee its technical and practical efficiency, which ultimately is one of this work main goals: benefiting the cognitive fields while also providing a fun and entertaining experience to the end user – and that can only be achieved when considering the opinion of people who work ”in the field”, in order to validate the techniques required to play it.

Later in this document, all intended implementation ideas will be thoroughly and extensively discussed, breaking down every concept, theory, and reasoning behind the decision making process and potential problems raised, while simultaneously pointing out possible solutions to fix them.

1.4 Document structure

After this introductory chapter, the work background and respective state of the art can be found in **Chapter 2**. Then the definition of serious games is presented, followed by examples supporting how effectively the latter can be to fulfil the role of helping seniors' daily lives. In **Chapter 3**, the project is viewed from a software engineering perspective, by covering an extensive analysis of functional & non-functional requirements, development model, and respective requirement priority. Then, the overall serious game design will be explored in **Chapter 4**. This chapter contains the decision-making process behind its general concept and architecture. The technical design will also be covered with a detailed and comprehensive description of the developed data structures, featuring respective structures and algorithms. **Chapter 5** compares all the contemplated game developing software and explains why Unity was the chosen one. The relevant stages of the software system development will also be described and clarified, such as source code, documentation, graphic user interface, faced problems, and functional tests. After

that, testing procedures with the end users are introduced in **Chapter 6**, providing the reader with an illustrative representation of the obtained results and respective conclusions. To finalize, in **Chapter 7** conclusions are drawn – an overview of what went as planned but also could have been done differently or more efficiently. A future work path suggestion is also included, as an effort to ensure the viability of this piece of work's possible continuation.

This document also includes four annexes in the final pages:

- Annex A - User manual;
- Annex B - Reference manual;
- Annex C - Detailed project documentation;
- Annex D - Testing sessions tables.

2

Background and related work

Contents

This chapter will develop some context regarding the work background and its respective state of the art. To do that, four fundamental concepts will be explained and elaborated:

- Active ageing;
- Cognitive stimulation;
- Cognitive rehabilitation;
- Hand-eye coordination.

Essentially, it is crucial to understand how to trigger those concepts simultaneously to ensure the developed software satisfies and embraces the latter notions. Afterward, the definition of serious games is presented to understand what the developed software backbone should look like before inter-twinning it to the first explained concepts. Following that, scientific experiments can be found in the related work section, supporting how effectively those can be and how much potential they have regarding therapeutic purposes. To sustain the previous premise, the research scope section includes the explored tools regarding the this serious game development, assuring they fulfill their focus on the previously mentioned topics, showing the reader what the envisioned version of the serious game looks like.

2.1 Active ageing, cognitive stimulation, and rehabilitation

As stated in [13] the active ageing concept encompasses a complex notion of multiple fields, which allow the healthy control of the worlds' population. Active ageing guarantees the older population remains healthy and employed for a more extended period, ensuring the mentioned group can still play an active role properly within society. The latter means they should maintain the ability, clarity and dexterity to perform social, political, economic, cultural and sporting activities – which sustain the choice of the word "active", present in the concept name. In turn, the term "ageing" covers the idea of completing the natural life-course cycle. As mentioned in the introductory chapter, a significant life longevity increase was witnessed in the last years, culminating in a scenario where the elder group is the most abundant among all age sections. This situation affects social care, pensions management, employment rates and political decisions.

The definition provided by the World Health Organization (WHO) policy framework [13] agrees with the previous paragraph. According to this organization, active ageing is defined as optimizing opportunities for health, social activities participation, and security to enhance life quality. The previous should be achieved while accompanying people's ageing process.

This definition also explicitly beholds the ratio between life quality and health opportunities. This correlation happens thanks to the support of strategies aiming to promote quality and quantity of life increase, autonomy, and independence enhancement, while reducing costs in the

healthcare system. Elder roles within society is also not disregarded, fostering activities of pedagogical, and practical nature related to employment, politics, education, arts and religion. With that being said, WHO also guarantees protection, dignity, care systems, socials, and finances of the older people unable to secure the rights mentioned above.

Regarding cognitive intervention, it is usual to resort to stimulation and rehabilitation [14], and the latter approaches complement each other. However, they could be done individually, according to the aimed objective and user needs [15].

Cognitive stimulation pretends to foster activities for users to adopt a problem-solving and keen attitude towards a specific end goal [16], as well as trying to exercise their memory – which, combined to active ageing, will be this dissertation main focus. Some examples of these activities are relevant topics discussion, puzzle-solving, listening to music and baking. This approach is associated with keeping the user's social standard functioning levels [17], and it mainly prevents impairments related to neurodegenerative diseases [13] .

Cognitive rehabilitation, on the other hand, is a far more specific treatment. It focuses on restoring a previously damaged brain function. This strategy works towards a particular user's objectives to increase life quality, and it is only possible because its therapy process gradually passes through four essential steps [18]:

- Education;
- Process training;
- Strategy training;
- Functional activities training [19].

2.1.1 Hand-eye coordination

As stated in [20], arm movements performed to achieve a specific premeditated objective, like reaching or touching an object, usually have eye movement associated to it. According to multiple studies [21–23], eyes start to move in the direction of the desired target from around 40 to 100 milliseconds first before the following arm/hand movement. Regarding tasks that require a fast reaction timing, the existing latency between the eye and the arm/hand movement is usually correlated to a situational basis [21], suggesting that recurrent sources trigger the same eye and arm movements with the same delay.

On the one hand, a reduction in reaction time is verified when an experimental temporal gap is added between the central eye fixation and the appearance of a new target. On the other hand, older studies on this eye and arm movement area [24] have evaluated arm movement latencies with different arm reference positions, counting with the influence of neuromuscular delays.

The results were underwhelming, as determining the precise time correlation between eyes and superior members is still somewhat imprecise.

Despite this impreciseness, the realized studies [25] still proved the existence of a mechanism that associates both the ocular and limb motor systems. This conclusion undoubtedly points to the possibility of cognitive development and consequent reaction time improvement through rehabilitation therapies.

2.2 Serious games

Serious games [26, 27] could be defined as virtual games that pretend to take advantage of people's interest in interactive platforms for entertaining purposes. At the same time, those games aim to stimulate the player in any way, either in the pedagogical, physical, or cognitive field – ideally, all of them simultaneously.

The term "serious" is applied to conceptually transmit an idea of a relevant implementation on a significant theme, seeking to go beyond the pure entertainment side of it – usually characteristic of classic video games. Serious games have the purpose of adequately stimulating an intended area of peoples' lives, looking forward to creating a positive impact by improving the well-being or some specific characteristic on those who play it.

Thus, what differentiates serious games from regular video games is solely the purpose behind them [27]. While the first one mainly focuses on the entertainment component, the second one consists of one software able to merge both "serious" and "game" dimensions, ultimately resulting in a helpful and entertaining experience, with a particular end goal already predefined. The process of tackling all the previous plotted quests should result in intentional player development in some specific field, whether cognitive or physical.

Serious games are developed to satisfy the necessities of a particular target audience, and because of that, they should be adapted to the player's reality, conditions, needs and environment. Thus, different strategies should be implemented in the developing process, aiming to guarantee a basic level of adaptability. This strategical aspect intends to add in-game versatility, allowing different players to play it, no matter their capabilities.

Also, creating a certain level of familiarity with the serious game mechanics is crucial. When eventually transitioning to a higher skill level, the player should be able to complete all the requested virtual tasks. If the serious game is effectively developed, tackling these quests should stimulate and enhance the desired pedagogical, physical, and cognitive components, reaching its full enhancing potential.

2.3 Related work on serious games to help elderly

Developed in A&M University, the serious game "Smart Thinker" [28] recruited 59 elders for testing purposes. This game is free, can be played online and it was built to be fun, en-

gaging, and, more importantly, to improve people's cognitive skills. Due to the serious game architecture, attention and memory abilities are the most exercised ones, encouraging the players' cognitive and memory stimulation while playing it. This serious game uses the computer mouse to bridge the end-users thoughts and screen actions. It contains a set of reasonably known sub-games like Rock Paper Scissors, High-Low, and Color Game. According to data previously shown to the player, the latter requires the computer's cursor to select and guess predictable details related to pictures, numbers, words, and color. All participants played all games over 200 times during the testing phase. To accurately determine each participant's progress level, all participants did a Mini-Mental State Examination (MMSE) before and six weeks after. Later, they were also asked to fill an inquiry with questions regarding their own opinion about serious game efficiency. Ultimately, end users' feedback was positive. Project developers analyzed the results and confirmed elders' cognitive capabilities improvement. They admittedly experienced a fun time playing the serious game as a bonus. This information confirms how this serious game became what it was meant to be: engaging, user-friendly and straightforward to understand. Dissecting the results, attention was the most improved skill, followed by memory in second place. These details could mean a step towards the possibility of using serious games as a tool to prevent certain cognitive conditions, with the most popular one being the Alzheimer's disease.

According to [29], the helping tool "Wiihab" is the perfect example to demonstrate how stimulative serious games can potentially be. The latter takes advantage of Wii Fit, a functional console video game that creates an interactive virtual reality environment through the player's physical movement. This console has familiar access and its easily purchased due to its worldwide distribution. This game revolves around virtual quest tackling, which is only possible when moving the body directly to an existing paired motion sensor. Functioning with remote control allows performing physical exercises while also influencing the video game virtual environment pose through the data being sent to the receiver based on the controller motion. In this experience, Wii Fit is used under the nickname of "WiiHab", a wordplay between "Nintendo Wii" and the word "rehabilitation", aiming to increase physical activity and speed up the rehabilitation process of the disabled users. St. Mary's Medical Center of the United States pioneered this program in 2008, applying it to approximately 100 patients suffering from multiple diseases and disorders [29]. From the results, it is possible to conclude how the fitness avatar from "WiiHab" positively impacted the physical activity of disabled people with spinal cord injury. For this study's sake, the real-time avatar interaction was used to project a patient self-representation to a virtual reality scenario. This fitness avatar represents a massive advantage for people going through rehabilitation processes. Due to its immersive environment experience, the players can purposely deceive their brains, sensing the enjoyment of performing physical activities. However, they are only using their Wii remote control within the limited infrared

range to send data to the motion sensor without the concern about any potential injury. This study proved how therapeutic methods could include ”WiiHab” in an adequate physical activity plan to treat spinal cord injury disabled people, inside a rehabilitation environment. It [29] provided essential and interesting guidelines for future serious games design, ensuring health, and promoting fitness tracking before and after physical activity. Finally, it should support the regular performance of physical activities to promote multiple training variants amongst disabled people with spinal cord injury. The fitness avatar model should be implemented to contemplate various performing phases, provide suited feedback to each one, and minutely analyze this method's results. This last step is essential regarding the self-motivation chapter. Once the user identifies the model's positive impact on his progress, it will likely be a motivator factor to continue working under the fitness avatar model.

The next and last example [30] was created through the popular XBOX series add-on, Kinect. Once again, it also proved to speed up the rehabilitation process of disabled people. The article showcases how a physical rehabilitation system named ”Kinerehab”, once again, a wordplay combining Microsoft ”Kinect” and ”rehabilitation”, promoted the possibility of stimulating the lower body on two young adults with impairments by using the previously mentioned system. It was implemented in a public school setting, and after eleven experimental sessions, the two users were delighted with the results. Given the context, the number of correct movements achieved with the rehabilitation process was significantly high. As a bonus, this recovery concept of playing video games seemed effortless for the end users. Tracked data confirmed how the two participants significantly improved their physical performances over time. They also greatly increased their motivation and willingness to engage in physical rehabilitation sessions. After the predefined experiment phases, they manifested interest in maintaining this kind of therapy after the testing phase was over. Since this rehabilitation system was tested by two users only, more research needs to be done to certificate its validation. However, it proved how effective it could be. It also highlighted how this idea could minimize specialized staff intervention during rehabilitation. This example takes advantage of Kinect system technology, which features voice control settings and infrared sensors capable of capturing body movement precisely.

In [3, 31], an intuitive comparison between Leap Motion, Kinect, and Wii resulted in Table 2.1. However, it should be noted that the Wii Motion-sensor receives arm movement real-time data from the Wii remote motion, unlike Kinect and Leap Motion, which receive it from body movements. For that reason, ”Image refresh rate” row does not contain any value.

Table 2.1 Kinect, Leap Motion and Wii comparison table. Reproduced from [3–7]

	Kinect	Leap Motion	Wii
Manufacturer	Microsoft	Leap Motion Inc	Nintendo
Technology	1 infrared transmitter 1 infrared camera of 0.3 megapixels 1 RGB camera 4 directional microphones	3 infrared transmitters 2 infrared cameras of 1.3 megapixels	1 motion sensor 1 infrared pointer
Image refresh rate	30Hz	200Hz	/
Recognition	Body movements Facial recognition Voice recognition	Hand movements Finger movements	Arm movements
Precision	In centimeters	Hundredth of a millimeter	In centimeters
Field of vision	Horizontal 57º Vertical 43º	Anteroposterior 120º	Almost 180º vertically and horizontally
Captor's range	1.2 - 3.5m	0.10 - 0.80m	0 - 5m
Work space's floor surface	6m	1.16m	9m
Configuration	SDK for Windows by Microsoft	AirSpace Home	Not configurable on PC

Based on the compared data in Table 2.1, LeapMotion controller could be a viable option. Since it recognizes hand and finger movements, multiple concepts can be turned into serious games with the objective of rehabilitating the end user upper body and stimulating its cognitive domain. Also, its 200Hz refresh rate can promote a smooth and immersive virtual reality experience. Its captor's range also allows to implement a wide variety of required movements, as well as its small work space's floor surface portability to the projected software system.

2.4 Research scope

This work strives to achieve cognitive stimulation and rehabilitation in the elderly group, relying on hand-eye coordination inside a virtual scenario to do so. This software was developed under Unity [32], a cross-platform game engine able to create three-dimensional (3D) games to reach the latter. To build a bridge between the previous software and real-world movements, Leap Motion [6,7], a computer hardware sensor device that supports hand and finger motions as input, similar to a mouse, but requires does not require hand contact or touching. This hardware was chosen due to its portability, high frequency image refresh rate and versatility to be featured in diverse software implementations, which could be to enhance hand-eye coordination while also promoting rehabilitation.

Learning about them was crucial to planning a flawless serious game to develop the project with the tools mentioned above. Since this software must be attractive and appealing to an older age group that generally has little to no technology sensibility, it should logically cover focal points such as interlacing cognitive stimulation and rehabilitation, promoting and hand-eye coordination with the previous concept and how to acquire therapeutic validation.

It is also fundamental to understand the average user rehabilitation time window. Comprehending this type of data allows an accurate estimation of when the first progress became visible. This data is relevant to confirm the serious game is having the desired impact on the end-user cognitive dimensions.

The following bar chart depicted in Figure 2.1 was built with data provided in [2] and according to it, the first 3 month rehabilitation stage is where the standard healing aches and pains phase takes place.

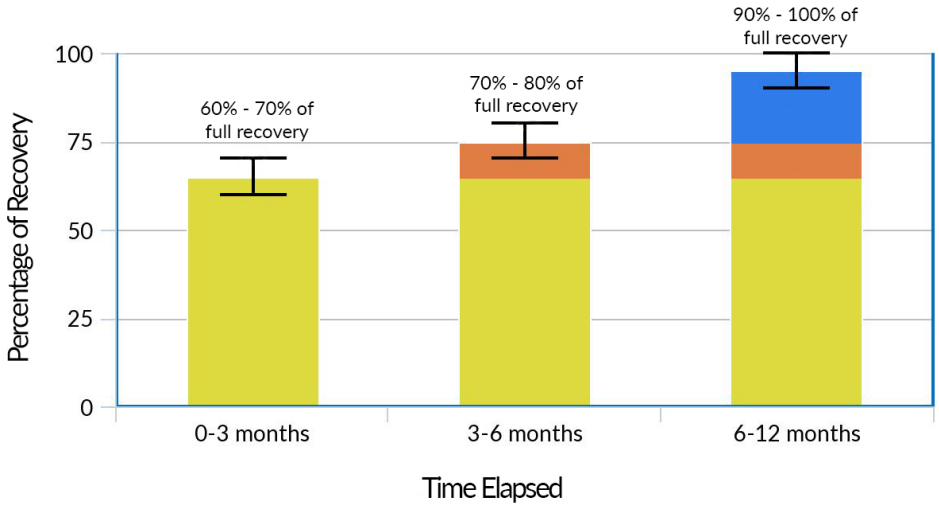


Figure 2.1: Rehabilitation progress over the course of a year. Reproduced from [2].

2.4.1 Key requirements

When it comes to these specific serious game key requirements, an immediately eye-catching factor is professional validation. Since the primary goal of this work focuses on therapeutic rehabilitation, the serious game implementation should have the previously mentioned requirement in absolute consideration. Its technical and practical efficiency should be indisputable, allowing all the cognitive fields to benefit. This registered progress should be approved and confirmed by a professional who directly contacts a source of data or subject of interest to validate the techniques required to play it.

Implementing the serious game to make it a fun and entertaining experience is also a mandatory point. The gameplay experience should resemble an upgrade to standard therapeutic techniques, excelling for its pleasing involvement in the process. In fact, there is no other way of turning the act of playing serious games for rehabilitation purposes a habit.

Finally, it is essential to have a genuinely user-friendly interface to engage older people. For that, simplicity is critical: an inexperienced player should be able to comprehend and absorb its physics and mechanics quickly. Consequently, this will appeal and require a massive intuition level from the software end. This point is fundamental because unnecessary barriers in the playing and learning processes may discourage older people from using it. To better comprehend the serious game, a user manual is bundled in the final pages of this project. It was written to try to reduce the player's wasted time overcoming installation difficulties and other misconceptions – fast-forwarding the player way into the serious game playing phase is a priority.

2.4.2 Envisaged serious game design

Regarding the envisaged serious game design, when starting the game, the end-user should be faced with an initial menu composed by four different options, represented in a considerable sized font: Play, Scoreboard, Options, and Quit (see Figure 2.2).

Figure 2.2: Overall view of the main menu screen

This serious game will be composed of 5 different sub-games.

In the first sub-game, the user should grab different shaped objects and drop them in properly labeled containers. It aims to promote arm transversal movement, finger dexterity, geometrical recognition, and logical reasoning.

Regarding the second one, different colorway spheres will be presented on the screen. The player should use the virtual hands to hit those with a specific color displayed on the screen. This color will change from time to time and each successful hit will add points to the player score, stimulating arm movement and color recognition.

The third one consists of a ”bowling like” mini-game, where the player has multiple spheres at its disposal and has to throw them to take down a set of cylinders. This sub-game intends to develop the user’s sense of depth, wrist strength, and finger sensibility.

The fourth sub-game will require the player to use the virtual hand to answer multiple-choice general knowledge questions. It aims to promote the player’s memory and reasoning.

Finally, in the fifth game, the player has to steer rolling balls into the proper compartment. This sub-game has the objective of promoting transversal arm movements and stimulating reasoning to make the right decision.

Given that this serious game is designed for older people, it is also relevant to notice that in-game objects should also be considerably sized. The camera perspective should be situated

in a virtual place close enough for the objective to be as intuitive and simplified as possible.

Also, the scenario and objects' color palette should be chosen to prevent unnecessary misunderstandings, allowing the clear perception of every element in each sub-game.

The following Figure 2.3 depicts the intended serious game virtual environment.

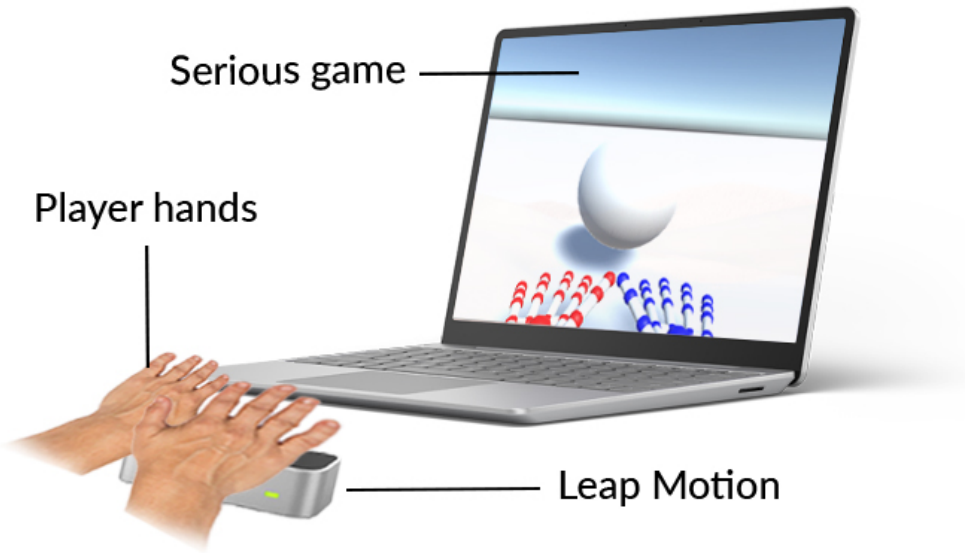


Figure 2.3: Overall view of the serious game virtual environment

2.4.2.A LeapMotion controller

The LeapMotion controller [6, 7] is a small USB peripheral device that is designed to be placed on a physical desktop or table. It uses monochromatic IR cameras and infrared LEDs and can capture gestures within 1 meter of distance. Its cameras generate almost 200 frames per second of captured data, which are sent through a USB cable to the computer. Then, the latter creates virtual 3D position data with the controller's installed software by comparing all the 2D frames generated by the two included cameras. The overall controller average accuracy is approximately the hundredth of a millimeter, which will enable the required human-machine interaction level to play the idealized game smoothly.

As a consequence of LeapMotion controller use, only hands, finger, and wrist movements will be registered by the software. Even though arms movement is not tracked by it, they play an essential role for the player to succeed at the serious game's objectives as they allow the player to control the virtual hands' depth. On the one hand, this could be seen as an advantage. Since this serious game only requires upper members' movement, understanding the game's mechanics and movements becomes more accessible, promoting simplicity. On the other hand, this implication could also represent a disadvantage – it does limit the number of possible physical

activities, which will condition and restrict the implementation options regarding rehabilitation exercises involving other body parts than upper limbs. Also, the complexity of setting the gameplay environment up is lowered due to the low quantity of technological equipment needed. The required devices are a general purpose computer, the LeapMotion controller and the serious game software.

Figure 2.4: Leap Motion hardware

2.5 Summary

This chapter addressed the concepts of active ageing, cognitive stimulation, and cognitive rehabilitation. The term serious game was also covered in the interest of this work validation, for a solid workflow to build a complete software contemplating all the latter concepts. As firmly believed in Chapter 1, serious games have high potential to positively impact society, which was proved to be scientifically valid in the related work section of this chapter. In the research scope section, the key requirements are set, and the serious game design is sketched. After that, the software/hardware were chosen based on their potential to fulfill the project goals. Since this serious game needs to be implemented without previous knowledge, Unity was selected due to its steep learning curve. This chapter also explained the bridge between the previously analyzed concepts and the serious game software – this connection allows data transfer between real-life movements and the created virtual 3D pair of hands. The next chapter will elaborate on the serious game requirement analysis, thus going beyond the “key requirements” sub-section.

This page intentionally left blank.

3

Serious game requirements analysis

Contents

As stated in Chapter 2, cognitive stimulation, rehabilitation, and hand-eye coordination can be exercised in multiple ways. The previous concepts rely on activities that put the player out of his comfort zone, regarding physical and mental domains. That chapter also showcased how consistently performing similar activities can result in the desired progress, achieved by abusing the latter repetition process. To successfully fulfill its purpose, this serious game should also be fun and engaging. Chapter 2 also states how LeapMotion controller is used to create a bridge between reality and the serious game virtual environment. This strategy obligatory entails some critical details that must be stressed when plotting the requirements addressed in this following chapter.

With this in mind, Chapter 3 will scrutinize every detail related to the serious game requirements, ensuring all the preparation is done to promote the desired repetition with the available tools. To achieve the latter, the present chapter will essentially display the thought process behind all requirements' appointments while contemplating possible constraints and limitations. In the functional requirements section, this concept will be explained, the requirements will be elicited and the respective use cases will be displayed with the help of UML diagrams. This will also define non-functional requirements, followed by a description of all requirements belonging to that group. The following section will present design constraints and limitations and a brief insight into the decision-making process that led to their identification. The development model section will explain the chosen development model, explaining why it is the most suitable approach for the intended software. Requirements will be ranked according to their priority in the requirement prioritization section, and additional notes will be provided. To finalize the chapter, conclusions about the chosen requirements are drawn in the summary section.

3.1 Functional requirements

According to [33], the functional requirements describe the set of services that ultimately result in experience the software can provide to the end user. Those requirements include any functionality ensuring the system can execute every featured initially established. These implemented features' utility can be envisioned in the projected use cases. Overall, the latter ensure full software functionality by guaranteeing the system can fulfill all the expected end user's needs.

In this serious game development process, the following functional requirements were established:

- Once the software system initializes, the end user should be faced with the initial menu;
- The end user should be able to select any of the four displayed options in the initial menu, with them being Play, Scoreboard, Options and Quit;

- When the end user selects the ”Play” option, he should be forwarded to the first sub-game warm-up scene;
- The end user’s latest score should be visible when selecting the ”Scoreboard” button;
- At the main menu, selecting the ”Options” button should take the end user to a page where the game volume could be changed;
- ”Quit” button should immediately exit the game and close the software application;
- Every sub-game instruction should be visible on top of the screen;
- Whenever the end user completes the requested task in the warm-up scene, he should be forwarded to the respective sub-game scene;
- End user’s total score should be visible on the right side of the screen;
- A score increase should occur each time the end user successfully completes the sub-game requested task and decreased each time the end user performs it wrongly;
- The end user should be able to quit the game at any given moment;
- The end user’s final score should be displayed as a final scene, right after the last sub-game;
- The end user should be able to save his score at the end of the game.

3.1.1 UML use case diagrams

As stated in [34], Unified Modeling Language (UML) is a modeling language represented through a diagram cluster. This illustration aims to facilitate the requirement specification role and to visualize and document the specified goals. Since a software system is being built, the previous language will be used to create multiple use cases, forming a use case model [33]. That methodology previews the software’s expected functionalities and reactions within the system environment. This model can have different interveners, known as ”actors”, to simulate the end users usability process. Overall, this UML use case diagram is a simplified way of illustrating which features the end-user can explore and the expected system response to each actor action.

