PM

Design and Development of RehabMarket
An inspiring interactive experience for motor and cognitive rehabilitation after stroke

MASTER'S DEGREE PROJECT

Guilherme Spínola Freitas
MASTER IN INFORMATICS ENGINEERING

UNIVERSIDADE da MADEIRA
A Nossa Universidade
www.uma.pt

February | 2023

Design and Development of RehabMarket
An inspiring interactive experience for motor and cognitive rehabilitation after stroke

MASTER'S DEGREE PROJECT

Guilherme Spínola Freitas
MASTER IN INFORMATICS ENGINEERING

ORIENTATION
Sergi Bermudez I Badia

CO-ORIENTATION
Teresa da Conceição Madureira Paulino

Acknowledgments

I would like to express my gratitude to my supervisors, Prof. Teresa Paulino and Prof. Sergi Bermudez i Badia, for all the tips, feedback and support during the project's development.

Like all the work that was done in the NeuroRehabLab laboratory, I want to thank the group belonging to it since they made me feel free to use any material existing in the laboratory that was necessary to develop the work. In particular, I want to thank Rodrigo Lima, who helped me understand how CAVE worked and was always ready to provide any needed help.

I also want to thank all the people who volunteered to participate in the user tests because, with their opinions and results, it is possible to assess the degree of success of RehabMarket.

To finish, I want to leave a special thanks to the people who accompanied me from the beginning of this adventure, from friends of mine to, in particular, my family, who always supported me in the good times and those in which I faced more difficulties, where they always had a word of encouragement and motivation.

This page was intentionally left blank

Abstract

Nowadays, there are many cases of people who have suffered a stroke and need rehabilitation to recover motor and cognitive functions