To easily comprehend these multiple use cases, the following UML use case diagrams were developed:

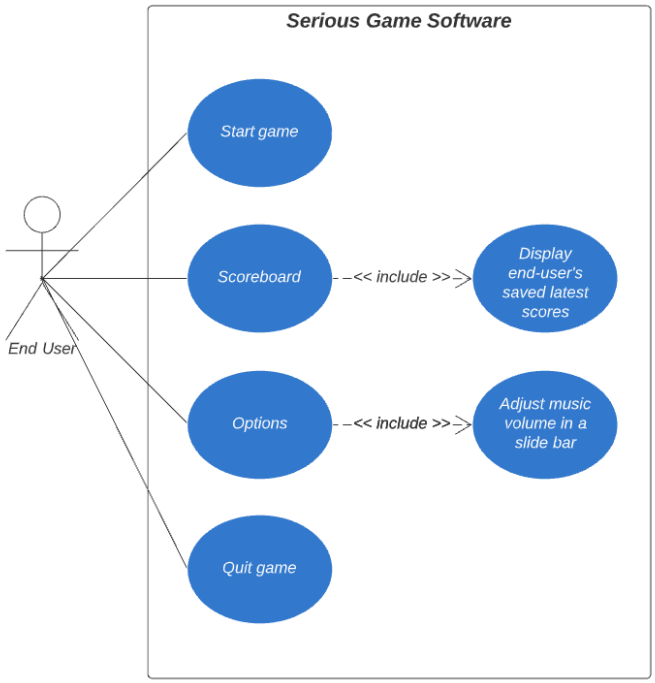


Figure 3.1: Main menu use case

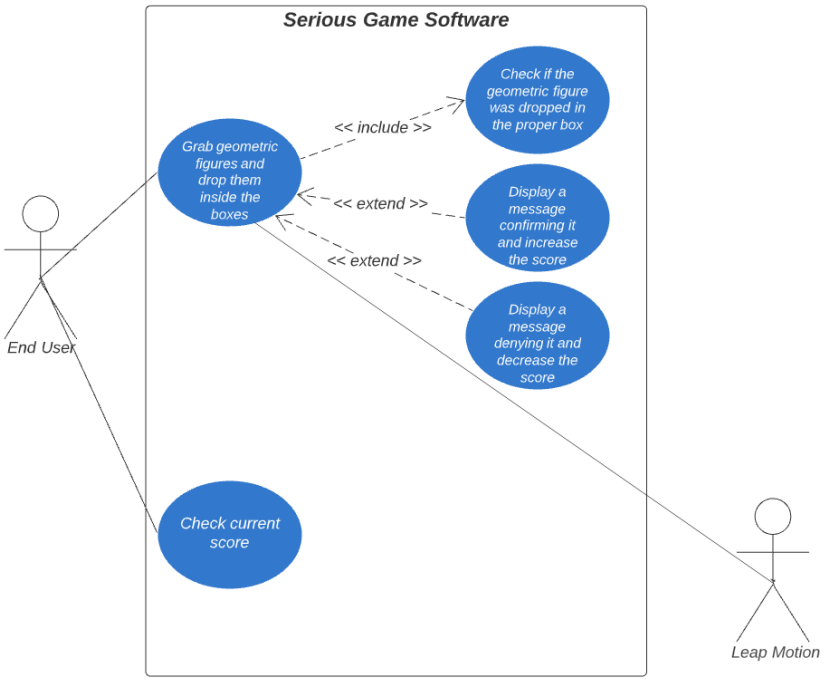


Figure 3.2: Sub-game 1 use case

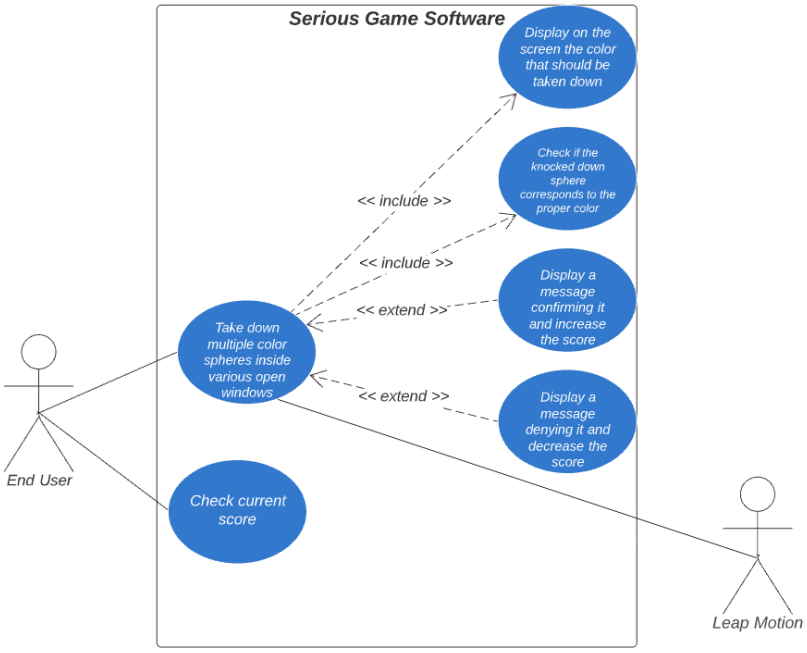


Figure 3.3: Sub-game 2 use case

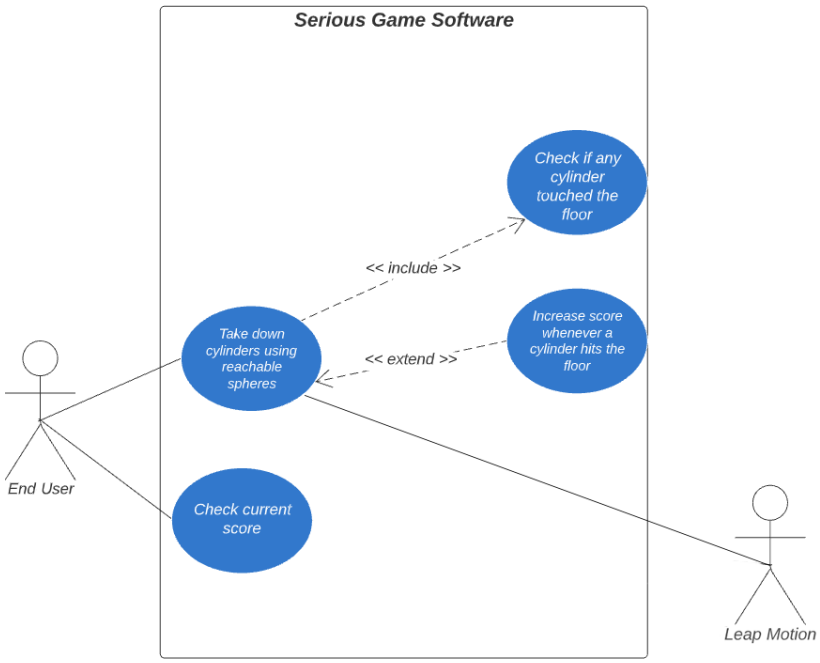


Figure 3.4: Sub-game 3 use case

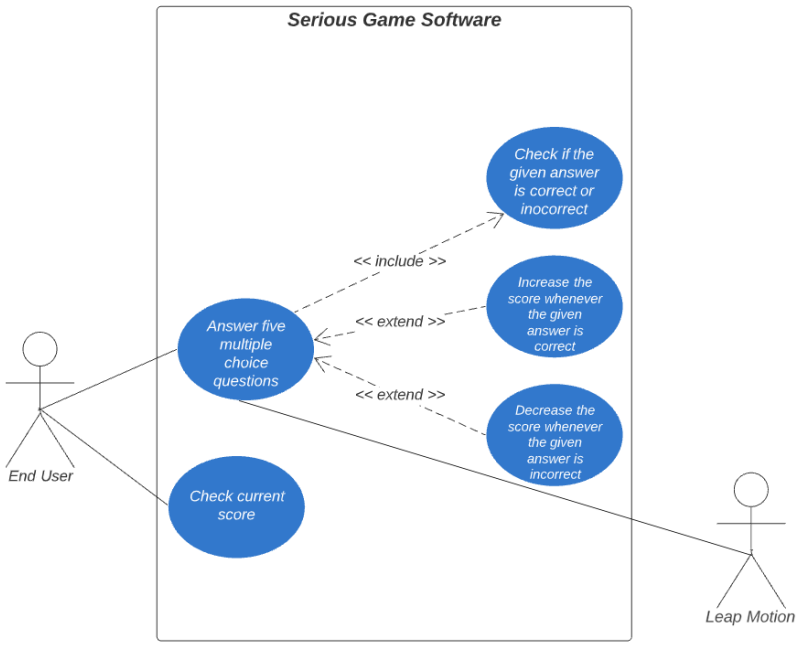


Figure 3.5: Sub-game 4 use case

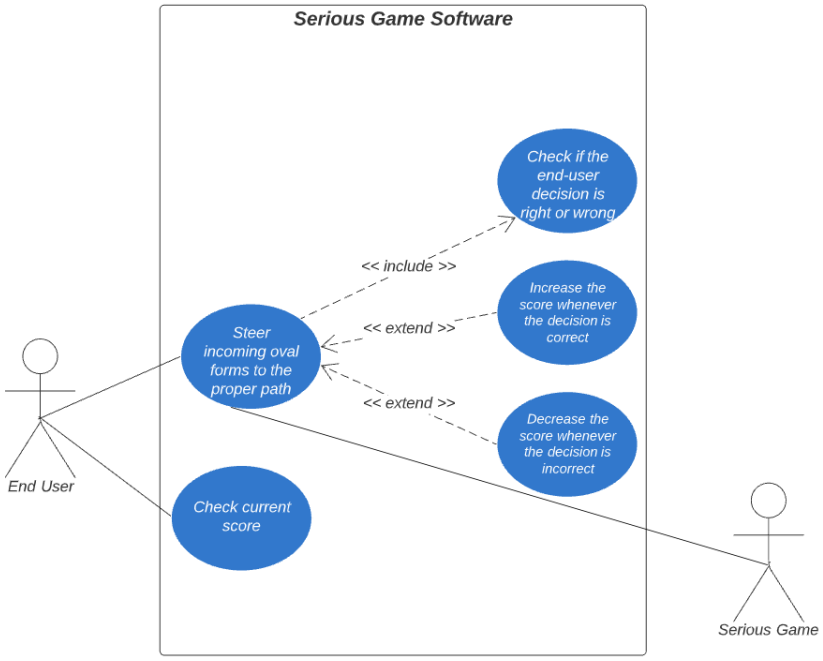


Figure 3.6: Sub-game 5 use case

3.2 Non-functional requirements

As stated in [35], non-functional requirements evaluate the quality attributes of the software system as a whole. Those requirements guarantee the system being analyzed can function under

certain limiting conditions. The non-functional requirements can be seen as product properties and represent software quality attributes. In the interest of this project validation, the most essential quality attribute is undoubtedly usability. To ensure the software possesses the latter characteristic, a sequence of functional and performance tests will be considered, allowing the end user to try out a prototype of the serious game, as well as to provide its feedback. According to the previous feedback, some implementation adjustments can be made, to guarantee the highest possible usability level for this software's target audience.

This specific project contains the following non-functional requirements:

- Simplified options and menus;
- Intuitive graphic user interface;
- Clear in-game instructions through warm-up scenes, before loading any of the five sub-games;
- Possibility of using the mouse to navigate through the serious game menus and also to quit the application at any given moment;
- Engaging and immersive virtual environment gameplay;
- In-game goals directly correlate to and validate cognitive stimulation, rehabilitation and eye-hand coordination.

3.3 Design constraints and limitations

According to [36], the design constraints concept encompasses a set of required conditions that guarantee project success. By eliminating the possibility of taking certain decisions right from the beginning, the project is tapered to follow the best possible path, suiting the end user's exact necessities and demands, while also avoiding time-wasting on unnecessary features and resources. In the developed software, the following design constraints were adopted:

- As Unity was the chosen game engine to develop the intended serious game, the latter must be developed in C#;
- The developed serious game should be adapted to elder people struggles, since they represent the software target audience;
- The menus and additional instructions should be implemented in Portuguese, to reach the intended audience of end-users;

- Since the LeapMotion controller was the chosen sensor to bridge real-life movements with the respective virtual reality actions, this software should be executed in a computer with Windows OS, guaranteeing both Unity and LeapMotion controller software support;

- Considering LeapMotion controller tends to sporadically take a few seconds to calibrate the end user's hand position properly, a warm-up that quickly leads to the sub-game scenes must be introduced. The latter will allow less time wasted with calibration issues.

3.4 Development model

Since this project required professional validation while also keeping the fun and entertainment values, an objective approach needed to be taken. Therefore, it should encompass a strategy able to accommodate overtime changes to satisfy the end user's requirements and demands. Due to that, the Agile Project Management approach was chosen. According to [37], this methodology focuses on splitting heavy projects into multiple iterations to quickly achieve fundamental tasks while simultaneously supplying the "client" with usable software prototypes. The latter method introduces flexibility, allowing the end user to test the software before the final version, opening the opportunity to implement any desired change.

It is also important to mention the "client" role was played by representative from Cáritas Diocesana de Coimbra [1], whose help and feedback were fundamental. Given this serious game target audience is constituted by senior users, the client experience with that age group was crucial for complementing this project development phase.

This approach holds four main values:

1. Individuals and interactions over processes and tools – during the development of this serious game, client feedback was always very important and appreciated;

2. Working software over comprehensive documentation – only a small amount of documentation was initially created, containing the requirements and design constraints. Then a basic system was built and continuously developed for the client to test it, so changes in the features could be made;

3. Customer collaboration over contract negotiation – client opinion always had priority over the previously created documentation, which ended up implying plenty of changes to the latter;

4. Responding to change over following a plan – client feedback obtained between iterations was crucial to change the serious game.

[1] Click here for Cáritas Diocesana de Coimbra website

Overall, this project hugely relied on the Agile Project Management approach since each iteration provided a new version, including a new feature.

3.5 Requirements prioritization

The requirement prioritization concept encompasses the requirement ranking process, ordering them from the most to the least significant software functionality. This labeling system helps to ensure fundamental features get implemented adequately and in time. The previous idea prevents time-wasting while also promoting a heavy focus on task completion – leaving less important features unimplemented, if needed. Those could still be added later, if the time required in their implementation process does not compromise the final software version deadline.

This section demonstrates the importance of the highest priority ranked requirements, and how influential they can be regarding its functionality. The MoSCoW method [38] was the chosen scale to perform this ranking – this approach consists on distinguishing the fundamental from the dispensable features. The chosen name for this method is an acronym that encompasses the initial letter of the following terms:

- ”Must have (M)” refers to a mandatory feature to include in the software. Not including any functionality with the previous label automatically flags the application as a failure. In this project, an example of this could be not including a critical virtual object that would prevent the end-user from completing the requested task.
- ”Should have (S)” covers good but not strictly necessary features. The particular requirement aimed to be satisfied most likely is already fulfilled by one of the requirements labeled as ”Must have”. Even though it could represent an excellent addition to promote software diversity, it is not crucial enough to compromise its success.
- ”Could have (C)” encompasses a recommendable feature that could improve the end user experience when using the system, but it is nowhere close to being necessary.
- ”Will not have (W)” refers to an idea that came to be equated but will not make the envisioned plans for a reasonable reason. Thus, any requirement labeled as the latter is excluded from the implementation process.

Overall, using the MoSCoW method as a scale comparison term seems to perfectly fit this project’s needs, which led to the assembly of the Table 3.1:

Table 3.1: Requirement prioritization following the MoSCoW method.

Requirement	Description	Priority
1	Graphic User Interface	M
1.1	Create initial menu	M
1.1.1	Create sub-menus buttons to initialize the game, check the latest obtained scores, adjust end user's preferences and close the game	M
1.1.2	Implement and develop the latter sub-menus	M
1.1.2.1	Allow the end user to register end user's latest scores	S
1.1.2.2	Allow the end user to change the music volume	S
1.1.2.3	Allow the end user to change the font size	C
1.1.2.4	Include an in-game menus translation from Portuguese to English	C
1.2	Creation of a graphic matrix for the to be developed 5 sub-games	M
1.3	Add background music	S
2	Implementation of each 5 sub-games	M
2.1	All sub-games include a temporal limit	M
2.2	All sub-games include a score indicator transverse across them	M
2.3	First sub-game grass scenario implementation	M
2.3.1	Add five boxes with different labels	M
2.3.2	Add multiple geometric forms easily reachable for the end user	M
2.3.3	Create a fence around the previous geometric forms to avoid them from going out of the end user's reach	S
2.3.4	Each geometric form can be dropped in every single box	M
2.3.5	The end user's score is increased if the geometric form was dropped in the proper box	M
2.3.6	The end user's score is decreased if the geometric was dropped in the wrong box	M
2.3.7	A message is displayed if the end user dropped the geometric for in the proper box informing the latter the right action was performed	S
2.3.8	A message is displayed if the end user dropped the geometric form in the wrong box, informing the latter the right action was performed	S
2.3.9	Brand new geometric forms with a different color spawn from time to time	S
2.4	Second sub-game implementation	M
2.4.1	Add a set of multiple color spheres inside open windows	M
2.4.2	Display text on the screen indicating the color of the sphere that should be taken down	M
2.4.3	Increase end user's score if the proper color was taken down	M

Continued on next page

Table 3.1 – continued from previous page

Requirement	Description	Priority
2.4.4	Decrease the end user's score if the wrong color was taken down	M
2.4.5	Text showing if the decision was right or wrong	C
2.5	Third sub-game implementation	M
2.5.1	Add multiple easily knock-able down cylinders	M
2.5.2	Add three spheres that the end-user could easily reach and grab	M
2.5.3	Allow spheres to spawn from time to time	S
2.5.4	Add a bowling alley	S
2.5.5	End user's score increases when a cylinder is hit	M
2.5.6	Any sphere disappears when hitting any cylinder	S
2.5.7	Any cylinder disappear once it hits the floor	S
2.5.8	A fence is build around the area where the spheres are located	S
2.6	Fourth game implementation	M
2.6.1	Add a text box containing a single question	M
2.6.2	Add two text boxes containing two possible answers for the previous question	M
2.6.3	If the end user chooses the right answer, the score should be increased	M
2.6.4	If the end user chooses the wrong answer, the score should be decreased	M
2.6.5	Display a message informing the end user if the decision taken was right or wrong	S
2.7	Fifth game implementation	M
2.7.1	Add an alley with diverse path changers along its area	M
2.7.2	Add different two color oval shaped forms that will roll on the previous mentioned alley	M
2.7.3	Add two different possible destinations for the end-user to forward the oval shaped forms to the one that matches the oval shaped form color	M
2.7.4	Add spawn of oval shaped forms to promote in-game dynamism	S
2.7.5	If the end user chooses the right answer, the score should be increased	M
2.7.6	If the end user chooses the wrong answer, the score should be decreased	M
2.7.7	Display a message informing the end user if the decision made was right or wrong	S
2.8	Final game scenario	M
2.8.1	Play again button	M
2.8.2	Save stats button	S

3.6 Summary

This chapter started by addressing the serious game functional requirements, which allowed a better understanding of the indispensable software necessities that needed to be fulfilled due to minutely description of the latter. UML use case diagrams were beneficial to visualize these needs better, displaying the functionalities the main menu and the remaining five sub-games should have intuitively. Following that, non-functional requirements were also analyzed. The software properties were highlighted with those, guaranteeing the main quality attributes are satisfied, thereby certifying the software value. In the next section, design constraints and limitations were covered, which narrowed the developing stage from the beginning, leaving less relevant decisions already taken, consequently promoting a better focus towards the implementation phase. Consecutively, the development model was scrutinized to explain why the agile approach method was the chosen strategy to carry out this project. This chapter also explained its application in this specific project and identified it. Lastly, the requirements prioritization under the MoSCoW examined was analyzed. This section described the latter method and helped the development phase by ranking the requirements priority, according to these a scale provided by the previous form.

The already mentioned serious game design will be described in detail during the next chapter, focusing on explaining the thought process responsible for the final design.

4

Serious game design

Contents

The previous chapter covered all relevant information about system requirements, including a brief but pertinent mention of the initial design constraints. These enabled to limit projecting options initially, fostering a clearer vision of the different possibilities regarding this serious game design. Chapter 3 also demonstrated the importance of transitioning from the requirement gathering phase to the design stage – the latter have a fundamental role towards software aesthetic since the serious game design must match the requirements' demands.

This chapter will demonstrate how the serious game design can be projected around the previously mentioned requirements by providing a general idea of the system design concept. After comprehending that, the system architecture will be covered by displaying how it portrays the different system components coexisting. The preliminary built UML class diagrams will be used as an explaining tool to better understand that concept. It is essential to envision the set of projected foundations for the intended sub-games and their anticipated appearance. For this, the design mock-ups will be explained and depicted – UML sequence diagrams of and UML activity diagrams were the used charts to demonstrate how these mock-ups may include the planned event interaction between graphic design and code. After that, technical design will be defined, showcasing its importance while also explaining its influence in the idealized development concept.

4.1 General concepts of software design

As stated in [39], software design consists in the previewing process responsible for a significant part of the software appearance in its finished version. This concept also encompasses solution anticipation to eventual hazards and problems. These goals include low-level components and algorithm projection – foreseeing [39] the expected system behavior regarding human interactions and its attached hardware. The software design process embraces a group of sequential phases, promoting the conceptualization and shifting of the software requirements to the software implementation, i.e. the specification of a solution to the problem defined in the requirements analysis phase. This methodology opens space for a better understanding from the developing perspective, fostering an absolute comprehension of the intended goals and mandatory software building constraints. Ideally, the software design [40] should be adequately versatile to consume the available resources exclusively. It also must be accurate to the analysis model and preferably build off existing field-tested design patterns for time management and viability reasons. In this designing stage, source code is frequently ruled out, with pseudo code [41], an informal and less effort-requiring language, being preferred. The latter concept allows pseudo coders to provide a descriptive view of the intended software, describing the intended structure, and previewing necessary algorithms. Pseudo code could also be extremely helpful for projects where the developing team have to write source code in a language that they are not familiar with, fast-forwarding the implementation process. Overall, software

design encourages designers to create visual solutions able to solve eventual problems while respecting the established initial requirements – the best possible design leads to ideal solutions regarding project goals.

Software design is frequently represented through design modeling languages [42] to portray the intended concepts and details. Different variations of UML charts will be used to achieve accurate chart representations. Documenting software design through the previous strategy ensures all stated specifications, requirements and features are met just before the coding phase begins. It also turns the latter stage more straightforward and objective regarding achieving the intended goals.

As previously mentioned, this serious game must be fun, engaging, and, most importantly, therapeutic. Therefore, all in-game objectives must be sketched to anticipate and selectively determine the required movements to achieve success within the projected application. With this in mind, software design is crucial to set the foundations that the implementation phase should replicate from the available documentation.

4.2 Architecture

As stated in [43], the software architecture promotes the system overview of its organization and structure, while explaining how the projected software development components behave with each other. Therefore, when exploring the latter process elaboration, a decision-making process contemplating every decision's pros and cons must occur. It is essential to understand that each option may affect system attributes like performance, maintainability and quality. Hence, with these possible risks in mind, it is frequent to plot multiple architectures styles for software systems. These sketches usually overlap in any regard, so a combination of them is generally considered – ultimately resulting in the final software architecture archetype.

Compared to the software design concept, which mainly describes data structures and algorithms, the software architecture concept distinguishes itself for providing an overall system structure sketch, accompanied by explanations about the component interaction and behavior with each other. However, the previous two ideas also have commonalities, which lead to a situational application when considering which one to use for software mounting – ultimately, the system architecture ends up being a design, but not all designs represent an architecture. The chosen concept varies from case to case, but usually it is the one that provides a straightforward solution to the existing issue. A usual tool for this consists of process analysis, where the problem in question is decomposed in multiple sub-problems, and combining all of these sub-problems' resolutions, the ideal solution is obtained.

Ideally, the design phase should gradually be updated and improved as the implementation process advances. Newer proposed solutions for the implementation process may provoke design changes, which constrains the depicted architecture to accommodate those changes without

compromising the conceived software system outcome.

Software-related issues are often associated with implementation bugs and initial design planning flaws. Consequently, architecture analysis can be a powerful tool to solve them, since most of the time, reviewing software code reveals to be insufficient. Significant problems are frequently related to the system's fundamental components malfunctions, attributing architecture design the required leverage to analyze and solve those issues. In other words, it can identify what is incorrect with these components and troubleshoot them towards the desired problem-solving. This procedure promotes environment comprehension towards the surroundings where the projected system will operate. Other essential procedures revolving around different architectural activities [44] are also:

- Architectural synthesis – the act of creating and improving the software architecture, based on requirements analysis and design documentation;

- Architectural evaluation – consists of two stages, with the first one being validation. This stage contemplates sketched design validation by evaluating how well the latter can meet the requirements initially provided, and may occur whenever a design decision is being considered or whenever a design stage got completed. The second stage of this concept incorporates validation and verification, submitting the built system through a process to determine whether or not the architecture meets its specifications;

- Architectural evolution – the existing architecture is gradually evolved to accommodate requirement or environmental changes. This evolution should happen whenever a new feature needs to be implemented while never disregarding the preserve of the remaining system's functionalities behavior.

This serious game's main classes architecture was preliminarily represented in a UML class diagram. The following subsection will minutely explain how the previous diagramming technique accurately portrays and exemplifies the intended system architecture.

4.2.1 Preliminary UML class diagram

UML class diagrams aim to illustrate main software classes, including their respective names and attributes, containing required data values to accurately conceptualize the latter classes into the desired code. These diagrams also provide the previewed methods, consisting of functions able to alter the class attributes or interact with other classes. Depending on its visibility, defined by the indicative behind the method name, those can be private ("-"), public ("+") or protected ("#").

Regarding the system preliminary software architecture, its illustration in a UML class diagram allows overviewing the system as a whole, promoting a more straightforward implementation process.

The overall system is depicted in Figure 4.1:

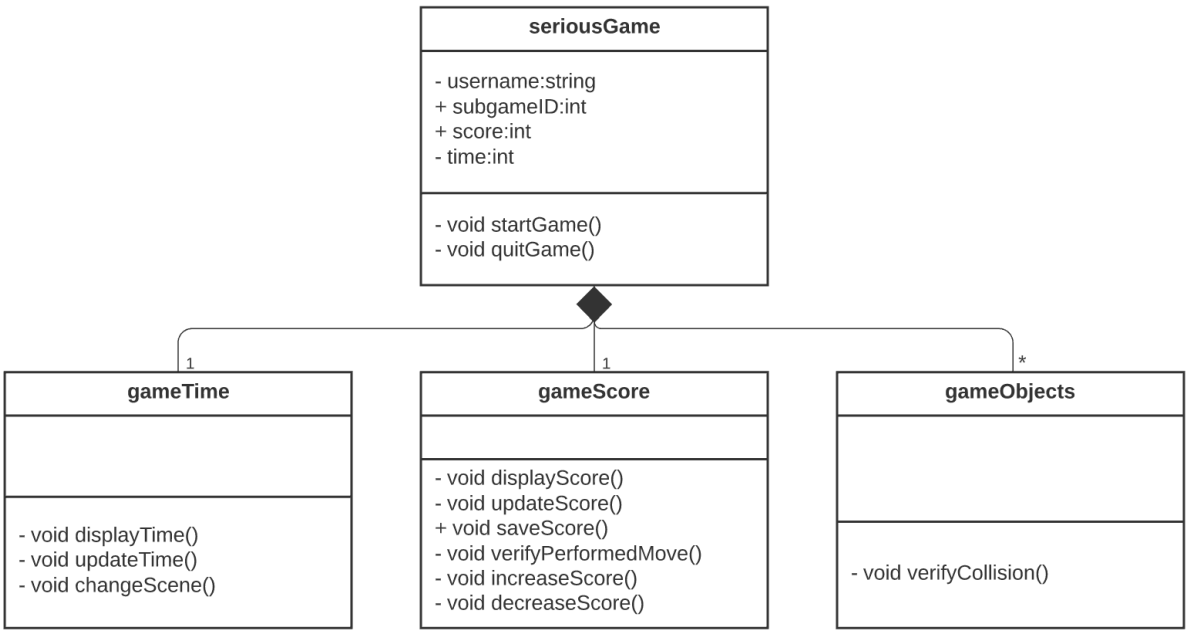


Figure 4.1: Preliminary UML class diagram

This preliminary diagram represents a main class for the serious game and one for objects, score and time control. As an example, in this diagram it can be stated that one serious game has a single time and score, but can have multiple objects. These three classes are dependent from the serious game class since they would not exist without the ”startGame()” method. The main class is also responsible for storing all the relevant variables.

4.3 Mock-ups for the design of the graphical user interface

In graphic design, mock-ups [45] represent a prototype structure model of a particular concept or object intended to be turned into a concrete and tangible design. The latter can be used for studying, testing, or feature showcasing purposes, as they can set the foundations for an evolved and more complex version of the intended idea. Mock-ups are characterized to illustrate an early draft, previewing a future project while also easily accommodating needed changes.

In this serious game, the mock-ups of graphical design aim to portray an objective yet introductory preview of the menus and the sub-scenes environment. Regarding this specific software, mock-ups were designed to properly envision each sub-game required set of movements

to tackle the existing quests. To efficiently develop this software's graphic user interface, mock-up designs depicted in Figures 4.2 were sketched:

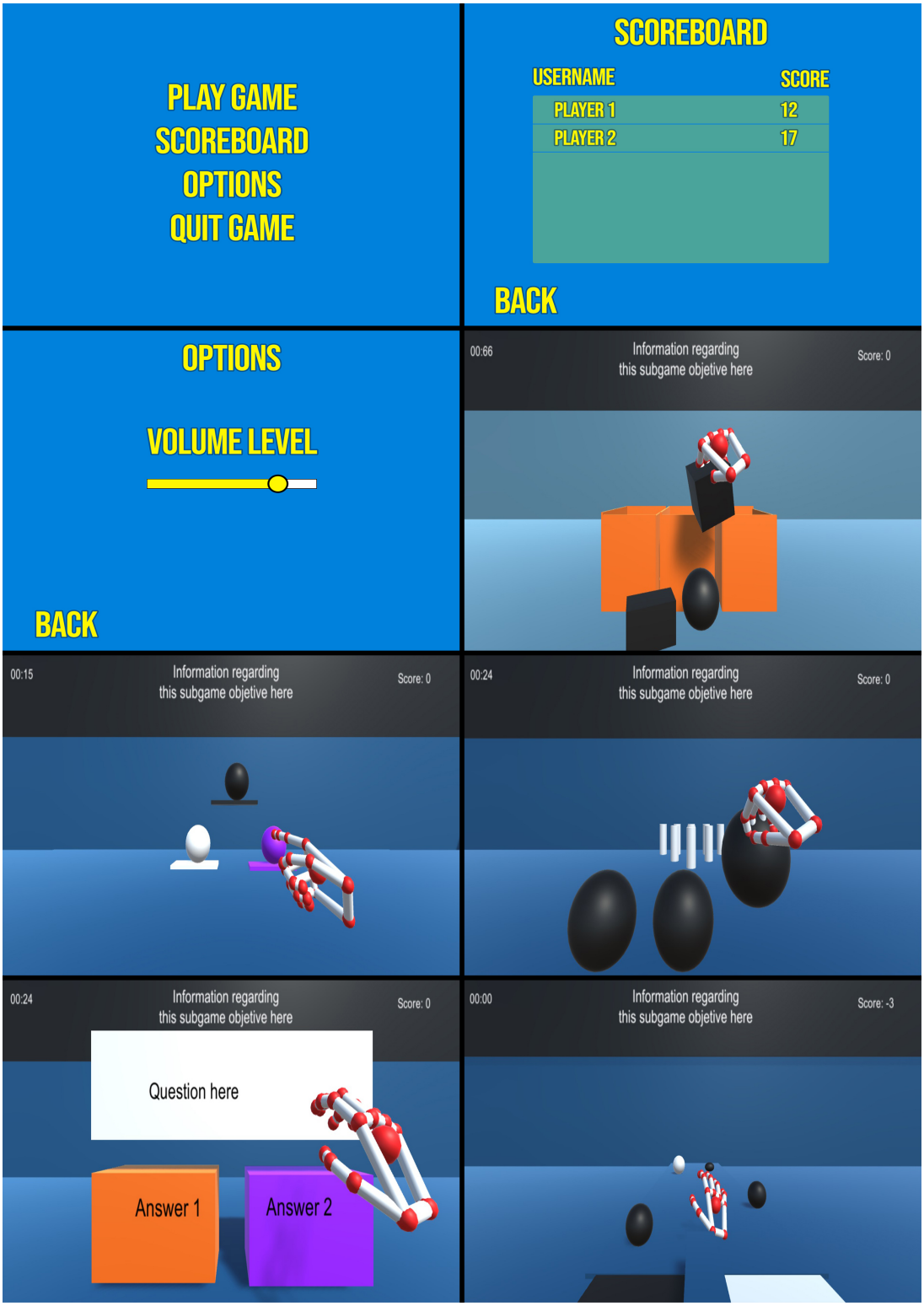


Figure 4.2: Graphic User Interface mock-ups #1

Since the mock-up concept consists of less elaborated sketches, changes can easily be introduced to them without wasting excessive time. They also promote a faster and more objective implementation phase, as the developing team will know how the intended graphic user interface structure should look. In this case, some sub-games concepts had to be reviewed, as the mock-ups showed how some previewed required movements could be too complicated for the end users to perform.

In the following sub-section, two new UML diagrams types will be introduced to properly show the interaction between the previous mock-ups and the earlier displayed preliminary classes.

4.4 Conceptual design

Sequences diagrams [46] are a type of UML diagram that aims to demonstrate how the classes within the code behave and interact with each other. These are particularly useful because of their ability to neatly display the interactions in an event-based timeline, hence the term ”sequence”. These diagrams excel at documenting system processes and demonstrating software’s requirements since the events appear according to their respective order. In this case, a UML sequence diagram was built to promote both the document building phase and the developing process. To do this, the following Figure 4.3 depicts how the classes within the sub-games scenes interact with each other in a regular playing scenario.

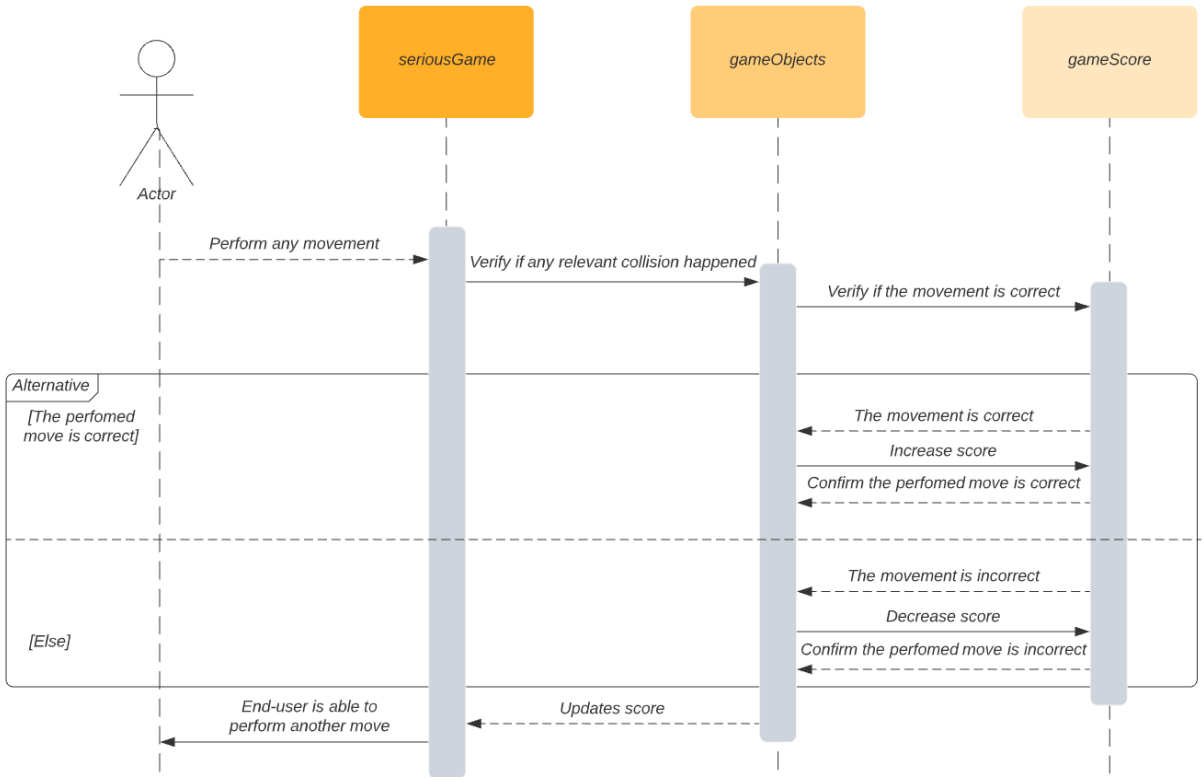


Figure 4.3: Serious game UML sequence diagram

Activity diagrams [47] illustrate a flow control representation within any event occurring inside the system. These diagrams have the advantage of being able to display sequential and concurrent activities simultaneously, to better comprehend the overall system behavior.

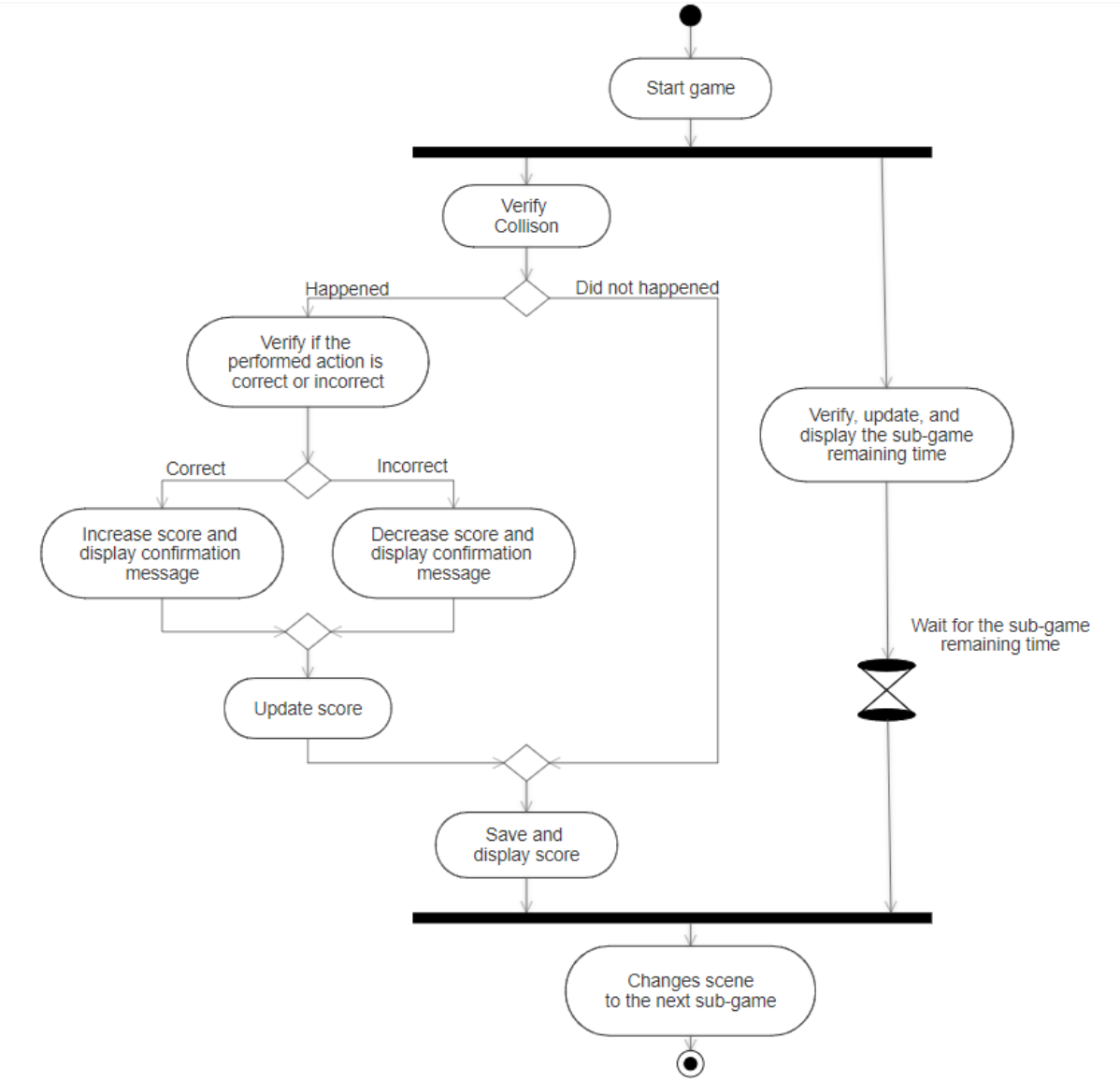


Figure 4.4: Serious game UML activity diagram

To provide context, the previous diagram shows the interaction between the serious game class and the respective objects and score. Even though the time class does not directly interact with the score or objects, it can be seen how crucial its role is to the regular serious game course.

4.5 Technical design

As described before, software design uses diagrams as basis to specify a solution that meets the specified requirements into an intuitive and interfaced visual illustration. This strategy makes the design documentation understandable in software system building process stage. These foundations are shifted into code in the implementing phase by the developing team.

Therefore, technical design [48] is a developing phase where the developers minutely describe all design previews before starting to write source system code. The existing diagrams outline what needs to be implemented in the coding stage while considering the available resources to turn the envisioned descriptions into tangible software. Overall, the technical design aims to facilitate critical analysis while contemplating the possible solutions to the respective issues. By documenting this describing process, developers have a more straightforward task when searching for anomalies, considering everything needed to be done is already minutely specified. This strategy also has the objective of simplifying the debugging process – in case of any eventual issue, searching for the flaw becomes more manageable, as every code foundation was previously specified. This concept is proven to be more efficient than reviewing the entire source code – the problem is easily spotted in the technical design document, thanks to its accurate and detailed conceptual design description.

In this project, the technical design stage was divided into two phases. Firstly, additional data structures will be presented. Then, the anticipated algorithms will be described, portraying the envisioned methods to create in-game objectives and ensuring the end-user is righteously rewarded for tackling the implemented quests.

4.5.1 Data structures

Beyond the previous classes, the following C# additional structures were also used during this project development, to support and complement the implementation process:

- Collision;
- GameObject;
- Quaternion;
- SerielizeField;
- Vector3.

4.5.2 Algorithms

In order to complete certain goals inside the serious game, a few algorithms had to be developed to make it possible. Those objectives change in each sub-game. However, they

rely on the same type of behaviors, which implies that all of them have one same algorithm in common. The main algorithms in each sub-game are the following:

- The end user has to grab and drop five geometric figures into their adequately labeled boxes in the first sub-game. Its score should increase or decrease depending on whether the move performed was correct or incorrect. The implemented algorithm attaches a specific tag to each geometric figure (for example, the sphere was attached with the tag ”sphere”). Then, a platform was placed inside of each box and, whenever any geometric figure collides with this platform, a code function linked to the platform will determine if the dropped figure matches the anticipated tag or not;

- The end user has to knock down different color spheres from open windows regarding the second sub-game. To do this, a function has to print multiple strings containing the corresponding color names to the screen, changing over a predefined time. The end-user should take down the sphere colored by the tonality indicated on the screen-printed string. A similar algorithm is applied to achieve this – the spheres are labeled with tags indicating their color, and a function is attached to the scene floor. Whenever any sphere collides with it, this function determines, with the help of the attached tags, whether or not it was the proper collision – benefiting right clashes and decreasing score when an incorrect collision happens;

- In sub-game number three, the end user faces a game similar to a bowling one. Its goal is to grab reachable spheres and throw them to take down a set of cylinders. An algorithm must count the score correctly in this specific case, so a few lines were added on the floor. Each time any cylinder collides with any line drawn in the bowling lane, the end-user score is incremented;

- In sub-game four, the end user has to answer a few multiple-choice questions – each question will be displayed in a large panel, while the two possible answers will be displayed in two smaller panels next to each other. The end user has to touch and take down one of the smaller panels to answer the questions, provoking a collision between the scene floor and the fallen panel. Applying a similar thought process to the one used in the previous sub-games, each panel containing the answers will be tagged as ”right” or ”wrong”, corresponding to the possible outcomes of the end-user response. The same function is also applied to the scene floor, and each time the panels fall, the latter function verifies if the answer is correct or incorrect;

- While playing the fifth and last sub-game, the end user has to deflect rolling oval figures, colored with two different colors, into two different paths. Two platforms were added

at the end of each direction and, consequently, the spheres have tags attached to them representing their color. Once any collision happens with the platforms, the end users score changes.

4.6 Summary

This chapter showcased the importance of sketching design models to fulfill the software requirements. When a solid and cohesive design is sketched, it adds versatility to the developing phase, efficiently opening an opportunity window for any desired change in a posterior stage. The system software architecture was then explored using UML class diagrams to demonstrate the software system component interaction. Following that, the built mock-ups of the serious game menus and sub-games were shown, since they were used to facilitate the feedback obtention process. After that the UML sequence and activity diagram were displayed to comprehend the relevance of the different system events that occur simultaneously. Lastly, the technical design is explored by explaining the intended developing process in advanced stages with a description of the sub-games required algorithms.

In the next chapter, the software implementation phase will be covered. Reasoning breakdowns behind code writing decisions will be presented, as well as other relevant information regarding code structure and software performance will be displayed.

5

Serious game implementation

Contents

The previous chapter covered all the relevant details about the serious game's software system design – it showed its illustrative process while providing a minute description of the previewed serious game. In the same chapter, this software was scrutinized through the description and respective breakdown of its classes, methods, and algorithms.

Chapter 5 presents and contextualizes the current computer graphics programming technologies about two possible game engines to develop the intended serious game. After that, a tiebreaker description takes place to choose the most proper one for the implementation requirements. Shortly after, the winning environment is analyzed, in order to understand which advantages did provide in this specific situation, as well as, which particularities did it display to create the playable prototype that later lead to the final software system. Then, the source code structure and its respective class and component diagrams are displayed and explained. Following that, the chosen source code repository is scrutinized alongside some relevant information regarding the documentation present on it. After that, the thought process behind the graphic user interface event handling is explained, providing some context regarding screen changes that happen during the gameplay flow. Ensuing this, some of the main issues found during the implementation phase are going to be discussed, followed by a detailed reference analysis of the functional and performance tests done before the testing stage.

5.1 Computer graphics programming technologies

Computer graphics [49] are computational techniques used to create and change information to generate visual content. This concept is mainly applied to three-dimensional graphics, which will be the case of this project. The latter concept includes applications for video games, and can divide itself in three main sub-fields [49, 50], being them:

- Geometry, which comprises the capability to represent and process three-dimensional objects;
- Animation, as a way of simulating the physics of a certain motion;
- Rendering, responsible for the lighting and generating images from a certain specified scene model.

The best way to manipulate the previous concept while creating a video game is to use game engines. Those can easily and quickly create a significant set of foundations, straightforwardly evolving into a real video game as they can render graphics, detect collisions, and create animations with the help of programming languages. Regarding the used programming technologies, C++, C#, and Java are amongst the most used ones. Unity is a game engine based on C# and was used to create this project. The engine uses a programming language to implement the desired algorithms and link them up to the latter mechanisms.

When looking for the best possible option to take the implementation stage forward, an in-depth research was done. After analyzing some likely game engines to develop it, two alternatives stood out among the others – Unity and Unreal Engine, both compatible with the chosen sensor, the LeapMotion controller.

The following subsection will compare both engines and select the most suitable for this work.

5.1.1 Unity versus Unreal Engine

Unity is a well-known game engine software where a game can be developed (freely for personal use). However, the latter software was not the only considered option to develop this project – Unreal Engine, another popular game development software was also considered. After analyzing and considering both options, the choice was not easy at all. On the one hand, Unreal Engine works with C++ and allowed the development of popular games, such as "Splinter Cell" and "Gears of Ware" – big-budget games that introduced next-gen physics and graphics. The latter engine also has a big advantage when it comes to creating realistic and superb visuals. On the other hand, Unity allowed to build iconic mobile games like "Temple Run" and "Pokemon GO", as well as some computer video game classics like "Assassin's Creed: Identity" and "Rust". The fact that it works with C# undoubtedly represents an advantage to Unreal Engine's programming language C++, since no memory allocation and pointers are required. Unity also has a steep learning curve, representing a huge advantage for inexperienced developers regarding video game implementing [51]. Also, Unity is more focused on lightweight games development, which is exactly the case with this serious game. Unity also promotes faster development of a playable prototype due to its intuitive interface. The biggest downside of Unity is the difficulty to make a game look as good as it does in Unreal in terms of graphic display [51], due to less available lightning options and shaders.

Table5.1 showcases a Unity head-to-head comparison with Unreal Engine. Comparison data was obtained from [52–54].

Table 5.1 Unreal Engine and Unity head to head comparison

	Unreal Engine	**Unity**
Devoloper	Tim Sweeney and Epic Games	Unit tecnologies
Language	C++	C#
Excels at	creating next gen physics and graphics	creating flash and mobile games
Typical Use	developing games with a big budget in a really detailed 3D environment specially first person shooters	developers interested in 2D games side-scrollers or mobile games
Other observations	shallow learning curve but simple to create superb visuals	steep learning curve and quick to create playable prototypes

Considering the characteristics previously presented, Unity was the chosen game engine to carry out this project. Also, since the serious game is intended to be light in order to run on any computer, there is no need to create stunning graphics. Another point in favor of Unity is

the quickness to develop a playable prototype, because the client opinion is needed as soon as possible to implement possible constraints useful in the future. Therefore, the implementation model will take place in the game engine software Unity, by building the graphic user interface, the environment, the respective objects and adding scripts to the latter objects. Unity will rely on the LeapMotion controller and hardware to capture finger, hand, and arm movement data, to virtually interact with the developed environment in the previous software.

5.2 Development environment

As stated in the previous section, Unity was the chosen game engine to implement this project. With this in mind, this software is responsible for this environment, i.e., it creates all the existing scenes, objects, and surrounding visual elements. It can also attribute fundamental particularities such as gravity, colliding behaviors, and rendering properties.

Due to the LeapMotion controller libraries being compatible with Unity, this environment also gives specific properties to the objects, allowing them to get manipulated – libraries to connect interaction behaviors to all the intended objects. This feature makes human-machine interaction possible, creating the previously mentioned bridge between fingers, hand, and arm movements within the virtual reality environment.

This environment also allows linking scripts containing classes and relevant functions to the objects, essential for algorithm development. It also provides a significant amount of freedom in designing, providing customized options for menus, fonts, and colors. For a more immersive experience, the latter software also added background music to the serious game.

5.3 Source code structure

In this project, all graphic user interface was done through the Unity development environment interface. Source code was required to implement the desired buttons, as well as to develop the needed classes and sub consequent methods and functions. Inside a folder known as ”scripts”, every source code file[1] can be found. The scripts have the same name as the classes portrayed in the following ”Final UML class diagram” that appears in the next section.

5.3.1 Final UML class diagram

A preliminary version of a UML class diagram was presented in chapter 4, containing the previewed classes and subsequent attributes and methods. After the implementation phase was over, the introduction of new classes changed the software system structure, reasonably justifying the creation of an updated UML class diagram, as depicted in Figure 5.1.

[1] Hyperlink for the Fun2HelpElderly project files and source code here.

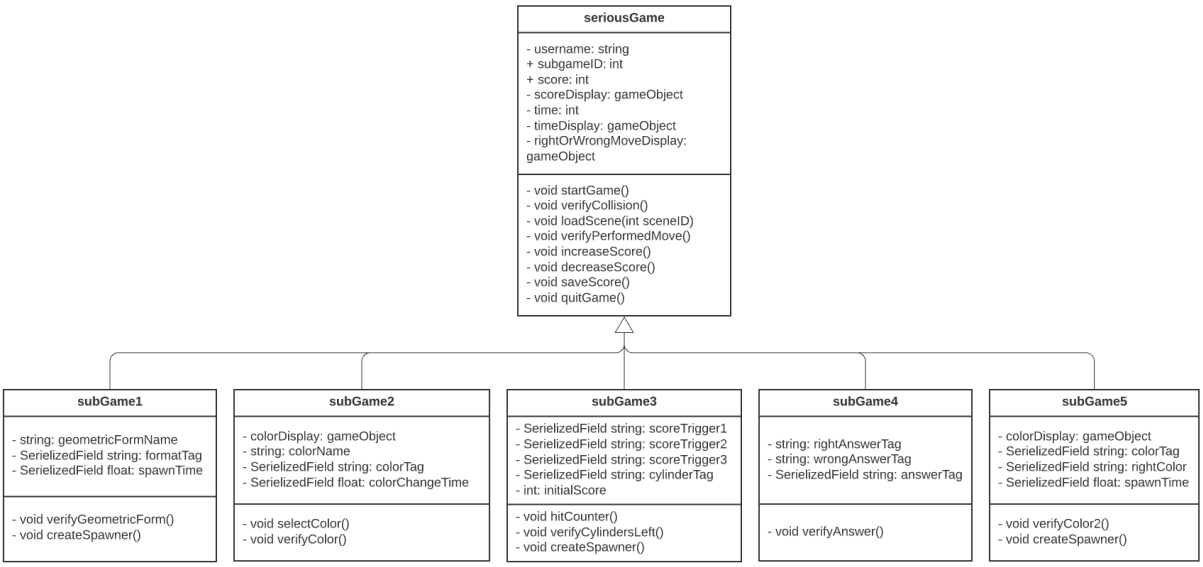


Figure 5.1: Final UML class diagram

In contrast to the preliminary UML class diagram where the time, score, and objects class revolved around the ”seriousGame” class, after some iterations, a class for every sub-game was developed, including the needed methods. This decision was made because, otherwise, the final UML class diagram would become too extensive and unnecessarily difficult to comprehend.

5.3.2 UML component diagrams

UML component diagrams [55] display the interactions and behaviors of the different components. These components equal to a group of classes representing independent systems and subsystems, and the way they interact with each other is depicted in Figure 5.2.

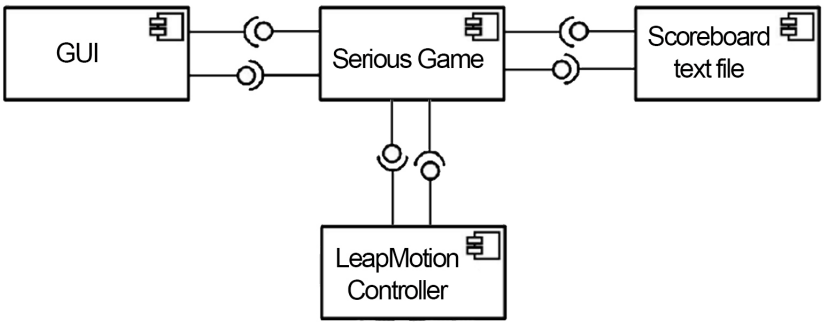


Figure 5.2: UML global system component diagram

This type of diagram facilitate the comprehension of the software system, through the concrete and accurate architecture visualization they provide. It also highlights some system components' importance by showing system simultaneous behaviors with mutual interaction. The serious game system component is depicted in the following Figure 5.3:

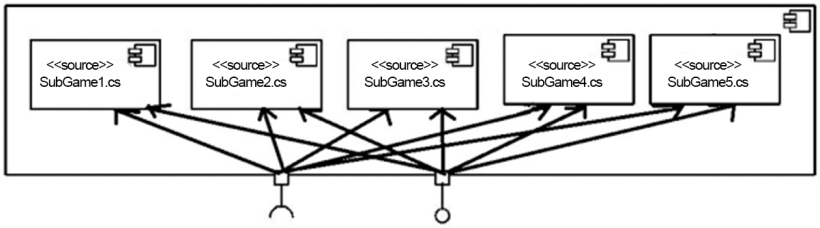


Figure 5.3: UML global system component diagram

The diagram above shows the objectives that the system components help to achieve – for both end users and non human actors.

5.3.3 Source code repository and documentation

According to [56], a source code repository consists of an archive that contains code files and simultaneously offers the opportunity to host the latter files alongside technical documentation and other relevant content. This archive can be accessed by anyone on the internet or it can be private/non-listed.

This type of repositories is useful for storing the desired project online without having to save it on a physical hard drive. Another advantage is the versatility it creates through its version control – every update to the uploaded files can be documented in the repository, which allows other programmers to join the project and be on pair with every change made, as well as to create changes themselves if they were allowed to do so. Whenever a change needs to be made to source code, there is no need to upload the whole project again, through its interface its simple to only update the file being worked on. For this project, the chosen source code repository was GitLab, so the serious game files and respective documentation can be found there, alongside a text file giving instructions on how everything inside that archives works. The latter repository can be accessed here.

All the files were uploaded to the previous repository through a software called GitKraken, responsible for bridging and allowing to transfer files between the computer directory and the online repository.

As seen before in Chapter 4, documentation [57] is a vital component of the software engineering process. It consists of the information that came along with the code to explain certain behaviors, to justify certain decision-making process and how can a random programmer use it. There are two forms of documentation, one being the comments inside the source code and the other supporting documentation to it. A well-documented project allows being worked in parallel with anyone interested, due to the minutely specified classes functions and objects. A cohesive set of documentation should make any coder understand any project, by providing enough context to continue it solely by reading the latter documentation.

To document this project, Doxygen software [58] was used. This software can automat-

ically generate documentation by capturing and reading comments in popular programming languages, in this case, C#. In this project case, besides the comments inside the source code files, the external documentation was generated through Doxygen and can be seen in Annex B section of this project.

5.4 GUI events handling

Events [59] are objects which can be handled to create certain interactions with the graphic user interface, which will change the screen appearance visible to the end-user. These objects are frequently known as ”Listeners”, as they represent an essential component of the event framework to achieve the desired event handling sequence. They are ”waiting” for a specified interaction to happen, and when it does, their response provokes a certain change in the screen.

This software is event-driven between menus because, due to human-machine interactions, their overall flow is controlled and carried by the actions performed with the computer mouse. This also excludes the possibility of this serious game being algorithm-driven, because even though the previous algorithms control the end user's score, the game will follow its natural overall flow, even if the end user does not perform any action. In the developed software, the existing GUI events [59] consist of mouse clicks to navigate between menus, time interruptions to alternate between sub-games, and windows closing events when the ”quit game” button is pressed. Regarding the main menu, the end user can select one of the four presented options with the mouse, which do a certain action through a Boolean function.

5.5 Main issues during implementation

During the implementation stage, it was noticeable how calibrations issues keep happening with the LeapMotion controller. It momentarily stops capturing or misreading hand and arm movement data, thus causing the hands models to disappear. This obligates the playing end user to re-position its hands in the default tracking position, to get its hands' model to display on the screen again. Unfortunately, after a few hours spent on investigation and troubleshooting, it was clear that it was not an implementation problem; instead, it was a LeapMotion controller issue. This unfortunate event offers sporadic unpleasant times to the end user, ultimately resulting in a less fun and overall, less engaging activity. However, with the warm-up scene implemented, the sensor saves a reference position of the end user hands, avoiding eventual issues in the beginning of each sub-game.

Another implementation issue was the difficulty to perceive the serious game environment from an elder perspective of view. This did not always allowed to certain details through an older end user lens. Meetings with the ”client” were crucial to make essential realizations about the necessity to change certain details. Even though everything ended up working out, a significant amount of time was lost in this corrective process.

5.6 Functional and performance tests

Before approaching a severe testing stage, later mentioned in Chapter 6, this serious game went through a series of functional and performance tests. Initially, these software essential tasks were tested – menu navigation, score counting and saving, scene changing and messages displaying. Then, after making sure the serious game prototype was not only playable, but also could validate a more severe testing stage, four elders were chosen to be part of six functional testing sessions, which consisted of the repetition of the five sub-games. In each session, each end user score was saved, and the fastest goal achievement movement in each sub-game was timed – depicted in Figure 5.4 in the Y axis named "Time required". End users' feedback was also considered, motivating strategic software aesthetic changes to turn the serious game more accessible. At the end of the sessions, a warm-up scene was introduced before every sub-game. These scenes contained a simpler sample of the goal the end user had to complete before the sub-game started. This created a calibration checkpoint, reducing the LeapMotion controller chances of momentarily missing calibrating, just like mentioned in the previous section. According to the table included in Annex D.1, a steep improvement curve is verified in all of the end-users, even for those that started the first session with low score points. The time improvement column regarding quest tackling also validates this serious game prototype positive impact this serious game had. To properly visualize the previously mentioned scenario, some graphics were built and depicted in Figure 5.4:

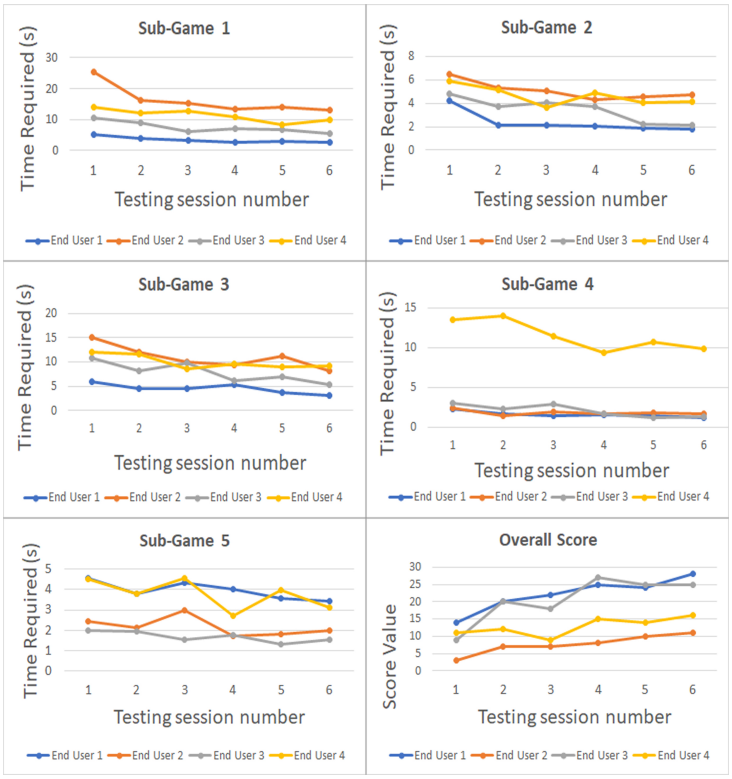


Figure 5.4: Time Required to tackle the desired quests in each sub-game and overall score

This serious game, alongside LeapMotion controller software, were also installed and tested in a different computer with average hardware specifications. This test aimed to validate if the developed project would run in any computer and it was successfully passed.

5.7 Summary

This chapter started to explain what revolves around computer graphics programming technologies by contextualizing their current place in the technology industry, how they work, and why was the serious game engine Unity the chosen one, between all the available options. After that, the development environment inside the latter game engine is explored, by explaining all the leading processes towards the aesthetic and the physics of the serious game. Regarding that, the source code structure is also dissected, followed by the addition of the final UML class diagram for better class comprehension, as well as the UML component diagram can display all the interactions between subsystems, classes, and objects. After that, the source code notion was explained, as well as, why it was used its consequent advantages of it. A brief mention of the existing documentation in this project annexes and repository and why it is so important is also provided. Later on, the GUI events handling is specified to give a better context about every change within the scenes and consequently on the screen. Following that, the main issues that happened during the implementation were detailed, mainly consisting of LeapMotion controller calibration issues and difficulties in observing the serious game through the perspective of view of an elder. Finally, functional and performance tests are specified, portraying how the first playable prototype behaves inside a chosen sample of elders. These results show that the developed software is ready to advance to a more serious testing scenario, with a random and wider end user sample – this scenario will expanded on the next chapter.

6

Acceptance tests with end users

Contents

The previous chapter focused on the software system implementation phase by providing information regarding the development environment, code structure, and other essential elements to describe the software building process. A relevant functional testing stage also was documented, showing conclusive results regarding the serious game prototype usability level. All the end users who participated in those sessions showed progress compared to their first tries, giving crucial feedback to fix minor issues within the serious game. The previous changes were implemented, and the serious game final release version was ready for a more severe, controlled, and documented testing stage.

In this chapter the testing scenario will be addressed by providing an ample explanation of the idealized concept. After that, a detailed description of the test specifications and preparations will be given, minutely detailing the main points evaluated in that given testing phase and scheduling the period in which they will occur. Essential documentation to take additional conclusions regarding the serious game efficiency is also displayed, alongside specified usability forms and consent statements. Following that, the obtained results will be analyzed to properly formulate substantiated conclusions.

6.1 Testing scenario

To properly develop a trustworthy testing environment, eleven elders were asked to participate in the acceptance test, of the serious game final version, which will let elders use the developed software during a specific period. Simultaneously, their performance will be studied to understand how useful, attractive, and entertaining the latter can be. This can happen through several ways, such as, initial and final performance comparison, forms fulfillment, and personal feedback acquisition. These concepts will be explored in the following sections, providing an in-depth view of what was done to achieve the latter and how that proves what needs to be confirmed.

6.2 Tests specification and preparation

The executed tests evaluated this serious game usability value according to the end user performance in the developed software, either through their in-game final score comparison or through the measurement of the fastest time required to perform a particular movement associated with goal achieving. Those movements were compared between the initial and final stages, to measure dexterity progress, which allowed to perceive the end user dexterity and hand-eye coordination progress. Testers that give permission to be video recorded also create an interesting comparison term – if the contrast between initial and final days footage is clear, the improvement is undeniable.

Form filling was also used to understand the serious game usability and entertaining value. In the initial and last sessions, the end users will be asked to fill out questionnaires regarding

their experience with the developed software. This would lead to the obtention of accurate feedback, as well as to understand if a perceptive evolution also happened.

The following subsections will discuss the previously mentioned ways to get essential data regarding the serious game validation.

6.2.1 Test scheduling

This testing stage encouraged the end users to play the serious game for half an hour during eight consecutive days. This phase started on February 3rd and lasted until February 10th.

The planned schedule aimed to create gradual progress regarding end user performances through consistency and practice – by repeating the same sub-games every day, the exact body mechanisms are also getting exercised and stimulated. Ideally, the end user would end this period with improved reaction times and a better performance regarding in-game goal achieving.

6.2.2 Environment setup

In this testing stage, the computer containing the serious game should be standing on a plain table. Then, the LeapMotion controller should be plugged into the computer, staying at approximately 25cm, so the end user has the opportunity to manage the virtual hands' depth control optimally. The setup setting used during this testing stage is depicted in Figure 6.1:

Figure 6.1: Environment setup

The latter setup allows the elder user to play the serious game while comfortably sitting on a chair, avoiding unnecessary tiredness and fatigue. Also, the fact that it only requires a laptop and a small but affordable controller represents portability and montage simplicity – a significant advantage to fulfill this project main objectives.

6.2.3 Usability forms

After using the serious game, it is important to question the end user about its experience. This will allow to get a better comprehension of the developed software usability and entertaining values. To achieve the previous goal, two usability scales were created, namely FlowShort Scale [19, 60] – also known as FSS – and System Usability Scale [19, 61], known as SUS.

The SUS scale was created to measure usability, and it is based on a ten-item adaptable form – the end user should rate from one (strongly disagree) to five (strongly agree), given its opinion towards the evaluated software. It is proven to be able to be used on small testers samples and still produce reliable results. Those previously mentioned ten core questions are usually similar but should be adapted to the system being evaluated.

Regarding the FSS scale, it aims to measure flow, defined as the perfect balanced state of mind between boredom and anxiety. More than usability, this form has the objective of understanding what feelings did a certain experience provide, by asking the end users who fill it to rate fourteen items on a scale of 1 to 7, given that the last three statements are asked to be rated on a scale of 1 to 9. Like in the previous model, the latter SUS statements should be adapted to the software interest.

Since the sample in question is mostly constituted by elder people, asking them to fill both forms could be an exhausting and complex task. To fix that, a new usability form was created, featuring the most crucial questions out of the latter two. This usability forms variants adapted to the developed serious game can be found at the end of this document, inside Annex C.

6.2.4 End user consent statement

As previously mentioned, statistical data regarding end users' personal information is crucial to properly carry out a cohesive study. Still, the end user continually consented to share the previously mentioned data. Therefore, to obtain end users' names, age, physical and mental conditions, forms responses, and video recordings of their gameplay, their authorization must be undoubtedly guaranteed. To acquire those important parameters that allow to analyze the end user's evolution, a consent statement contracted was written, ensuring every participant gave their consent about sharing their data with this project.

Besides the previously mentioned forms, Annex C also contains a copy of the developed consent statement.

6.3 Obtained Results

After the last tests, the results included in annex D.2 were collected. To provide context about the developed serious game impact, the obtained results were discussed and used to create following linear graphics 6.2:

Figure 6.2: Time Required to tackle the desired quests in each sub-game and overall score after eight testing days

The graphics depicted in Figure 6.2 leave no doubt about the serious game effectiveness. Based on its results, this serious game appears to represent a benefit to the end users that par-

ticipated in the eight days testing stage. According to the collected data related to the time required axis, every end-user managed to consistently perform the sub-games required tasks in less time compared to their starting point, which is directly related to a notorious hand-eye coordination improvement. The previous graphics also demonstrate how the overall score values improved in every end user during the previously mentioned time span. This sustains the idea that the end users' cognitive component was stimulated during the eight day testing stage, which allowed them to display a significant dexterity evolution represented by the serious game score improvement.

6.3.1 Characterization of the end-user sample

To contextualize the participant end users background, the following two pie graphs represented in Figures 6.3 and 6.4 below will portray both age and gender representation of those who participated in this testing stage.

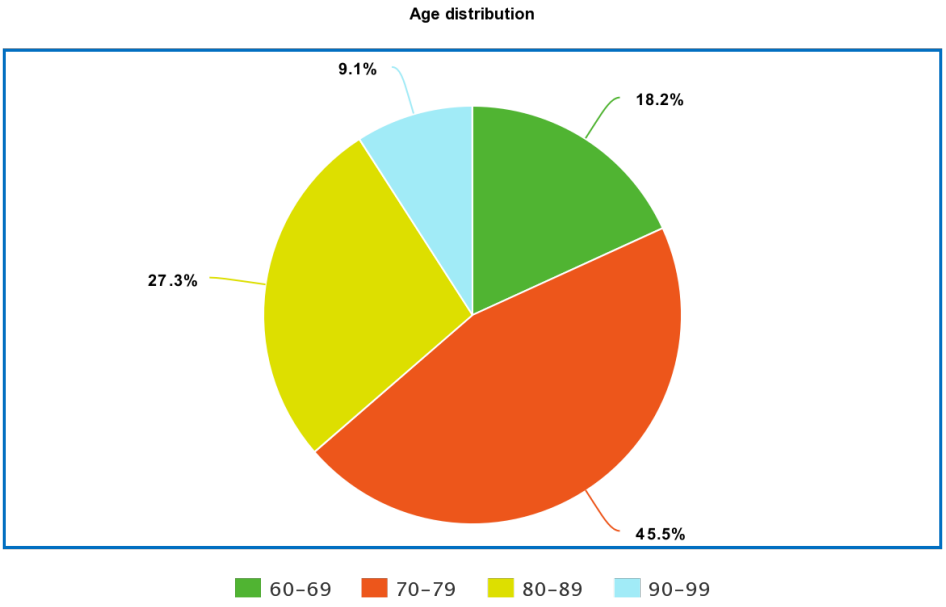


Figure 6.3: Age distribution of the end-user sample

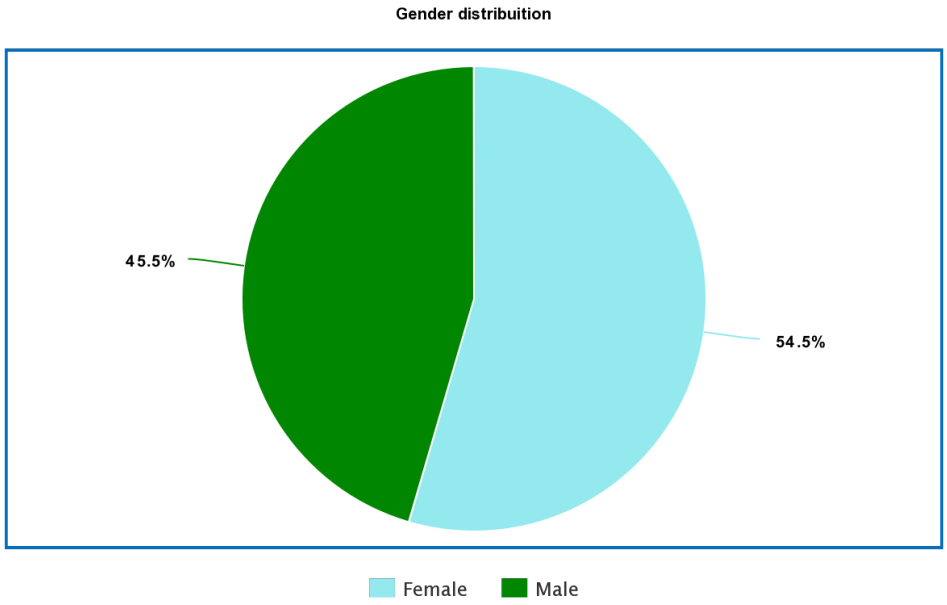


Figure 6.4: Gender distribution of the end-user sample

According to previous pie graphs depicted in Figures 6.3 and 6.4, it is easily seen that female participation in this study slightly overcame male participation. Also, end users with an age comprehended between 70 and 80 years were the most present.

It is also relevant to mention that three end users of the considered sample had physical impairments during the testing phase. This condition may lead to inferior scores, more time required to achieve the serious game objectives, and a different perception of the form questionnaires because of that – which end up being the case. Those three participants were the ones that revealed more difficulties to play the serious game. However, according to the filled forms, they seemed to enjoy this serious game more than other healthy participants did.

6.3.2 Analysis and discussion

According to the information above, this serious game firmly allowed its end users to manifest a significant amount of progress during the eight days testing stages. Every elder involved managed to improve their score gradually and fasten their reaction time regarding in-game objective completion. The overtime difference in ”Required time” and ”Score values” helps to prove how the end users improved during this testing sessions. Analyzing the ”Overall Score” chart, it can be seen that every participant made a huge progress. End user 4, which happened to be one of the participants with physical impairments, was the participant that made the least progress, but still managed to improve its performance score in 58% – started with a score of 23 in the first day, and in the last day obtained a score of 28 D.2.

It is also essential to notice that the end users were given one usability form in the first

day and one in the last day. The results were also expressive since all of them found the game more appealing according to the FSS scale and more useful regarding the SUS scale. This also demonstrates the evolution during the previously mentioned period.

6.4 Summary

This chapter minutely analyzed the testing stage. It started by briefly describing the testing scenario, followed by an explanation regarding its specific progress measures, test scheduling, and environment setup within the latter tests. Following that, the decision of choosing the FSS and the SUS scales to estimate usability and flow was also explained. To finalize the chapter, the obtained results were displayed and scrutinized to properly justify that the developed software gathers enough conditions to fall under the category of a serious game.

7

Conclusions

Contents

The world population average age is gradually increasing, which by consequence has a tremendous impact in an already ageing society. Hence, it is becoming crucial to create quality improvement conditions for the mentioned age group, which will ultimately also strengthen society as a result, since elders already represent such a big percentage of the overall population. To do this, the developed project aimed to help elder people increase their life quality. The objective was to build a serious game with therapeutic validation that was simultaneously fun and entertaining.

Based on the obtained results, this serious game was proven to represent a benefit to the targeted age group. According to the collected data, the elders that participated in the eight days testing sessions improved their in-game performance through continuous cognitive stimulation, ultimately resulting in a final score increase over that period. They also demonstrated a significant evolution in the required time to tackle each sub-game quests, which is directly related to hand-eye coordination. That data by itself unveils and attributes an undeniable potential to this project.

Also, by asking the end users to fill the forms in the first and in the last sessions, also became clear that based on the registered answers, their interest in the serious game also grew, which proves the entertainment and fun goal of this serious game was also achieved.

Finally, the recorded video footage of the testing sessions displays a significant contrast between the first and the last days by showing a visible improvement in the end user's dexterity and cognitive capabilities.

This serious game not only differentiates itself from other similar ones due its portability – LeapMotion controller is a small device and it is playable on almost any laptop – but also because of the low level assistance and professional monitoring it requires to be played.

7.1 Future Work

This work future directions point to the continuation of the serious game graphical and technical improvement, by including sound and visual effects to turn the virtual experience even more immersive and entertaining. Also, innovative and challenging new algorithms could be added to game, to engage and motivate players that already understood the required game mechanics and already complete it effortlessly.

Developing more and distinct sub-games could also be a convenient way of introducing therapeutic validation to a wider sample of end users. These additional scenes would establish a significant new range of options, which could open an opportunity window to specialize and group the latter scenes towards a specific goal or rehabilitation role. This concept would encompass a set of sub-games focused on exercising and stimulating exclusively a specific predefined characteristic of the end user.

It would be also interesting to test this serious game, and analyze the obtained results, with an end user sample made of people with different physical and cognitive impairments. This would provide a more concrete idea of this serious game's physical and mental restrictions, as well as allowing to comprehend if the obtained success rate during this project testing stage would remain similar with less capable end users.

Bibliography

[1] P. INE, *População residente: total e por grandes grupos etários*, 2020, "Available at https://www.pordata.pt/DB/Portugal/Ambiente+de+Consulta/Tabela.

[2] J. Heafner, "Explaining Your Rehab Time Frames," 2019.

[3] J. Pauchot, L. Di Tommaso, A. Lounis, M. Benassarou, P. Mathieu, D. Bernot, and S. Aubry, "Leap motion gesture control with carestream software in the operating room to control imaging: Installation guide and discussion," *Surgical Innovation*, vol. 22, no. 6, pp. 615–620, 2015.

[4] *Nintendo Wii – Hardware Information*, 2006, available at https://web.archive.org/web/20080212080618, archived from the original http://wii.nintendo.com/controller.jsp.

[5] *Kinect Protocol Documentation*, 2013, available at https://openkinect.org/wiki/Protocol_Documentation#Control_Commands;a=summary.

[6] *LeapMotion controller*, available at https://www.ultraleap.com/product/leap-motion-controller/.

[7] Leapmotion.com, "Firmware Reset _ Leap Motion Developers," 2021.

[8] Completemyassignment.com, "Healthcare Assignment - Sample Healthcare Assignment - Sample," 2017.

[9] S. Martel, "Beyond imaging: Macro-and microscale medical robots actuated by clinical mri scanners," *Science Robotics*, vol. 2, no. 3, p. eaam8119, 2017.

[10] M. Knapp, L. Thorgrimsen, A. Patel, A. Spector, A. Hallam, B. Woods, and M. Orrell, "Cognitive stimulation therapy for people with dementia: cost-effectiveness analysis," *The British Journal of Psychiatry*, vol. 188, no. 6, pp. 574–580, 2006.

[11] D. Borton, "What are some types of rehabilitative technologies? — NICHD - Eunice Kennedy Shriver National Institute of Child Health and Human Development," 2013. [Online]. Available: https://www.nichd.nih.gov/health/topics/rehabtech/conditioninfo/use

[12] M. Gordon, "Ageing Europe," *Bmj*, vol. 315, no. 7115, p. 1103, 1997.

[13] P. Stenner, T. McFarquhar, and A. Bowling, "Older people and 'active ageing': Subjective aspects of ageing actively," *Journal of Health Psychology*, vol. 16, no. 3, pp. 467–477, 2011, pMID: 21224334. [Online]. Available: https://doi.org/10.1177/1359105310384298

[14] S. Tardif and M. Simard, "Cognitive stimulation programs in healthy elderly: a review," *International journal of Alzheimer's disease*, vol. 2011, 2011.

[15] B. A. Wilson, "Cognitive rehabilitation: How it is and how it might be," *Journal of the International Neuropsychological Society*, vol. 3, no. 5, pp. 487–496, 1997.

[16] M. Eckroth-Bucher and J. Siberski, "Preserving cognition through an integrated cognitive stimulation and training program," *American Journal of Alzheimer's Disease & Other Dementias®*, vol. 24, no. 3, pp. 234–245, 2009.

[17] P. Gamito, J. Oliveira, D. Morais, C. Coelho, N. Santos, C. Alves, A. Galamba, M. Soeiro, M. Yerra, H. French *et al.*, "Cognitive stimulation of elderly individuals with instrumental virtual reality-based activities of daily life: pre-post treatment study," *Cyberpsychology, behavior, and social networking*, vol. 22, no. 1, pp. 69–75, 2019.

[18] L. C. PhD and R. T. Woods, "Cognitive training and cognitive rehabilitation for people with early-stage alzheimer's disease: A review," *Neuropsychological Rehabilitation*, vol. 14, no. 4, pp. 385–401, 2004. [Online]. Available: https://doi.org/10.1080/09602010443000074

[19] P. Menezes and R. P. Rocha, "Promotion of active ageing through interactive artificial agents in a smart environment," *SN Applied Sciences*, vol. 3, no. 5, pp. 1–15, 2021.

[20] D. H. Ballard, M. M. Hayhoe, F. Li, and S. D. Whitehead, "Hand-eye coordination during sequential tasks," *Philosophical Transactions of the Royal Society of London. Series B: Biological Sciences*, vol. 337, no. 1281, pp. 331–339, 1992.

[21] P. L. Gribble, S. Everling, K. Ford, and A. Mattar, "Hand-eye coordination for rapid pointing movements," *Experimental brain research*, vol. 145, no. 3, pp. 372–382, 2002.

[22] M. Inaba and H. Inoue, "Hand eye coordination in rope handling," *Journal of the Robotics Society of Japan*, vol. 3, no. 6, pp. 538–547, 1985.

[23] P. L. Gribble, S. Everling, K. Ford, and A. Mattar, "Hand-eye coordination for rapid pointing movements," *Experimental brain research*, vol. 145, no. 3, pp. 372–382, 2002.

[24] P. Boulinguez, J. Blouin, and V. Nougier, "The gap effect for eye and hand movements in double-step pointing," *Experimental brain research. Experimentelle Hirnforschung. Expérimentation cérébrale*, vol. 138, pp. 352–8, 07 2001.

[25] D. P. Carey, "Eye–hand coordination: Eye to hand or hand to eye?" *Current Biology*, vol. 10, no. 11, pp. R416–R419, 2000.

[26] H. Barbosa, "S g r e a," vol. 6, no. 1, pp. 275–283, 2018.

[27] S. Spiegel and R. Hoinkes, "Immersive serious games for large scale multiplayer dialogue and cocreation," *Serious Games: Mechanisms and Effects*, no. 2005, pp. 469–485, 2009.

[28] H. Chi, E. Agama, and Z. G. Prodanoff, "Developing serious games to promote cognitive abilities for the elderly," in *2017 IEEE 5th International Conference on Serious Games and Applications for Health (SeGAH)*, 2017, pp. 1–8.

[29] S. Kang, S. Kang, and H. Kim, "Fitness image model for individual with disabled," *2014 International Conference on IT Convergence and Security, ICITCS 2014*, pp. 2008–2010, 2014.

[30] Y. J. Chang, S. F. Chen, and J. D. Huang, "A Kinect-based system for physical rehabilitation: A pilot study for young adults with motor disabilities," *Research in Developmental Disabilities*, vol. 32, no. 6, pp. 2566–2570, 2011. [Online]. Available: http://dx.doi.org/10.1016/j.ridd.2011.07.002

[31] Web.archive.org, "Wii," 2008.

[32] *Unity Game Engine Guide: How to Get Started with the Most Popular Game Engine Out There*, 2020, available at https://www.freecodecamp.org/news/unity-game-engine-guide-how-to-get-started-with-the-most-popular-game-engine-out-there/.

[33] R. Malan, D. Bredemeyer *et al.*, "Functional requirements and use cases," *Bredemeyer Consulting*, 2001.

[34] G. T. Guedes, *UML 2-Uma abordagem prática*. Novatec Editora, 2018.

[35] L. Chung, B. Nixon, E. Yu, and J. Mylopoulos, "Non-functional requirements," *Software Engineering*, 2000.

[36] A. Tang and H. van Vliet, "Modeling constraints improves software architecture design reasoning," in *2009 Joint Working IEEE/IFIP Conference on Software Architecture European Conference on Software Architecture*, Sep. 2009, pp. 253–256.

[37] R. Hoda, J. Noble, and S. Marshall, "Agile project management," in *New Zealand computer science research student conference*, vol. 6, 2008, pp. 218–221.

[38] K. Waters, "Prioritization using moscow," *Agile Planning*, vol. 12, p. 31, 2009.

[39] R. N. Taylor and A. Van der Hoek, "Software design and architecture the once and future focus of software engineering," in *Future of Software Engineering (FOSE'07)*. IEEE, 2007, pp. 226–243.

[40] A. Tang, A. Aleti, J. Burge, and H. van Vliet, "What makes software design effective?" *Design Studies*, vol. 31, no. 6, pp. 614–640, 2010.

[41] G. G. Roy, "Designing and explaining programs with a literate pseudocode," *Journal on Educational Resources in Computing (JERIC)*, vol. 6, no. 1, pp. 1–es, 2006.

[42] R. F. Paige, J. S. Ostroff, and P. J. Brooke, "Principles for modeling language design," *Information and Software Technology*, vol. 42, no. 10, pp. 665–675, 2000.

[43] D. Garlan, "Software architecture," 2008.

[44] M. Shahin, P. Liang, and M. A. Babar, "A systematic review of software architecture visualization techniques," *Journal of Systems and Software*, vol. 94, pp. 161–185, 2014.

[45] J. M. Rivero, G. Rossi, J. Grigera, J. Burella, E. R. Luna, and S. Gordillo, "From mockups to user interface models: an extensible model driven approach," in *International Conference on Web Engineering*. Springer, 2010, pp. 13–24.

[46] S. Bernardi, S. Donatelli, and J. Merseguer, "From uml sequence diagrams and statecharts to analysable petri net models," in *Proceedings of the 3rd international workshop on Software and performance*, 2002, pp. 35–45.

[47] M. Dumas and A. H. Ter Hofstede, "Uml activity diagrams as a workflow specification language," in *International conference on the unified modeling language*. Springer, 2001, pp. 76–90.

[48] R. Bähre, B. Döbrich, J. Dreyling-Eschweiler, S. Ghazaryan, R. Hodajerdi, D. Horns, F. Januschek, E.-A. Knabbe, A. Lindner, D. Notz *et al.*, "Any light particle search ii—technical design report," *Journal of Instrumentation*, vol. 8, no. 09, p. T09001, 2013.

[49] J. Bruno, V. Jan, and A. Slaby, "Computer graphics in computer science education," *Problems of Education in the 21st Century*, vol. 11, p. 60, 2009.

[50] A. Watt, "3d computer graphics," 1993.

[51] M. Dealessandri, *What is the best game engine: is Unity right for you?*, 2020, available at https://www.gamesindustry.biz/articles/2020-01-16-what-is-the-best-game-engine-is-unity-the-right-game-engine-for-you.

[52] B. W., "How to Choose The Right Game Engine For Your Video Game_ 7 Factors To Consider [Unreal Engine or Unity Engine_]," 2021.

[53] Xsolla, "Which Game Engine Is Best For You _ Xsolla," 2020.

[54] J. Byrne, "Design Notes, Tutorials and Articles_ UDK or Unity_ Which is better_," 2014.

[55] D. Bell, "Uml basics: The component diagram," *IBM Global Services*, 2004.

[56] J. Sayago-Heredia, R. Pérez-Castillo, and M. Piattini, "A systematic mapping study on analysis of code repositories," *Informatica*, vol. 32, no. 3, pp. 619–660, 2021.

[57] T. C. Lethbridge, J. Singer, and A. Forward, "How software engineers use documentation: The state of the practice," *IEEE software*, vol. 20, no. 6, pp. 35–39, 2003.

[58] D. van Heesch, *Doxygen index*, available at https://www.doxygen.nl/index.html.

[59] J. Chen, "Formal modelling of java gui event handling," in *International Conference on Formal Engineering Methods*. Springer, 2002, pp. 359–370.

[60] S. Engeser and F. Rheinberg, "Flow, performance and moderators of challenge-skill balance," *Motivation and Emotion*, vol. 32, no. 3, pp. 158–172, 2008.

[61] P. T. Kortum and A. Bangor, "Usability ratings for everyday products measured with the system usability scale," *International Journal of Human-Computer Interaction*, vol. 29, no. 2, pp. 67–76, 2013.

[62] D. C. Rios, T. Gilbertson, S. W. McCoy, R. Price, K. Gutman, K. E. Miller, A. Fechko, and C. T. Moritz, "Neuro game Therapy to improve wrist control in children with cerebral palsy: A case series," *Developmental Neurorehabilitation*, vol. 16, no. 6, pp. 398–409, 2013.

[63] M. Jensen, "Design vs," 2016.

Annex A - User manual

Para jogar o jogo sério, introduza a porta USB do LeapMotion controller na entrada USB do seu computador, como indicado na Figura A.1 seguinte:

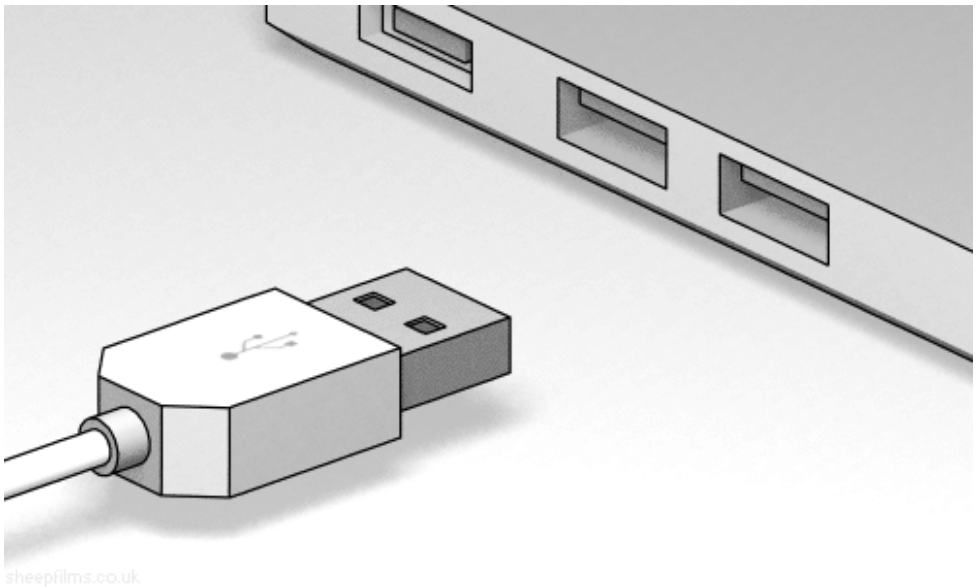


Figure A.1: Ilustração da porta USB do LeapMotion controller e respetiva entrada no computador

Após isto, é necessário instalar o software do LeapMotion controller. Para o fazer siga os passos representados nas figuras abaixo:

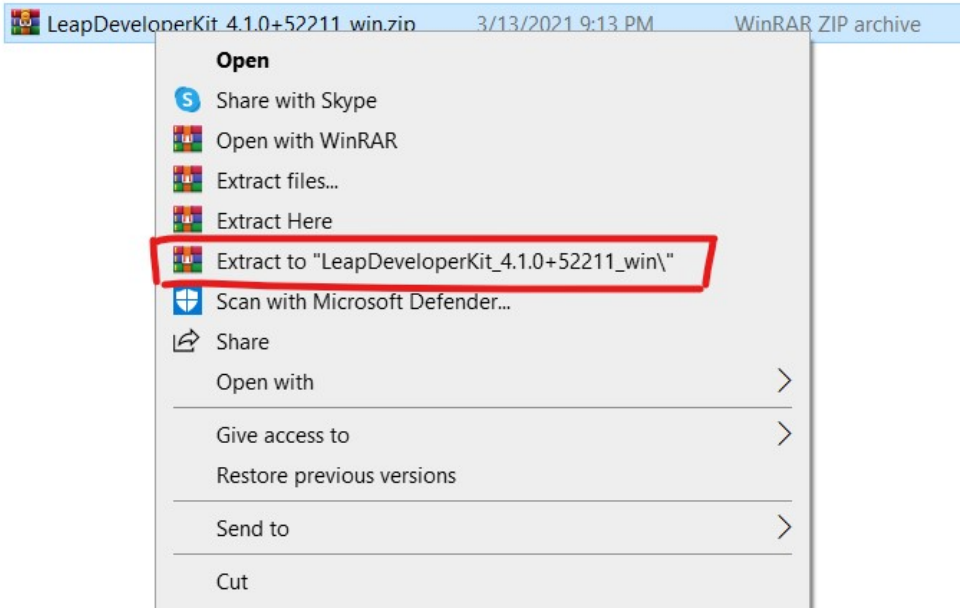


Figure A.2: Extrair a pasta do instalador que se encontra num arquivo comprimido

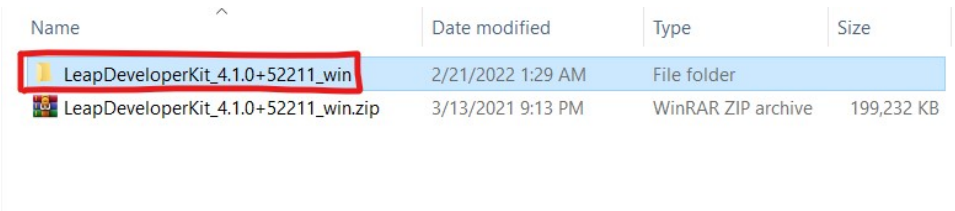


Figure A.3: Abrir a pasta resultante do processo anterior

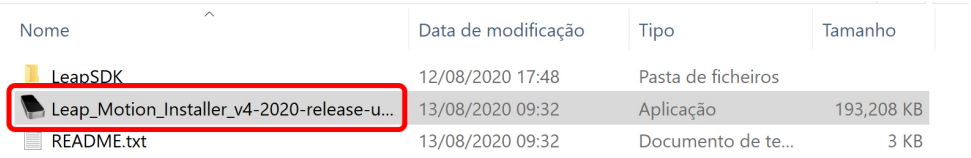


Figure A.4: Instalador do LeapMotion controller software – Selecionar e clicar na aplicação assinalada a vermelho

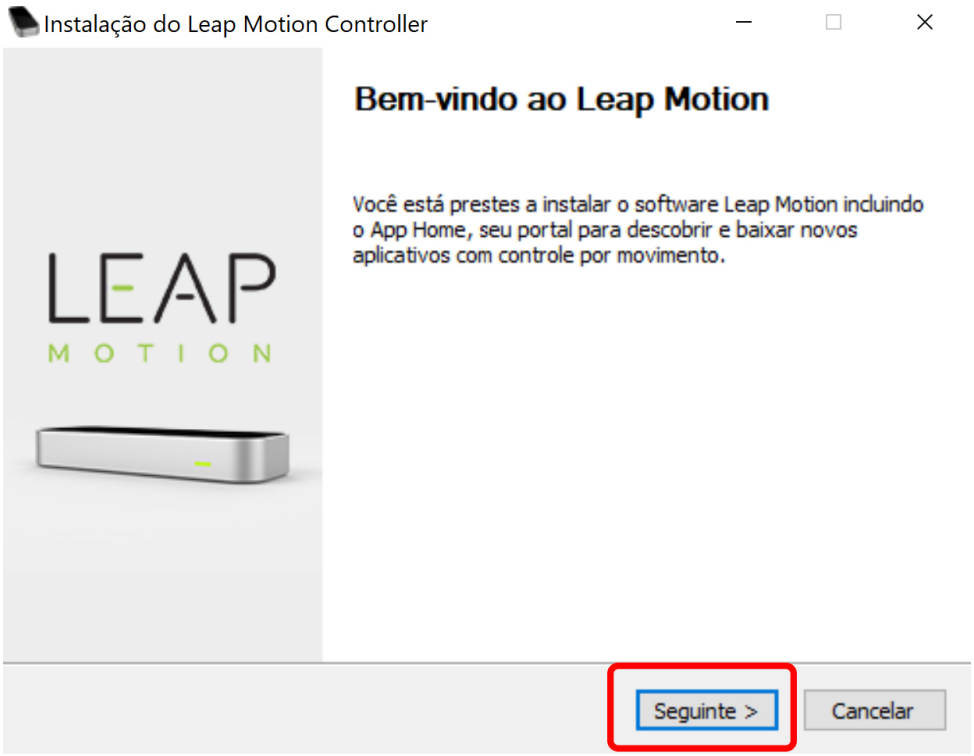


Figure A.5: Selecionar o botão ”Seguinte”

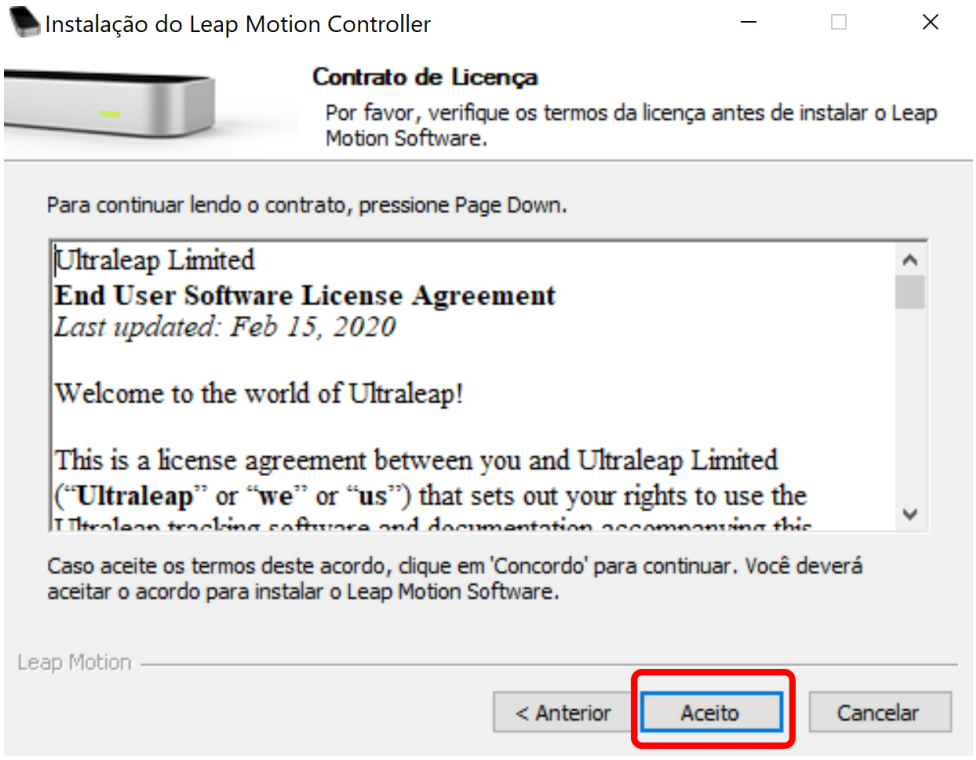


Figure A.6: Ler os termos do contrato de licença e pressionar o botão "Aceito"

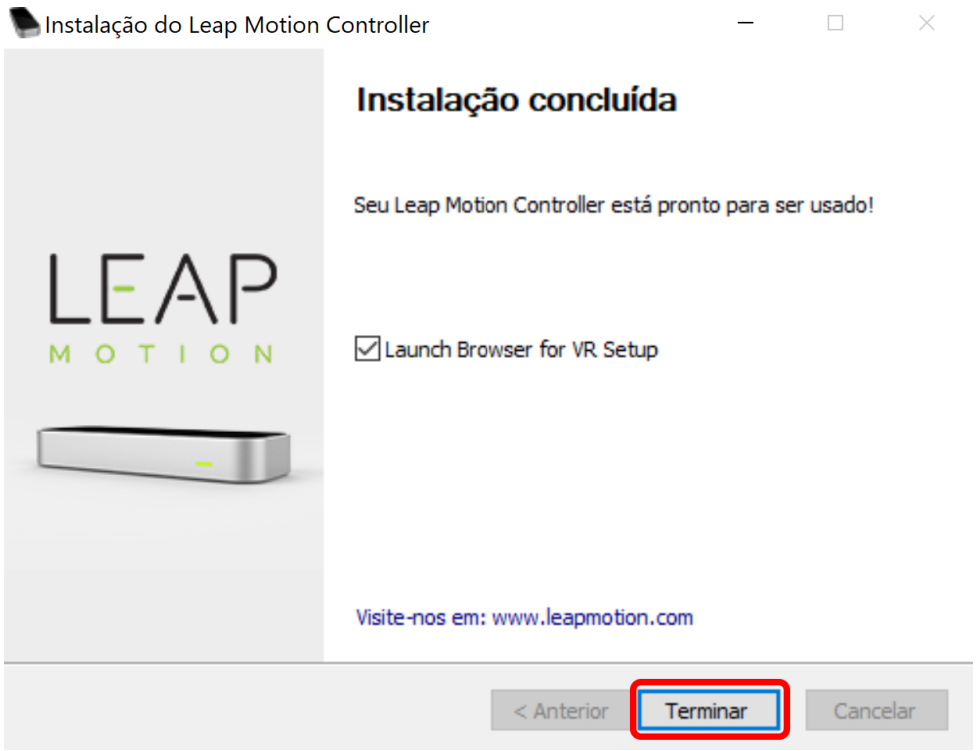


Figure A.7: Selecionar o botão "Terminar"

Deverá aparecer no seu computador a seguinte notificação, no canto inferior direito, repre-

sentada na Figura A.8, para confirmar que o software ficou devidamente instalado.

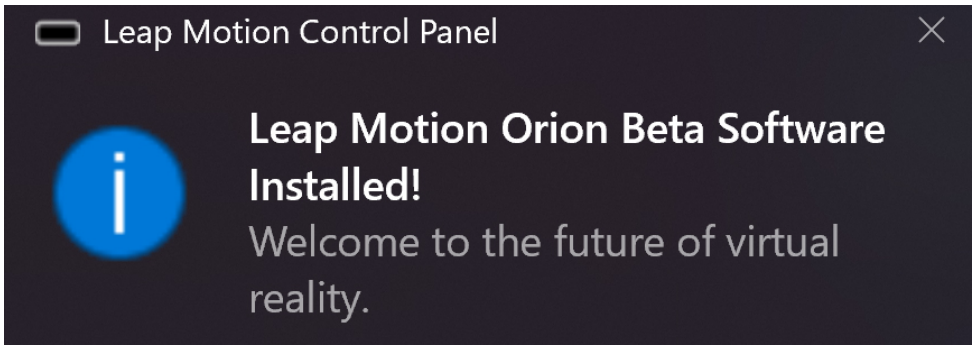


Figure A.8: Notificação de confirmação da instalação do software do LeapMotion controller

De forma a confirmar esta informação abra o software do LeapMotion controller. Para isto clique no botão representado como ”1” na Figura A.9, e de seguida, clique no icon representado como ”2”, também nessa mesma figura. Este último deverá aparecer aparecer com a cor verde ou amarela. Caso apareça com a cor vermelha significa que o LeapMotion controller não está devidamente conectado ao computador.

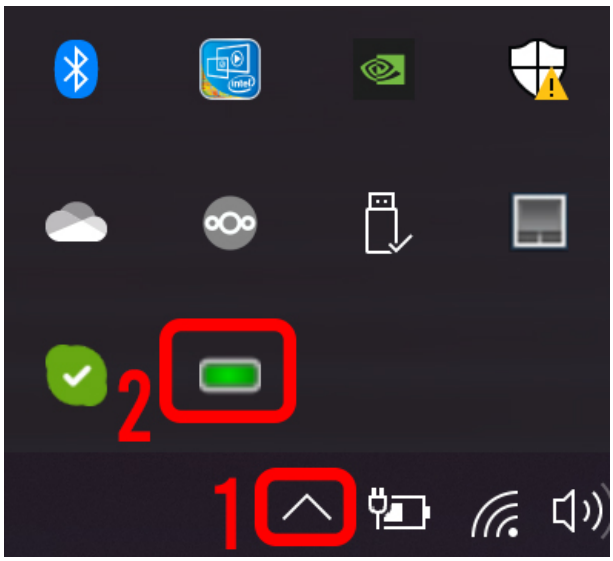


Figure A.9: Abrir a janela do software do LeapMotion controller

Siga os passos seguintes, assinalados a vermelho nas Figuras A.10 e A.11:

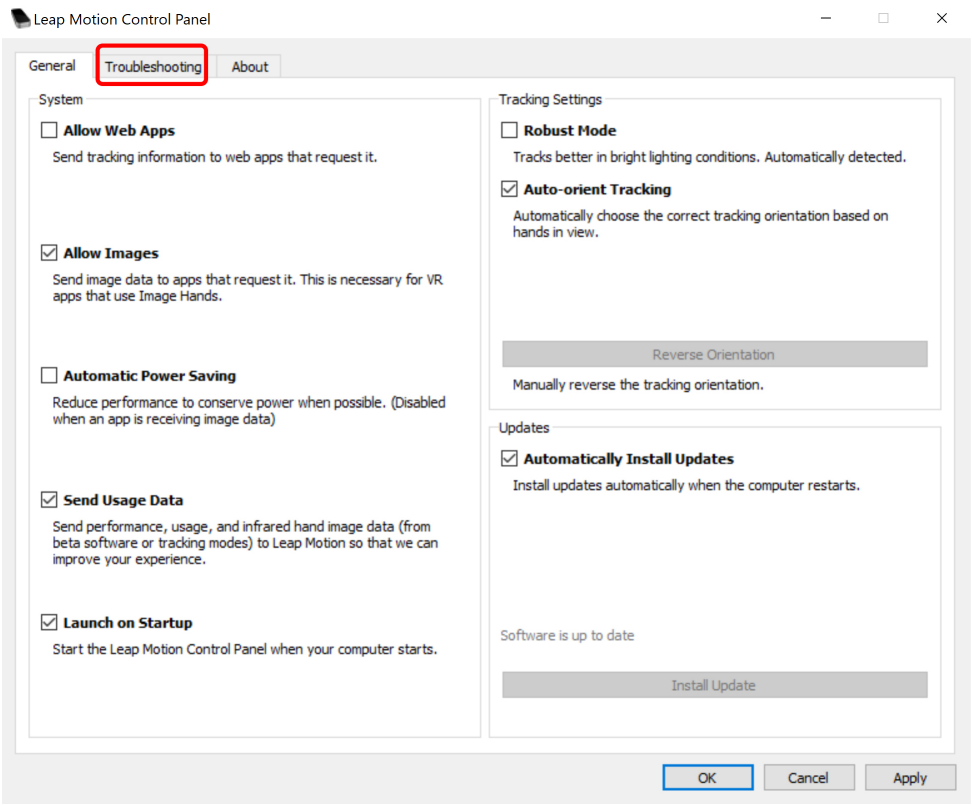


Figure A.10: Clique no botão ”Troubleshooting”

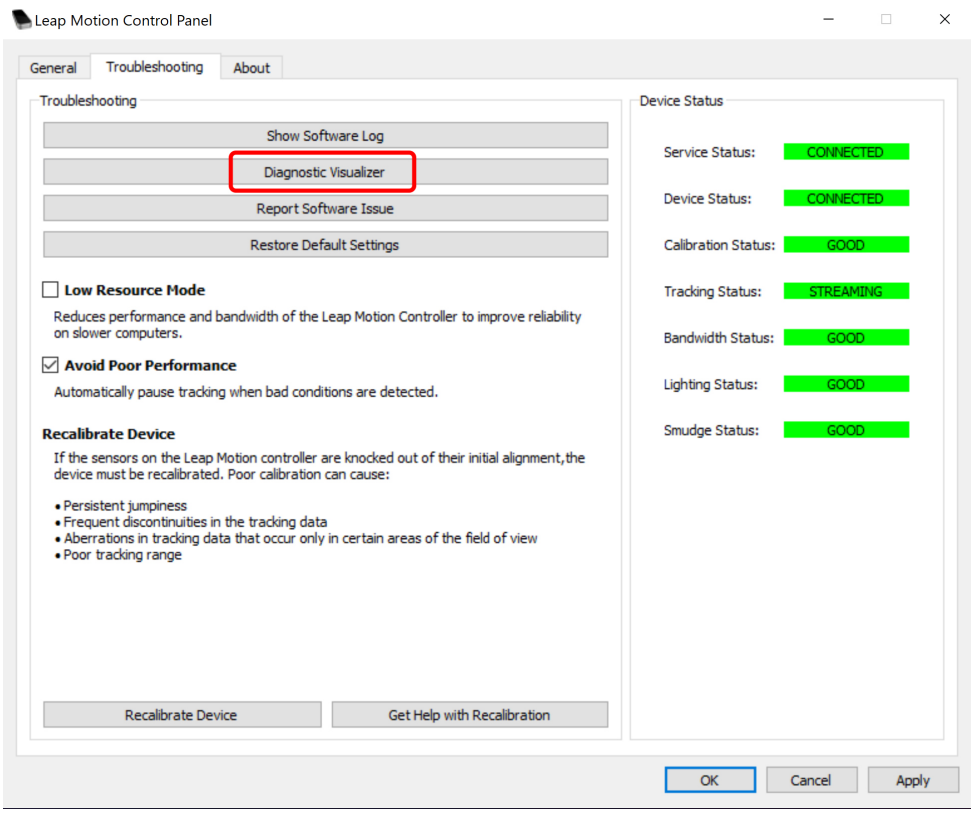


Figure A.11: Clique no botão ”Diagnostic Visualizer”, para confirmar que o LeapMotion controller está a ler a posição das mãos corretamente

Depois deverá colocar as mãos por cima do sensor, exatamente como indicado nas Figuras A.12 e A.13:

Figure A.12: Como colocar as mãos por cima do LeapMotion controller – I

Figure A.13: Como colocar as mãos por cima do LeapMotion controller – II

Se tudo estiver configurado corretamente, os movimentos das suas mãos deverão ser fielmente replicados no ecrã do seu computador, tal como demonstrado na Figura A.14:

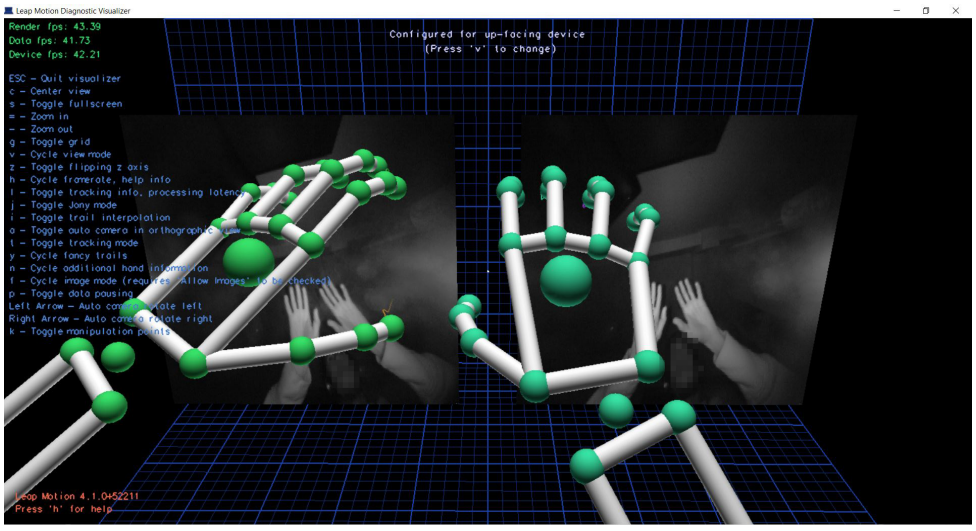


Figure A.14: Representação da posição das mãos no ecrã do computador

Após tudo relativamente ao LeapMotion controller estar funcional, o jogo sério poderá ser instalado. Feche todas as janelas que atualmente tem abertas e execute os seguintes passos:

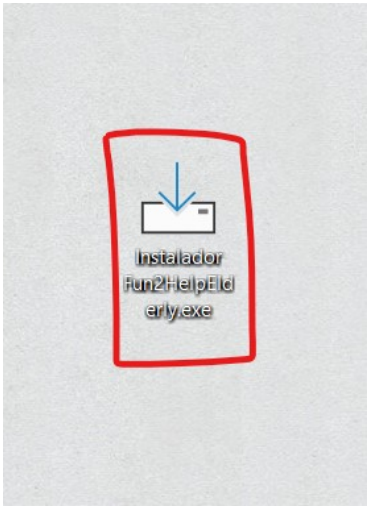


Figure A.15: Clique no ícone do instalador do jogo sério, denomaninado ”Instalador Fun2HelpElderly”

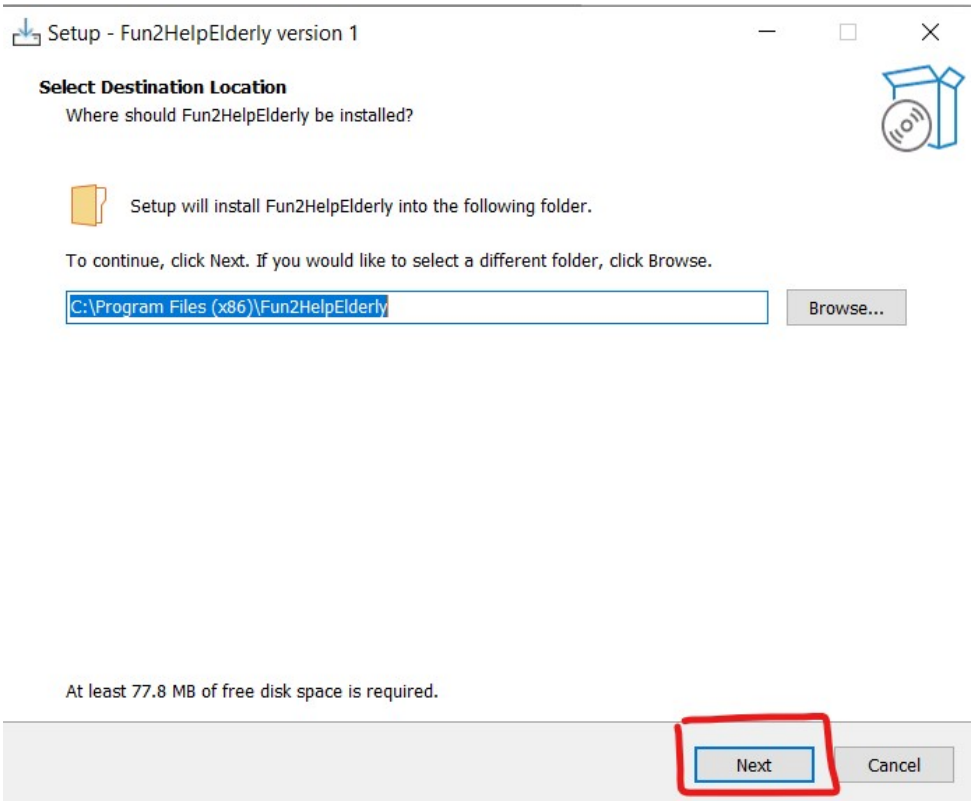


Figure A.16: Escolha a pasta onde pretende instalar o jogo sério. Nesta demonstração, o software será instalado no ambiente trabalho.

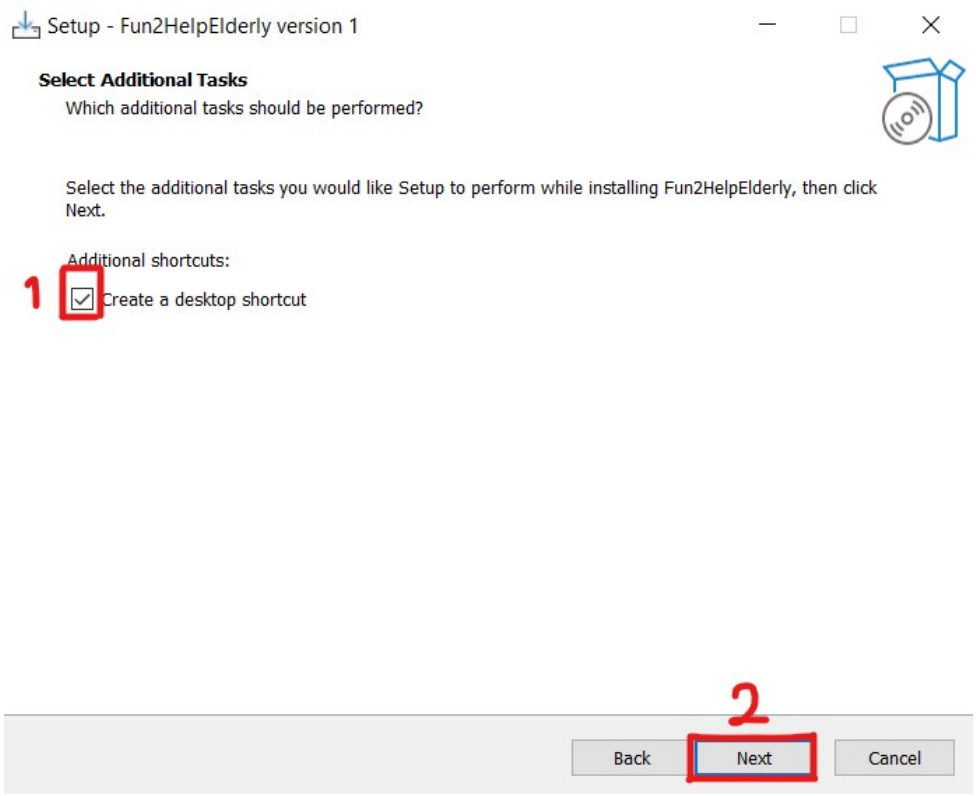


Figure A.17: Selecione a opção ”Create a desktop shortcut” e clique no botão ”Next”

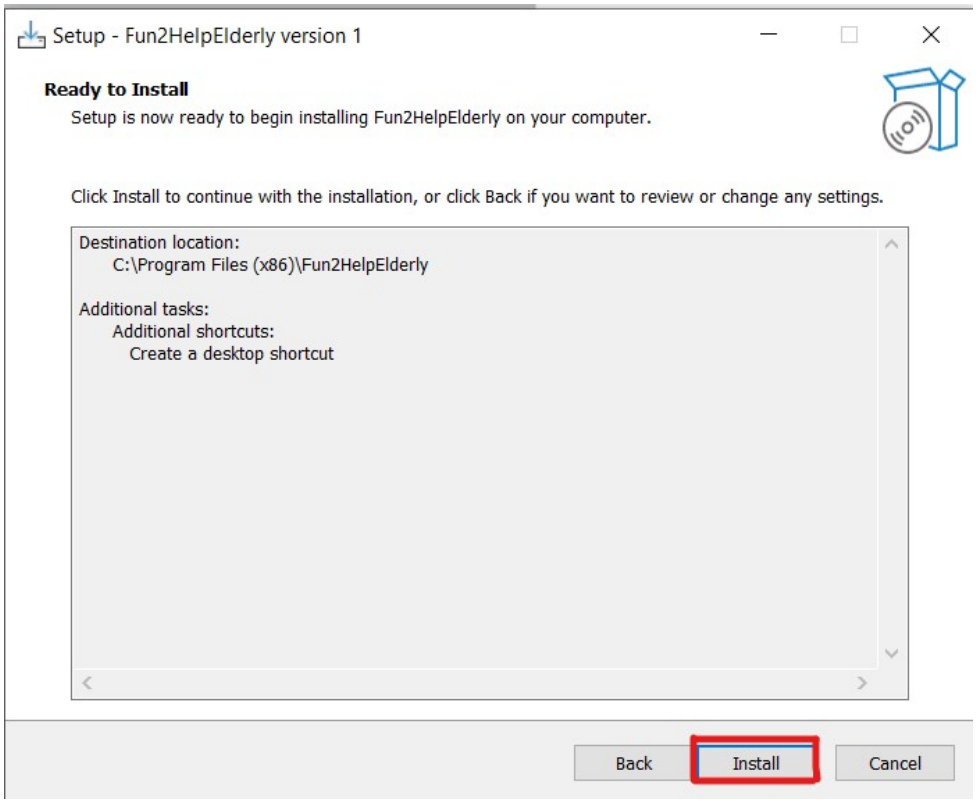


Figure A.18: Clique no botão ”Install”

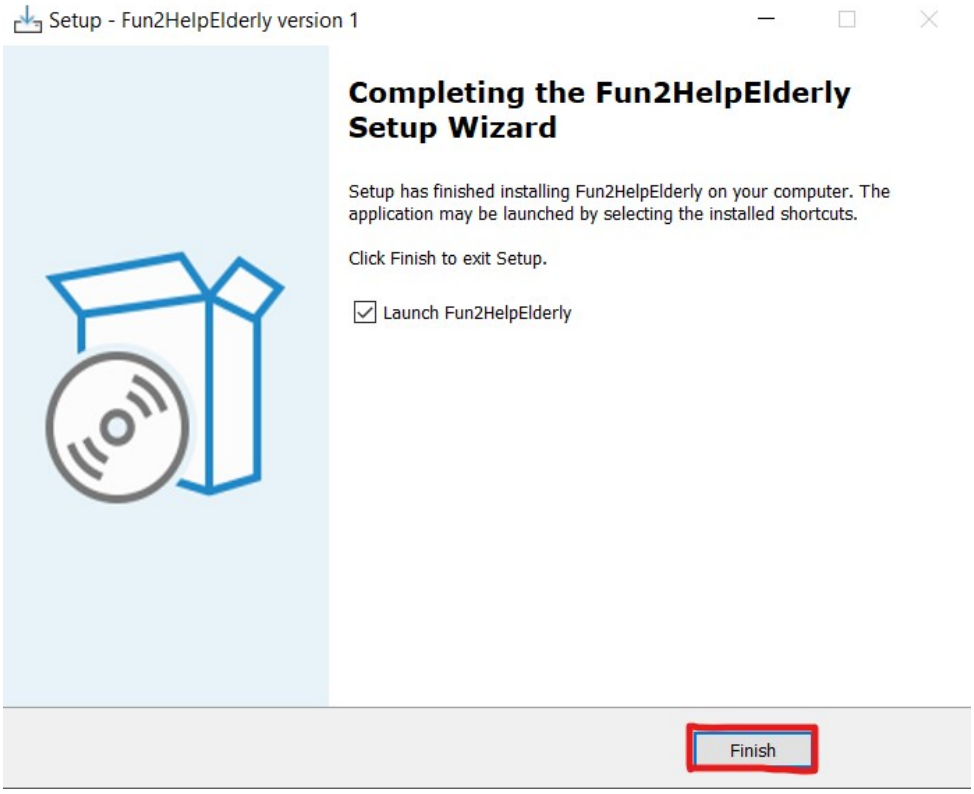


Figure A.19: Deixe a opção ”Launch Fun2HelpElderly” ativada e clique no botão ”Finish”

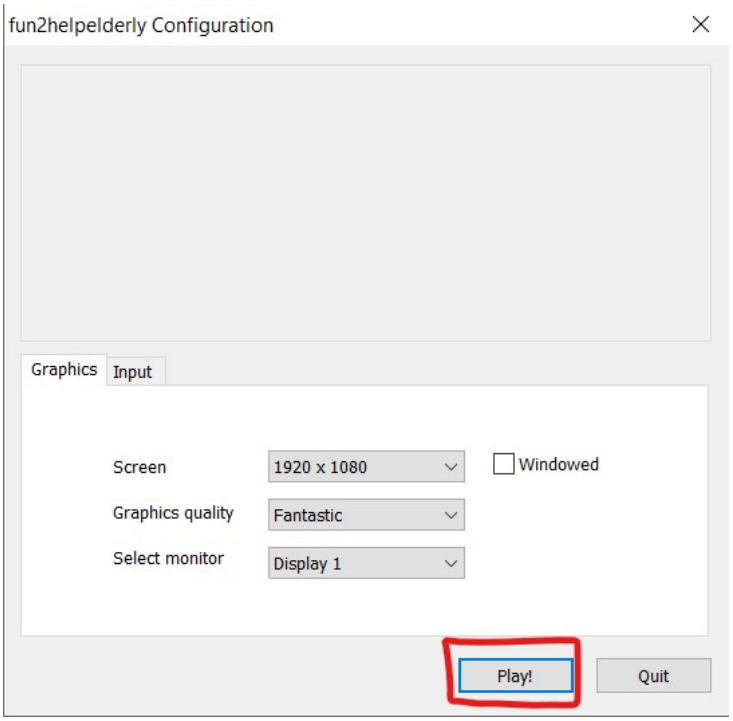


Figure A.20: Definições recomendadas para computadores com especificações de hardware médias-altas

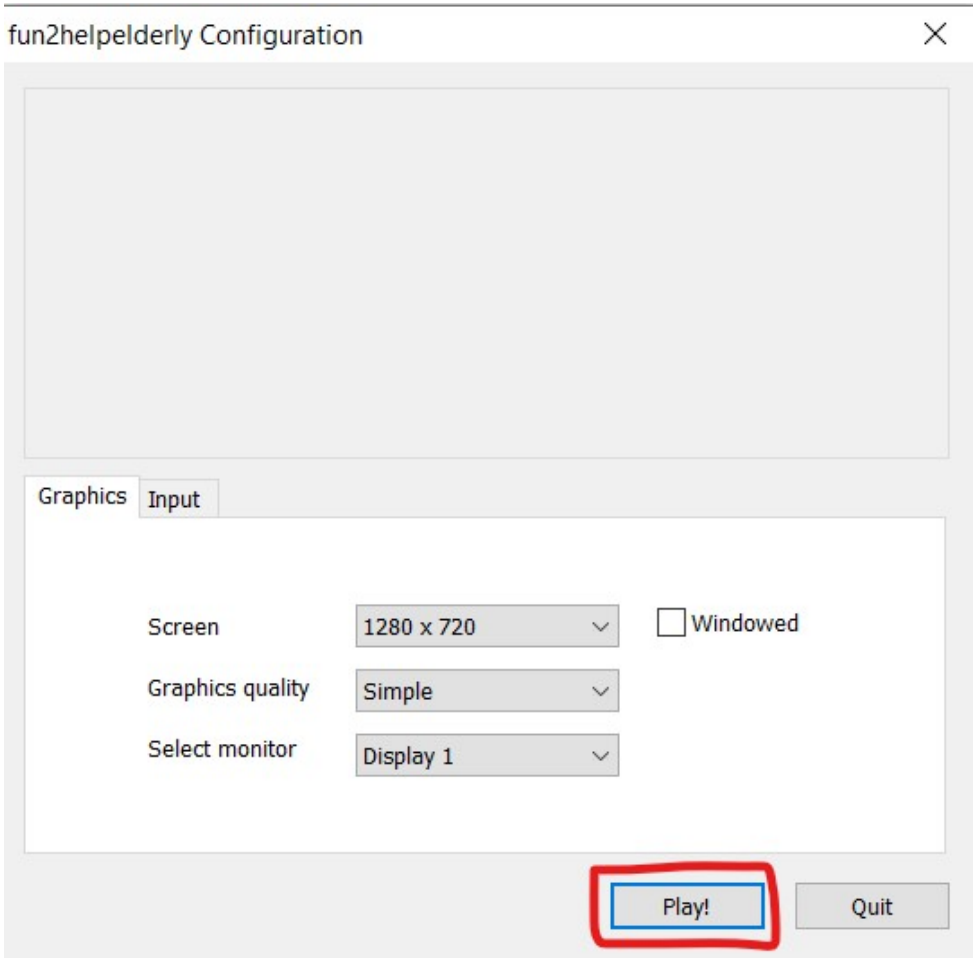


Figure A.21: Definições recomendadas para computadores com especificações de hardware médias-baixas

Após clicar no botão ”Start”, deverá deparar-se com a seguinte janela:

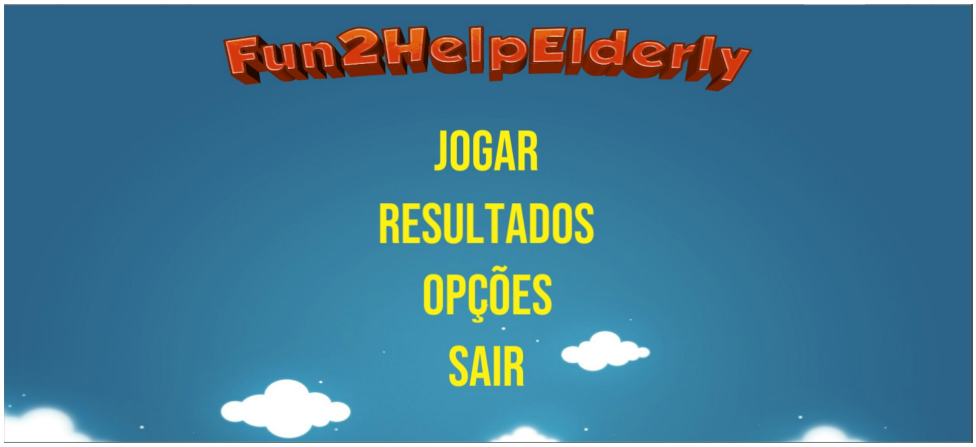


Figure A.22: Menu principal do jogo sério

Antes de começar a jogar, poderá ser conveniente conhecer os sub-menus do jogo sério:

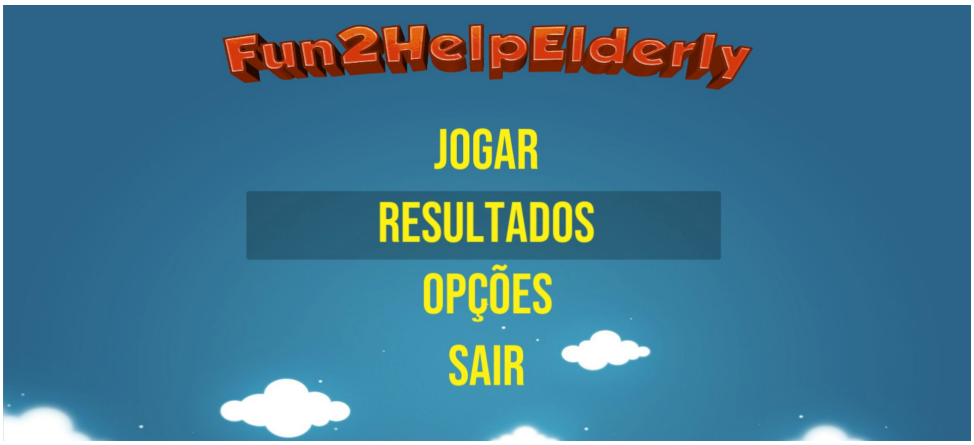


Figure A.23: Estatísticas do jogador – I

Figure A.24: Estatísticas do jogador II – Aqui deve poder visualizar as últimas pontuações guardadas pelo jogador. Neste caso a tabela aparece vazia pois nenhuma pontuação foi guardada até ao momento

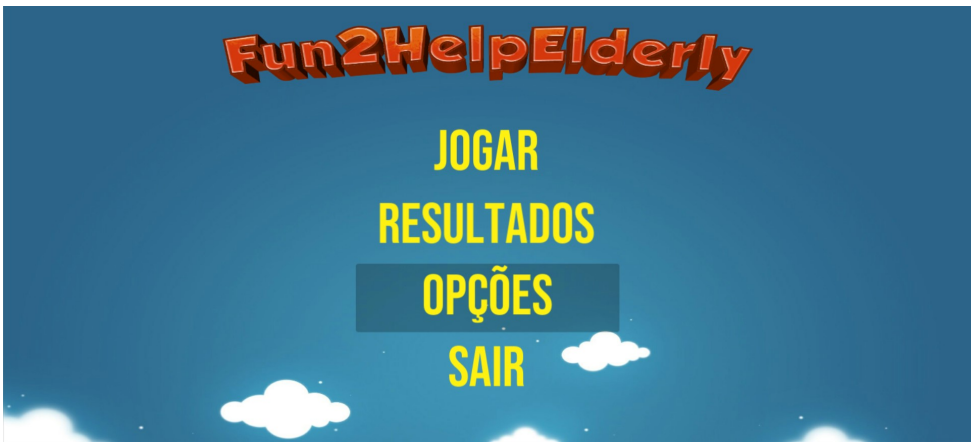


Figure A.25: Opções – I

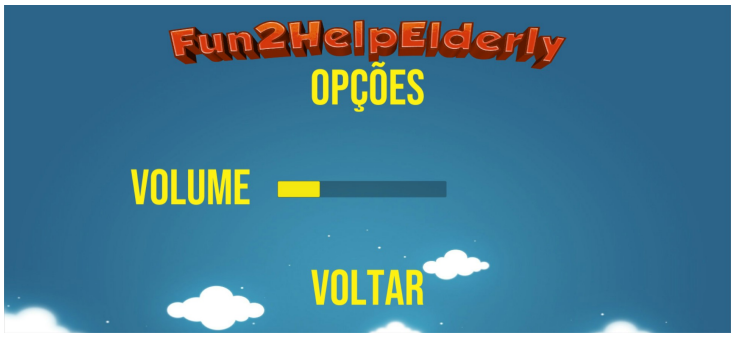


Figure A.26: Opções – II. Aqui deve poder ajustar o volume da música de fundo

Clicando no botão ”Play”, o jogo sério deverá começar, com as respetivas instruções na parte superior do ecrã, tal como se pode visualizar na Figura A.27:

Figure A.27: Primeiro cenário do jogo sério – Aquecimento para o sub-jogo número 1

B

Annex B - Reference manual

Fun2HelpTheElderly

Generated by Doxygen 1.9.1

Chapter 1

Hierarchical Index

1.1 Class Hierarchy

This inheritance list is sorted roughly, but not completely, alphabetically:

Chapter 2

Class Index

2.1 Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

Chapter 3

File Index

3.1 File List

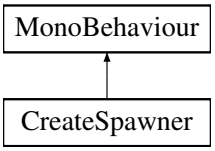
Here is a list of all files with brief descriptions:

Chapter 4

Class Documentation

4.1 CreateSpawner Class Reference

Inheritance diagram for CreateSpawner:



Private Member Functions

- void Start ()
- void Update ()

Private Attributes

- GameObject goCreate1
- GameObject goCreate2
- GameObject goCreate3
- GameObject goCreate4
- GameObject goCreate5
- GameObject goCreateAux
- int decideObject
- float fTimeIntervals
- Vector3 v3SpawnPosJitter
- float fTimer = 0

4.1.1 Member Function Documentation

4.1.1.1 Start()

```
void CreateSpawner.Start ( )  [private]
```

4.1.1.2 Update()

```
void CreateSpawner.Update ( )  [private]
```

4.1.2 Member Data Documentation

4.1.2.1 decideObject

```
int CreateSpawner.decideObject  [private]
```

4.1.2.2 fTimeIntervals

```
float CreateSpawner.fTimeIntervals  [private]
```

4.1.2.3 fTimer

```
float CreateSpawner.fTimer = 0  [private]
```

4.1.2.4 goCreate1

```
GameObject CreateSpawner.goCreate1  [private]
```

4.1.2.5 goCreate2

```
GameObject CreateSpawner.goCreate2  [private]
```

4.1.2.6 goCreate3

```
GameObject CreateSpawner.goCreate3  [private]
```

4.1.2.7 goCreate4

```
GameObject CreateSpawner.goCreate4  [private]
```

4.1.2.8 goCreate5

```
GameObject CreateSpawner.goCreate5  [private]
```

4.1.2.9 goCreateAux

```
GameObject CreateSpawner.goCreateAux  [private]
```

4.1.2.10 v3SpawnPosJitter

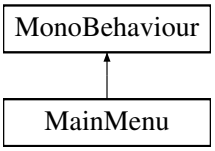
```
Vector3 CreateSpawner.v3SpawnPosJitter  [private]
```

The documentation for this class was generated from the following file:

- C:/Users/jddso/OneDrive/Documentos/fun2helpelderly/Assets/Scripts/CreateSpawner.cs

4.2 MainMenu Class Reference

Inheritance diagram for MainMenu:



Public Member Functions

- void PlayGame ()
- void BackToMenu ()
- void QuitGame ()

4.2.1 Member Function Documentation

4.2.1.1 BackToMenu()

```
void MainMenu.BackToMenu ( )
```

4.2.1.2 PlayGame()

```
void MainMenu.PlayGame ( )
```

4.2.1.3 QuitGame()

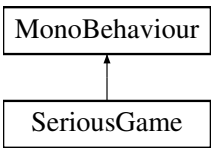
```
void MainMenu.QuitGame ( )
```

The documentation for this class was generated from the following file:

- C:/Users/jddso/OneDrive/Documentos/fun2helpelderly/Assets/Scripts/MainMenu.cs

4.3 SeriousGame Class Reference

Inheritance diagram for SeriousGame:



Static Public Attributes

- static int score

  Serious game score public variable.
- static int time

  Serious game time public variable.

4.3.1 Member Data Documentation

4.3.1.1 score

```
int SeriousGame.score  [static]
```

Serious game score public variable.

4.3.1.2 time

```
int SeriousGame.time  [static]
```

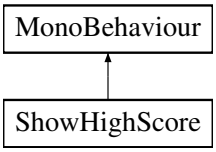
Serious game time public variable.

The documentation for this class was generated from the following file:

- C:/Users/jddso/OneDrive/Documentos/fun2helpelderly/Assets/Scripts/SeriousGame.cs

4.4 ShowHighScore Class Reference

Inheritance diagram for ShowHighScore:



Public Attributes

- string[ ] txtNamesNscore
- List< string > names = new List<string>()
- List< string > scores = new List<string>()

Private Member Functions

- void OnEnable ()
- void ReadAndShowScore ()

Private Attributes

- Text text_aux

4.4.1 Member Function Documentation

4.4.1.1 OnEnable()

```
void ShowHighScore.OnEnable ( )  [private]
```

4.4.1.2 ReadAndShowScore()

```
void ShowHighScore.ReadAndShowScore ( )  [private]
```

4.4.2 Member Data Documentation

4.4.2.1 names

```
List<string> ShowHighScore.names = new List<string>()
```

4.4.2.2 scores

```
List<string> ShowHighScore.scores = new List<string>()
```

4.4.2.3 text_aux

```
Text ShowHighScore.text_aux  [private]
```

4.4.2.4 txtNamesNscore

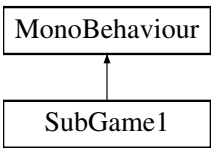
```
string [] ShowHighScore.txtNamesNscore
```

The documentation for this class was generated from the following file:

- C:/Users/jddso/OneDrive/Documentos/fun2helpelderly/Assets/Scripts/ShowHighScore.cs

4.5 SubGame1 Class Reference

Inheritance diagram for SubGame1:



Public Member Functions

- void OnCollisionEnter (Collision Collision)

  Function that allows to detect eventual collisions.

Public Attributes

- GameObject uiObject_right

  game objects to show confirmation messages
- GameObject uiObject_wrong

Private Member Functions

- void start ()

  hides confirmation messages
- IEnumerator WaitForSec_right ()

  show confirmation message for 1 second
- IEnumerator WaitForSec_wrong ()

  show confirmation message for 1 second

Private Attributes

- string tag_right

  Tag to verify is the right move was performed.
- string tag_wrong

  Tag to verify is the wrong move was performed.

4.5.1 Member Function Documentation

4.5.1.1 OnCollisionEnter()

```
void SubGame1.OnCollisionEnter (
            Collision Collision )
```

Function that allows to detect eventual collisions.

If the collision was the expected one, the system destoys the colliding object, display a confirmation message, enters WaitForSec function and increases score

If the collision was the expected one, the system destoys the colliding object, display a confirmation message, enters WaitForSec function and increases score

4.5.1.2 start()

```
void SubGame1.start ( )  [private]
```

hides confirmation messages

4.5.1.3 WaitForSec_right()

```
IEnumerator SubGame1.WaitForSec_right ( )  [private]
```

show confirmation message for 1 second

4.5.1.4 WaitForSec_wrong()

```
IEnumerator SubGame1.WaitForSec_wrong ( )  [private]
```

show confirmation message for 1 second

4.5.2 Member Data Documentation

4.5.2.1 tag_right

```
string SubGame1.tag_right  [private]
```

Tag to verify is the right move was performed.

4.5.2.2 tag_wrong

```
string SubGame1.tag_wrong  [private]
```

Tag to verify is the wrong move was performed.

4.5.2.3 uiObject_right

```
GameObject SubGame1.uiObject_right
```

game objects to show confirmation messages

4.5.2.4 uiObject_wrong

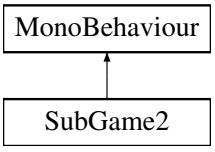
```
GameObject SubGame1.uiObject_wrong
```

The documentation for this class was generated from the following file:

- C:/Users/jddso/OneDrive/Documentos/fun2helpelderly/Assets/Scripts/SubGame1.cs

4.6 SubGame2 Class Reference

Inheritance diagram for SubGame2:



Public Member Functions

- void OnCollisionEnter (Collision Collision)

  Function that allows to detect eventual collisions.

Public Attributes

- GameObject textDisplay

  game objects to show the right color
- string stringAux

Private Member Functions

- void Start ()

  Initialization of the time interval and string responsible for showing the right color on the screen.
- void Update ()

Private Attributes

- string bola_branca

  Tag to verify is the wrong move was performed.
- string bola_roxa

  Tag to verify is the wrong move was performed.
- string bola_preta

  Tag to verify is the wrong move was performed.
- string bola_verde

  Tag to verify is the wrong move was performed.
- string bola_azul

  Tag to verify is the wrong move was performed.
- string bola_vermelha

  Tag to define the color changing intervals.
- float fTimeIntervals

  Tag to verify is the wrong move was performed.
- float fTimer = 0

  float number to control the color string changing process
- int color = 0

  integer number to be randomly sorted to change the colors

4.6.1 Member Function Documentation

4.6.1.1 OnCollisionEnter()

```
void SubGame2.OnCollisionEnter (
            Collision Collision )
```

Function that allows to detect eventual collisions.

If the collision was the expected one, user's score is increased. If it was not the expected one, user's score is decreased

4.6.1.2 Start()

```
void SubGame2.Start ( )  [private]
```

Initialization of the time interval and string responsible for showing the right color on the screen.

4.6.1.3 Update()

```
void SubGame2.Update ( )  [private]
```

Random color genarator algorithm

4.6.2 Member Data Documentation

4.6.2.1 bola_azul

```
string SubGame2.bola_azul  [private]
```

Tag to verify is the wrong move was performed.

4.6.2.2 bola_branca

```
string SubGame2.bola_branca  [private]
```

Tag to verify is the wrong move was performed.

4.6.2.3 bola_preta

```
string SubGame2.bola_preta  [private]
```

Tag to verify is the wrong move was performed.

4.6.2.4 bola_roxa

```
string SubGame2.bola_roxa  [private]
```

Tag to verify is the wrong move was performed.

4.6.2.5 bola_verde

```
string SubGame2.bola_verde  [private]
```

Tag to verify is the wrong move was performed.

4.6.2.6 bola_vermelha

```
string SubGame2.bola_vermelha  [private]
```

Tag to define the color changing intervals.

4.6.2.7 color

```
int SubGame2.color = 0  [private]
```

integer number to be randomly sorted to change the colors

4.6.2.8 fTimeIntervals

```
float SubGame2.fTimeIntervals  [private]
```

Tag to verify is the wrong move was performed.

4.6.2.9 fTimer

```
float SubGame2.fTimer = 0  [private]
```

float number to control the color string changing process

4.6.2.10 stringAux

```
string SubGame2.stringAux
```

4.6.2.11 textDisplay

```
GameObject SubGame2.textDisplay
```

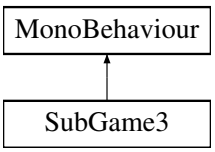
game objects to show the right color

The documentation for this class was generated from the following file:

- C:/Users/jddso/OneDrive/Documentos/fun2helpelderly/Assets/Scripts/SubGame2.cs

4.7 SubGame3 Class Reference

Inheritance diagram for SubGame3:



Public Member Functions

- void OnCollisionEnter (Collision Collision)

  Function that allows to detect eventual collisions.
- IEnumerator waitfunction ()

  waits one second before destroying the desired object

Private Member Functions

- void Start ()

  initial score value is store before the game begins
- void Update ()

Private Attributes

- string strTag_aux

  collision trigger
- string strTag

  collision trigger
- string strTag2

  collision trigger
- string strTag3

  collision trigger
- string strTag4

  collision trigger
- int aux_score

  integer number responsible for storing score value before the subgame begins

4.7.1 Member Function Documentation

4.7.1.1 OnCollisionEnter()

```
void SubGame3.OnCollisionEnter (
            Collision Collision )
```

Function that allows to detect eventual collisions.

If the collision was the expected one, the system waits 1 second and destroys the colliding object

If the collision was the expected one, user's score is increased

If the collision was the expected one, the system waits 1 second and destroys the colliding object

4.7.1.2 Start()

```
void SubGame3.Start ( )  [private]
```

initial score value is store before the game begins

4.7.1.3 Update()

```
void SubGame3.Update ( )  [private]
```

if the total score obtained in this subgame is bigger than 8 it means all cylinders were taken down and loads next scene

4.7.1.4 waitfunction()

```
IEnumerator SubGame3.waitfunction ( )
```

waits one second before destroying the desired object

4.7.2 Member Data Documentation

4.7.2.1 aux_score

```
int SubGame3.aux_score  [private]
```

integer number responsible for storing score value before the subgame begins

4.7.2.2 strTag

```
string SubGame3.strTag  [private]
```

collision trigger

4.7.2.3 strTag2

```
string SubGame3.strTag2  [private]
```

collision trigger

4.7.2.4 strTag3

```
string SubGame3.strTag3  [private]
```

collision trigger

4.7.2.5 strTag4

```
string SubGame3.strTag4  [private]
```

collision trigger

4.7.2.6 strTag_aux

```
string SubGame3.strTag_aux  [private]
```

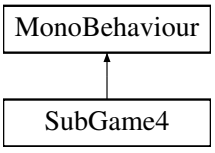
collision trigger

The documentation for this class was generated from the following file:

- C:/Users/jddso/OneDrive/Documentos/fun2helpelderly/Assets/Scripts/SubGame3.cs

4.8 SubGame4 Class Reference

Inheritance diagram for SubGame4:



Public Attributes

- GameObject uiObject_right

  game objects to show confirmation messages
- GameObject uiObject_wrong

Private Member Functions

- void start ()

  hides confirmation messages initially
- void OnCollisionEnter (Collision Collision)

  If the collision was the expected one, the system displays a confirmation message, activates "nextlvl" flag, enters WaitForSec function and increases score.
- IEnumerator WaitForSec_right ()

  show confirmation message for 1 second
- IEnumerator WaitForSec_wrong ()

  show confirmation message for 1 second
- void Update ()

  constantly confirmates if the "nextlvl" is activated.

Private Attributes

- string tag_right

  Tag to verify is the right move was performed.
- string tag_wrong

  Tag to verify is the wrong move was performed.
- float fTimer = 0

  Sets the timer to 0 to control new scene loading timming.
- bool nextlvl = false

  Tag to verify is the wrong move was performed.

4.8.1 Member Function Documentation

4.8.1.1 OnCollisionEnter()

```
void SubGame4.OnCollisionEnter (
            Collision Collision )  [private]
```

If the collision was the expected one, the system displays a confirmation message, activates "nextlvl" flag, enters WaitForSec function and increases score.

If the collision was not the expected one, the system displays a confirmation message, activates "nextlvl" flag, enters WaitForSec function and decreases score

4.8.1.2 start()

```
void SubGame4.start ( )  [private]
```

hides confirmation messages initially

4.8.1.3 Update()

```
void SubGame4.Update ( )  [private]
```

constantly confirmates if the "nextlvl" is activated.

if it is, the system wais 3 seconds and then loads the next scene

4.8.1.4 WaitForSec_right()

```
IEnumerator SubGame4.WaitForSec_right ( )  [private]
```

show confirmation message for 1 second

4.8.1.5 WaitForSec_wrong()

```
IEnumerator SubGame4.WaitForSec_wrong ( )  [private]
```

show confirmation message for 1 second

4.8.2 Member Data Documentation

4.8.2.1 fTimer

```
float SubGame4.fTimer = 0  [private]
```

Sets the timer to 0 to control new scene loading timming.

4.8.2.2 nextlvl

```
bool SubGame4.nextlvl = false  [private]
```

Tag to verify is the wrong move was performed.

4.8.2.3 tag_right

```
string SubGame4.tag_right  [private]
```

Tag to verify is the right move was performed.

4.8.2.4 tag_wrong

```
string SubGame4.tag_wrong  [private]
```

Tag to verify is the wrong move was performed.

4.8.2.5 uiObject_right

```
GameObject SubGame4.uiObject_right
```

game objects to show confirmation messages

4.8.2.6 uiObject_wrong

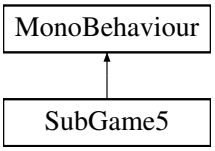
```
GameObject SubGame4.uiObject_wrong
```

The documentation for this class was generated from the following file:

- C:/Users/jddso/OneDrive/Documentos/fun2helpelderly/Assets/Scripts/SubGame4.cs

4.9 SubGame5 Class Reference

Inheritance diagram for SubGame5:



Public Member Functions

- void OnCollisionEnter (Collision Collision)

  Function that allows to detect eventual collisions.

Public Attributes

- GameObject uiObject_right

  game objects to show confirmation messages
- GameObject uiObject_wrong

Private Member Functions

- void start ()

  hides confirmation messages
- IEnumerator WaitForSec_right ()

  show confirmation message for 1 second
- IEnumerator WaitForSec_wrong ()

  show confirmation message for 1 second

Private Attributes

- string tag_right

  Tag to verify is the right move was performed.
- string tag_wrong

  Tag to verify is the wrong move was performed.

4.9.1 Member Function Documentation

4.9.1.1 OnCollisionEnter()

```
void SubGame5.OnCollisionEnter (
            Collision Collision )
```

Function that allows to detect eventual collisions.

If the collision was the expected one, the system destoys the colliding object, display a confirmation message, enters WaitForSec function and increases score

If the collision was the expected one, the system destoys the colliding object, display a confirmation message, enters WaitForSec function and increases score

4.9.1.2 start()

```
void SubGame5.start ( )  [private]
```

hides confirmation messages

4.9.1.3 WaitForSec_right()

```
IEnumerator SubGame5.WaitForSec_right ( )  [private]
```

show confirmation message for 1 second

4.9.1.4 WaitForSec_wrong()

```
IEnumerator SubGame5.WaitForSec_wrong ( )  [private]
```

show confirmation message for 1 second

4.9.2 Member Data Documentation

4.9.2.1 tag_right

```
string SubGame5.tag_right  [private]
```

Tag to verify is the right move was performed.

4.9.2.2 tag_wrong

```
string SubGame5.tag_wrong  [private]
```

Tag to verify is the wrong move was performed.

4.9.2.3 uiObject_right

```
GameObject SubGame5.uiObject_right
```

game objects to show confirmation messages

4.9.2.4 uiObject_wrong

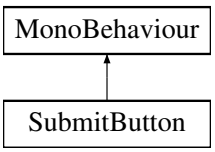
```
GameObject SubGame5.uiObject_wrong
```

The documentation for this class was generated from the following file:

- C:/Users/jddso/OneDrive/Documentos/fun2helpelderly/Assets/Scripts/SubGame5.cs

4.10 SubmitButton Class Reference

Inheritance diagram for SubmitButton:



Public Member Functions

- void SubmitName ()

Private Member Functions

- string CreateText ()

Private Attributes

- string name
- InputField aux_name
- string _path

4.10.1 Member Function Documentation

4.10.1.1 CreateText()

```
string SubmitButton.CreateText ( )  [private]
```

4.10.1.2 SubmitName()

```
void SubmitButton.SubmitName ( )
```

4.10.2 Member Data Documentation

4.10.2.1 _path

```
string SubmitButton._path  [private]
```

4.10.2.2 aux_name

```
InputField SubmitButton.aux_name  [private]
```

4.10.2.3 name

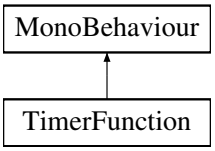
```
string SubmitButton.name  [private]
```

The documentation for this class was generated from the following file:

- C:/Users/jddso/OneDrive/Documentos/fun2helpelderly/Assets/Scripts/SubmitButton.cs

4.11 TimerFunction Class Reference

Inheritance diagram for TimerFunction:



Public Attributes

- GameObject textDisplay
- int secondsLeft = 70
- bool takingAway = false

Private Member Functions

- void Start ()
- void Update ()
- IEnumerator TimerTake ()

4.11.1 Member Function Documentation

4.11.1.1 Start()

```
void TimerFunction.Start ( )  [private]
```

4.11.1.2 TimerTake()

```
IEnumerator TimerFunction.TimerTake ( )  [private]
```

4.11.1.3 Update()

```
void TimerFunction.Update ( )  [private]
```

4.11.2 Member Data Documentation

4.11.2.1 secondsLeft

```
int TimerFunction.secondsLeft = 70
```

4.11.2.2 takingAway

```
bool TimerFunction.takingAway = false
```

4.11.2.3 textDisplay

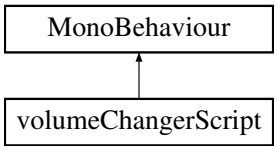
```
GameObject TimerFunction.textDisplay
```

The documentation for this class was generated from the following file:

- C:/Users/jddso/OneDrive/Documentos/fun2helpelderly/Assets/Scripts/TimerFunction.cs

4.12 volumeChangerScript Class Reference

Inheritance diagram for volumeChangerScript:



Private Member Functions

- void OnEnable ()
- void OnDisable ()
- IEnumerator MyUpdate ()

Private Attributes

- Slider SliderHandleValue
- AudioSource musicVolumeController

4.12.1 Member Function Documentation

4.12.1.1 MyUpdate()

```
IEnumerator volumeChangerScript.MyUpdate ( )  [private]
```

4.12.1.2 OnDisable()

```
void volumeChangerScript.OnDisable ( )  [private]
```

4.12.1.3 OnEnable()

```
void volumeChangerScript.OnEnable ( )  [private]
```

4.12.2 Member Data Documentation

4.12.2.1 musicVolumeController

```
AudioSource volumeChangerScript.musicVolumeController  [private]
```

4.12.2.2 SliderHandleValue

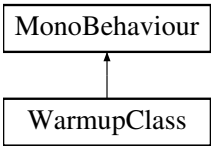
```
Slider volumeChangerScript.SliderHandleValue  [private]
```

The documentation for this class was generated from the following file:

- C:/Users/jddso/OneDrive/Documentos/fun2helpelderly/Assets/Scripts/volumeChangerScript.cs

4.13 WarmupClass Class Reference

Inheritance diagram for WarmupClass:



Public Member Functions

- void OnCollisionEnter (Collision Collision)

  Function that allows to detect eventual collisions.
- IEnumerator WaitForSec ()

Private Attributes

- string strTag

4.13.1 Member Function Documentation

4.13.1.1 OnCollisionEnter()

```
void WarmupClass.OnCollisionEnter (
            Collision Collision )
```

Function that allows to detect eventual collisions.

If the collision was the expected one, the system enters WaitForSec function

4.13.1.2 WaitForSec()

```
IEnumerator WarmupClass.WaitForSec ( )
```

The system waits 2 seconds

The system loads the next scene

4.13.2 Member Data Documentation

4.13.2.1 strTag

```
string WarmupClass.strTag  [private]
```

The documentation for this class was generated from the following file:

- C:/Users/jddso/OneDrive/Documentos/fun2helpelderly/Assets/Scripts/WarmupClass.cs

Chapter 5

File Documentation

5.1 C:/Users/jddso/OneDrive/Documentos/fun2helpelderly/Assets/↩ Scripts/CreateSpawner.cs File Reference

Classes

- class CreateSpawner

5.2 C:/Users/jddso/OneDrive/Documentos/fun2helpelderly/Assets/↩ Scripts/MainMenu.cs File Reference

Classes

- class MainMenu

5.3 C:/Users/jddso/OneDrive/Documentos/fun2helpelderly/Assets/↩ Scripts/SeriousGame.cs File Reference

Classes

- class SeriousGame

5.4 C:/Users/jddso/OneDrive/Documentos/fun2helpelderly/Assets/↩ Scripts/ShowHighScore.cs File Reference

Classes

- class ShowHighScore

5.5 C:/Users/jddso/OneDrive/Documentos/fun2helpelderly/Assets/↩ Scripts/SubGame1.cs File Reference

Classes

- class SubGame1

5.6 C:/Users/jddso/OneDrive/Documentos/fun2helpelderly/Assets/↩ Scripts/SubGame2.cs File Reference

Classes

- class SubGame2

5.7 C:/Users/jddso/OneDrive/Documentos/fun2helpelderly/Assets/↩ Scripts/SubGame3.cs File Reference

Classes

- class SubGame3

5.8 C:/Users/jddso/OneDrive/Documentos/fun2helpelderly/Assets/↩ Scripts/SubGame4.cs File Reference

Classes

- class SubGame4

5.9 C:/Users/jddso/OneDrive/Documentos/fun2helpelderly/Assets/↩ Scripts/SubGame5.cs File Reference

Classes

- class SubGame5

5.10 C:/Users/jddso/OneDrive/Documentos/fun2helpelderly/Assets/↩ Scripts/SubmitButton.cs File Reference

Classes

- class SubmitButton

5.11 C:/Users/jddso/OneDrive/Documentos/fun2helpelderly/Assets/↩ Scripts/TimerFunction.cs File Reference

Classes

- class TimerFunction

5.12 C:/Users/jddso/OneDrive/Documentos/fun2helpelderly/Assets/↩ Scripts/volumeChangerScript.cs File Reference

Classes

- class volumeChangerScript

5.13 C:/Users/jddso/OneDrive/Documentos/fun2helpelderly/Assets/↩ Scripts/WarmupClass.cs File Reference

Classes

- class WarmupClass

Index

Repository Documentation

1) Download Unity version 5.6.1 (https://unity3d.com/pt/get-unity/download/archive).
2) Download Leap Motion SDK (https://developer.leapmotion.com/sdk-leap-motion-controller).
3) Connect the Leap Motion device to a USB port on your computer.
4) Open an existing project and select the folder “fun2helpelderly”.
5) Open the folder “Game Scenes” (in column “Project”) and select the desired game scenery.
   (Note 1: Ideally select the menu.)
6) Click on the button “Play” at the top of Unity’s window.

   (Note 2: The scripts related to the game implementation are in folder path "fun2helpelderly\Assets\Scripts" or in the folder “Scripts” in column “Project”.)

7) The background music can be found in the folder “fun2helpelderly\Assets\Audio”
8) The used font can be found in the folder path “fun2helpelderly\Assets\Fonts”
9) The used textures can be found in the folder path “fun2helpelderly\Assets\Textures”
10) The “.txt” file containing the saved scores can be found in “fun2helpelderly\Assets\” with the name of “scores.txt”
11) Every game object, as well as camera angles and panels can be found in the Hierarchy tab, in the Unity top right corner.

C

Annex C - Detailed project documentation

DADOS PESSOAIS - DECLARAÇÃO DE CONSENTIMENTO

O Regulamento Geral sobre a Proteção de Dados Pessoais (RGPD), em vigor desde o dia 25 de maio de 2018, estabelece regras relativamente à proteção, tratamento e livre circulação dos dados pessoais das pessoas singulares, mesmo que tenham sido recolhidos antes daquela data, e que se aplica diretamente a todas as entidades que procedam ao tratamento desses dados.

De forma utilizar os seus dados pessoais para conclusões experimentais e potencial publicação científica das mesmas, necessitamos do seu consentimento, que deve ser livre, explícito, inequívoco e informado, que pode ser confirmado assinalando as seguintes opções:

▢ obtenção de informação pessoal do utente (nome, idade) e respetiva aptidão física e mental para jogar o jogo;

▢ elaboração de processos de inquérito e estudos de monitorização e avaliação de resultados do jogo;

▢ obtenção de conteúdo audiovisual (fotografias, vídeos) durante o tempo em que se encontrar a experimentar o jogo sério, com o objetivo de ilustrar a investigação realizada, em dissertações e teses académicas e em artigos científicos com ela relacionados, e apenas para estes tipos de divulgação científica;

▢ não dou consentimento a nenhuma das opções anteriores.

Os seus dados serão guardados de acordo com as imposições legais, nomeadamente, respeitando os prazos de conservação arquivística, neste caso particular, até à data de agosto de 2022.

Poderá contactar-nos para qualquer questão relacionada com a proteção dos seus dados, através dos seguintes endereços de email: joao.sousa@isr.uc.pt, rprocha@isr.uc.pt.

Enquanto titular dos seus dados pessoais, pode solicitar o acesso aos mesmos, alterá-los e limitar parcial ou totalmente a sua utilização.

___ /___/2022

O Titular dos Dados:

_____

Instituto de Sistemas e Robótica - Universidade de Coimbra

Fun2HelpElderly - Questionário de usabilidade

1 - Achei o jogo interessante.

1 (Discordo Totalmente)	2	3	4	5	6	7 (Muito)

2 - Achei o jogo fácil de jogar.

1 (Discordo Totalmente)	2	3	4	5 (Concordo Totalmente)

3 - Gostaria de jogar o jogo regularmente daqui em diante.

1 (Discordo Totalmente)	2	3	4	5 (Concordo Totalmente)

4 - O tempo passou depressa enquanto jogava o jogo.

1 (De modo algum)	2	3	4	5	6	7 (Muito)

5 - A experiência de jogo fez com que ficasse completamente concentrado no que estava a fazer.

1 (De modo algum)	2	3	4	5	6	7 (Muito)

6 - O jogo contém informações desnecessárias.

1 (Discordo Totalmente)	2	3	4	5 (Concordo Totalmente)

7 - Percebi sempre o que tinha de fazer em todos os momentos do jogo.

1 (De modo algum)	2	3	4	5	6	7 (Muito)

8 - Eu precisaria de ajuda técnica para conseguir jogar o jogo regularmente.

1 (Discordo Totalmente)	2	3	4	5 (Concordo Totalmente)

9 - Considero que o que tenho que fazer no jogo não faz sentido para pessoas da minha idade.

1 (Discordo Totalmente)	2	3	4	5 (Concordo Totalmente)

10 - Enquanto jogava, senti vontade de dar o meu melhor, de forma a obter a pontuação máxima.

1 (De modo algum)	2	3	4	5	6	7 (Muito)

11 - Enquanto jogava, tive receio de não estar a fazer as coisas bem.

1 (De modo algum)	2	3	4	5	6	7 (Muito)

12 - O jogo estimulou-me a nível físico ou mental.

1 (Discordo Totalmente)	2	3	4	5 (Concordo Totalmente)

13 - Penso que a maioria das pessoas da minha idade conseguiriam aprender facilmente a jogar este jogo.

1 (Discordo Totalmente)	2	3	4	5 (Concordo Totalmente)

14 - Acho o jogo cansativo.

1 (Discordo Totalmente)	2	3	4	5 (Concordo Totalmente)

15 - Senti-me confiante durante a utilização do jogo.

1 (Discordo Totalmente)	2	3	4	5 (Concordo Totalmente)

16 - Tive de aprender coisas novas para conseguir experimentar este jogo.

1 (Discordo Totalmente)	2	3	4	5 (Concordo Totalmente)

17 - Considero que o jogo tem uma dificuldade de:

1 (Muito fácil)	2	3	4	5	6	7 (Muito díficil)

Instituto de Sistemas e Robótica - Universidade de Coimbra

Fun2HelpElderly - Questionário de usabilidade (escala FSS)

1 - Achei o jogo interessante.

1 (Discordo Totalmente)	2	3	4	5	6	7 (Muito)

2 - O tempo passou depressa enquanto jogava o jogo.

1 (De modo algum)	2	3	4	5	6	7 (Muito)

3 - A experiência de jogo fez com que ficasse completamente concentrado no que estava a fazer.

1 (De modo algum)	2	3	4	5	6	7 (Muito)

4 - Percebi sempre o que tinha de fazer em todos os momentos do jogo.

1 (De modo algum)	2	3	4	5	6	7 (Muito)

5 - Senti vontade de não falhar e obter a máxima pontuação possível enquanto jogava.

1 (De modo algum)	2	3	4	5	6	7 (Muito)

6 - Tive receio de falhar enquanto tentava atingir os objetivos do jogo.

1 (De modo algum)	2	3	4	5	6	7 (Muito)

7 - Considero que o jogo tem uma dificuldade de:

1 (Muito fácil)	2	3	4	5	6	7 (Muito díficil)

Instituto de Sistemas e Robótica - Universidade de Coimbra

Fun2HelpElderly - Questionário de usabilidade (Escala SUS)

1 - Achei o jogo fácil de jogar.

1 (Discordo Totalmente)	2	3	4	5 (Concordo Totalmente)

2 - Gostaria de jogar o jogo regularmente daqui em diante.

1 (Discordo Totalmente)	2	3	4	5 (Concordo Totalmente)

3 - O jogo contém informações desnecessárias.

1 (Discordo Totalmente)	2	3	4	5 (Concordo Totalmente)

4 - Eu precisaria de ajuda técnica para conseguir jogar o jogo regularmente.

1 (Discordo Totalmente)	2	3	4	5 (Concordo Totalmente)

5 - Considero que o que tenho que fazer no jogo não faz sentido para pessoas da minha idade.

1 (Discordo Totalmente)	2	3	4	5 (Concordo Totalmente)

6 - O jogo estimulou-me a nível físico ou mental.

1 (Discordo Totalmente)	2	3	4	5 (Concordo Totalmente)

7 - Penso que a maioria das pessoas da minha idade conseguiriam aprender facilmente a jogar este jogo.

1 (Discordo Totalmente)	2	3	4	5 (Concordo Totalmente)

8 - Acho o jogo cansativo.

1 (Discordo Totalmente)	2	3	4	5 (Concordo Totalmente)

9 - Senti-me confiante durante a utilização do jogo.

1 (Discordo Totalmente)	2	3	4	5 (Concordo Totalmente)

10 - Tive de aprender coisas novas para conseguir experimentar este jogo.

1 (Discordo Totalmente)	2	3	4	5 (Concordo Totalmente)

D

Annex D - Testing sessions tables

Table D.1: Serious game prototype's functional and performance with a sample of four end-users. The "Time required" column refers to the fastest quest tackling done in each sub-game regarding every session, in seconds.

End-user number	Testing session number	Age	Gender	Score	Time required (s)
1	1	77	M	14	SG1: 5,13 SG2: 4,22 SG3: 6,01 SG4: 2,30 SG5: 4,54
1	2	77	M	20	SG1: 3,88 SG2: 2,10 SG3: 4,50 SG4: 1,67 SG5: 3,78
1	3	77	M	22	SG1: 3,15 SG2: 2,15 SG3: 4,56 SG4: 1,45 SG5: 4,31
1	4	77	M	25	SG1: 2,80 SG2: 2,07 SG3: 5,32 SG4: 1,56 SG5: 4,01
1	5	77	M	24	SG1: 2,95 SG2: 1,89 SG3: 3,80 SG4: 1,43 SG5: 3,56
1	6	77	M	28	SG1: 2,70 SG2: 1,80 SG3: 3,23 SG4: 1,22 SG5: 3,43
2	1	80	F	3	SG1: 25,32 SG2: 6,51 SG3: 15,12 SG4: 2,47 SG5: 2,45
2	2	80	F	7	SG1: 16,24 SG2: 5,29 SG3: 11,98 SG4: 1,54 SG5: 2,11

Continued on next page

Table D.1 – continued from previous page

End-user number	Testing session number	Age	Gender	Score	Time required (s)
2	3	80	F	7	SG1: 15,11 SG2: 5,04 SG3: 10,03 SG4: 1,98 SG5: 2,98
2	4	80	F	8	SG1: 13,24 SG2: 4,27 SG3: 9,50 SG4: 1,73 SG5: 1,70
2	5	80	F	10	SG1: 14,13 SG2: 4,56 SG3: 11,31 SG4: 1,80 SG5: 1,83
2	6	80	F	11	SG1: 13,07 SG2: 4,72 SG3: 8,29 SG4: 1,76 SG5: 1,99
3	1	82	M	9	SG1: 10,56 SG2: 4,83 SG3: 10,76 SG4: 3,01 SG5: 2,01
3	2	82	M	20	SG1: 8,90 SG2: 3,69 SG3: 8,13 SG4: 2,39 SG5: 1,96
3	3	82	M	18	SG1: 6,13 SG2: 4,01 SG3: 9,90 SG4: 2,90 SG5: 1,54
3	4	82	M	27	SG1: 7,01 SG2: 3,70 SG3: 6,12 SG4: 1,70 SG5: 1,76

Continued on next page

Table D.1 – continued from previous page

End-user number	Testing session number	Age	Gender	Score	Time required (s)
3	5	82	M	25	SG1: 6,90 SG2: 2,17 SG3: 7,01 SG4: 1,22 SG5: 1,30
3	6	82	M	25	SG1: 5,39 SG2: 2,12 SG3: 5,32 SG4: 1,40 SG5: 1,56
4	1	79	F	11	SG1: 13,87 SG2: 5,90 SG3: 12,11 SG4: 13,53 SG5: 4,50
4	2	79	F	12	SG1: 11,98 SG2: 5,17 SG3: 11,67 SG4: 14,01 SG5: 3,78
4	3	79	F	9	SG1: 12,59 SG2: 3,65 SG3: 8,54 SG4: 11,47 SG5: 4,57
4	4	79	F	15	SG1: 10,75 SG2: 4,90 SG3: 9,56 SG4: 9,45 SG5: 2,71
4	5	79	F	14	SG1: 8,46 SG2: 4,01 SG3: 9,01 SG4: 10,78 SG5: 3,95
4	6	79	F	16	SG1: 9,76 SG2: 4,12 SG3: 9,12 SG4: 9,93 SG5: 3,13

Table D.2: Serious game eight day testing stage with a sample of eleven end-users. The ”Time required” column refers to the fastest quest tackling done in each sub-game regarding every session, in seconds.

End-user number	Testing session number	Age	Gender	Score	Time required (s)
1	1	74	M	15	SG1: 7,01 SG2: 5,23 SG3: 8,24 SG4: 3,49 SG5: 6,32
1	2	74	M	20	SG1: 5,24 SG2: 4,01 SG3: 7,42 SG4: 3,21 SG5: 4,31
1	3	74	M	19	SG1: 8,52 SG2: 3,05 SG3: 6,22 SG4: 2,24 SG5: 4,51
1	4	74	M	25	SG1: 4,25 SG2: 3,06 SG3: 6,89 SG4: 2,41 SG5: 3,52
1	5	74	M	27	SG1: 3,76 SG2: 3,75 SG3: 5,99 SG4: 2,04 SG5: 4,25
1	6	74	M	24	SG1: 4,61 SG2: 3,01 SG3: 4,75 SG4: 1,53 SG5: 4,99
1	7	74	M	28	SG1: 4,41 SG2: 2,95 SG3: 5,13 SG4: 1,49 SG5: 3,61
1	8	74	M	31	SG1: 5,98 SG2: 2,64 SG3: 6,32 SG4: 1,99 SG5: 3,92

Continued on next page

Table D.2 – continued from previous page

End-user number	Testing session number	Age	Gender	Score	Time required (s)
2	1	68	F	16	SG1: 6,72 SG2: 4,37 SG3: 7,45 SG4: 3,01 SG5: 5,70
2	2	68	F	19	SG1: 3,45 SG2: 3,22 SG3: 6,11 SG4: 2,05 SG5: 4,01
2	3	68	F	26	SG1: 4,01 SG2: 2,90 SG3: 3,72 SG4: 1,27 SG5: 3,75
2	4	68	F	-	-
2	5	68	F	29	SG1: 3,96 SG2: 2,11 SG3: 5,99 SG4: 1,89 SG5: 4,10
2	6	68	F	35	SG1: 2,99 SG2: 1,29 SG3: 3,19 SG4: 1,21 SG5: 3,41
2	7	68	F	29	SG1: 3,01 SG2: 1,72 SG3: 3,21 SG4: 1,22 SG5: 3,85
2	8	68	F	33	SG1: 2,91 SG2: 1,80 SG3: 3,80 SG4: 1,43 SG5: 3,07
3	1	77	M	17	SG1: 6,73 SG2: 4,98 SG3: 9,61 SG4: 3,32 SG5: 6,32

Continued on next page

Table D.2 – continued from previous page

End-user number	Testing session number	Age	Gender	Score	Time required (s)
3	2	77	M	16	SG1: 6,22 SG2: 4,51 SG3: 7,23 SG4: 2,35 SG5: 5,21
3	3	77	M	24	SG1: 5,31 SG2: 4,04 SG3: 5,17 SG4: 1,99 SG5: 6,01
3	4	77	M	-	-
3	5	77	M	23	SG1: 8,32 SG2: 3,45 SG3: 5,26 SG4: 1,61 SG5: 4,51
3	6	77	M	23	SG1: 5,12 SG2: 2,12 SG3: 6,01 SG4: 1,79 SG5: 3,74
3	7	77	M	25	SG1: 5,92 SG2: 2,64 SG3: 5,61 SG4: 1,68 SG5: 4,51
3	8	77	M	28	SG1: 4,50 SG2: 2,15 SG3: 3,61 SG4: 1,8 SG5: 4,69
4	1	63	F	23	SG1: 24,16 SG2: 11,41 SG3: 13,41 SG4: 2,52 SG5: 3,01
4	2	63	F	17	SG1: 27,11 SG2: 9,64 SG3: 11,49 SG4: 1,44 SG5: 3,51

Continued on next page

Table D.2 – continued from previous page

End-user number	Testing session number	Age	Gender	Score	Time required (s)
4	3	63	F	19	SG1: 23,61 SG2: 9,69 SG3: 10,11 SG4: 1,01 SG5: 2,43
4	4	63	F	-	-
4	5	63	F	22	SG1: 21,58 SG2: 11,13 SG3: 9,28 SG4: 2,01 SG5: 2,12
4	6	63	F	29	SG1: 18,53 SG2: 7,32 SG3: 8,24 SG4: 1,42 SG5: 2,55
4	7	63	F	30	SG1: 17,75 SG2: 6,41 SG3: 8,55 SG4: 1,76 SG5: 1,29
4	8	63	F	27	SG1: 13,19 SG2: 5,90 SG3: 7,32 SG4: 1,61 SG5: 1,79
5	1	73	M	6	SG1: 14,51 SG2: 7,23 SG3: 10,43 SG4: 7,21 SG5: 2,52
5	2	73	M	9	SG1: 13,13 SG2: 6,41 SG3: 12,51 SG4: 6,37 SG5: 3,79
5	3	73	M	11	SG1: 20,24 SG2: 5,07 SG3: 13,51 SG4: 4,31 SG5: 2,61

Continued on next page

Table D.2 – continued from previous page

End-user number	Testing session number	Age	Gender	Score	Time required (s)
5	4	73	M	14	SG1: 11,47 SG2: 4,51 SG3: 7,32 SG4: 4,79 SG5: 3,58
5	5	73	M	13	SG1: 9,19 SG2: 3,95 SG3: 9,14 SG4: 3,27 SG5: 1,61
5	6	73	M	16	SG1: 8,17 SG2: 3,92 SG3: 8,84 SG4: 4,52 SG5: 1,14
5	7	73	M	15	SG1: 8,34 SG2: 2,41 SG3: 6,21 SG4: 4,06 SG5: 1,99
5	8	73	M	17	SG1: 7,11 SG2: 4,53 SG3: 7,31 SG4: 4,33 SG5: 1,43
6	1	74	F	5	SG1: 41,21 SG2: 10,12 SG3: 17,11 SG4: 3,14 SG5: 3,48
6	2	74	F	6	SG1: 34,14 SG2: 9,10 SG3: 21,14 SG4: 3,81 SG5: 2.18
6	3	74	F	6	SG1: 42,24 SG2: 9,74 SG3: 16,21 SG4: 2,12 SG5: 2,94
6	4	74	F	-	-

Continued on next page

Table D.2 – continued from previous page

End-user number	Testing session number	Age	Gender	Score	Time required (s)
6	5	74	F	11	SG1: 20,24 SG2: 6,12 SG3: 16,15 SG4: 2,17 SG5: 2,51
6	6	74	F	13	SG1: 23,52 SG2: 7,61 SG3: 17,22 SG4: 2,18 SG5: 2,30
6	7	74	F	14	SG1: 21,11 SG2: 6,56 SG3: 15,51 SG4: 1,59 SG5: 1,97
6	8	74	F	14	SG1: 23,64 SG2: 4,51 SG3: 15,32 SG4: 1,25 SG5: 1,12
7	1	84	M	9	SG1: 39,14 SG2: 11,42 SG3: 20,41 SG4: 2,98 SG5: 5,14
7	2	84	M	11	SG1: 37,19 SG2: 9,45 SG3: 19,53 SG4: 2,52 SG5: 5,71
7	3	84	M	16	SG1: 36,61 SG2: 9,21 SG3: 18,41 SG4: 3,14 SG5: 6,01
7	4	84	M	18	SG1: 26,51 SG2: 7,14 SG3: 17,11 SG4: 2,05 SG5: 3,20

Continued on next page

Table D.2 – continued from previous page

End-user number	Testing session number	Age	Gender	Score	Time required (s)
7	5	84	M	17	SG1: 29,41 SG2: 8,32 SG3: 17,76 SG4: 1,65 SG5: 5,61
7	6	84	M	20	SG1: 30,51 SG2: 8,02 SG3: 17,94 SG4: 1,91 SG5: 4,51
7	7	84	M	16	SG1: 20,82 SG2: 6,35 SG3: 17,01 SG4: 2,08 SG5: 3,67
7	8	84	M	17	SG1: 24,78 SG2: 5,17 SG3: 14,42 SG4: 2,16 SG5: 5,14
8	1	73	F	12	SG1: 5,12 SG2: 5,22 SG3: 8.12 SG4: 3,01 SG5: 5,32
8	2	73	F	19	SG1: 4,52 SG2: 3,69 SG3: 7,83 SG4: 2,52 SG5: 4,60
8	3	73	F	20	SG1: 3,59 SG2: 3,25 SG3: 4,51 SG4: 1,41 SG5: 4,69
8	4	73	F	-	-
8	5	73	F	19	SG1: 4,15 SG2: 3,02 SG3: 4,92 SG4: 1,04 SG5: 4,03

Continued on next page

Table D.2 – continued from previous page

End-user number	Testing session number	Age	Gender	Score	Time required (s)
8	6	73	F	25	SG1: 2,91 SG2: 2,99 SG3: 5,75 SG4: 1,07 SG5: 3,96
8	7	73	F	27	SG1: 2,25 SG2: 3,07 SG3: 4,61 SG4: 1,45 SG5: 3,36
8	8	73	F	27	SG1: 2,01 SG2: 3,51 SG3: 4,03 SG4: 1,31 SG5: 3,43
9	1	93	F	3	SG1: 29,21 SG2: 8,32 SG3: 16,29 SG4: 3,42 SG5: 2,89
9	2	93	F	2	SG1: 21,95 SG2: 6,11 SG3: 14,14 SG4: 2,81 SG5: 2,53
9	3	93	F	6	SG1: 17,86 SG2: 6,74 SG3: 9,87 SG4: 2,02 SG5: 2,13
9	4	93	F	-	-
9	5	93	F	5	SG1: 17,11 SG2: 4,53 SG3: 8,95 SG4: 2,56 SG5: 1,94
9	6	93	F	10	SG1: 13,71 SG2: 4,67 SG3: 10,22 SG4: 2,05 SG5: 1,32

Continued on next page

Table D.2 – continued from previous page

End-user number	Testing session number	Age	Gender	Score	Time required (s)
9	7	93	F	9	SG1: 11,09 SG2: 4,90 SG3: 11,54 SG4: 1,67 SG5: 1,63
9	8	93	F	12	SG1: 9,18 SG2: 3,99 SG3: 10,31 SG4: 1,56 SG5: 1,53
10	1	81	M	1	SG1: 56,21 SG2: 13,32 SG3: 20,53 SG4: 5,62 SG5: 4,51
10	2	81	M	2	SG1: 47,21 SG2: 10,61 SG3: 19,42 SG4: 4,21 SG5: 3,29
10	3	81	M	0	SG1: 34,32 SG2: 11,62 SG3: 17,45 SG4: 3,54 SG5: 4,01
10	4	81	M	5	SG1: 24,54 SG2: 9,69 SG3: 16,24 SG4: 4.10 SG5: 3,41
10	5	81	M	7	SG1: 21,75 SG2: 7,79 SG3: 17,29 SG4: 4,23 SG5: 3,05
10	6	81	M	6	SG1: 19,23 SG2: 6,16 SG3: 16,78 SG4: 3,53 SG5: 2,94

Continued on next page

Table D.2 – continued from previous page

End-user number	Testing session number	Age	Gender	Score	Time required (s)
10	7	81	M	12	SG1: 17,29 SG2: 7,32 SG3: 15,11 SG4: 3,45 SG5: 2,51
10	8	81	M	9	SG1: 19,76 SG2: 6,15 SG3: 16,01 SG4: 3,99 SG5: 2,08
11	1	85	F	11	SG1: 13,44 SG2: 6,70 SG3: 13,43 SG4: 6,23 SG5: 4,65
11	2	85	F	16	SG1: 12,12 SG2: 5,24 SG3: 13,01 SG4: 4,95 SG5: 4,78
11	3	85	F	19	SG1: 12,59 SG2: 4,51 SG3: 10,42 SG4: 5,61 SG5: 4,21
11	4	85	F	-	-
11	5	85	F	20	SG1: 9,53 SG2: 4,12 SG3: 9,04 SG4: 4,32 SG5: 2,71
11	6	85	F	25	SG1: 8,46 SG2: 4,41 SG3: 9,23 SG4: 3,59 SG5: 3,01
11	7	85	F	22	SG1: 9,43 SG2: 4,58 SG3: 9,63 SG4: 3,75 SG5: 3,24

Continued on next page

Table D.2 – continued from previous page

End-user number	Testing session number	Age	Gender	Score	Time required (s)
11	8	85	F	24	SG1: 9,01 SG2: 4,14 SG3: 9,51 SG4: 4,01 SG5: 3,64

This page intentionally left blank.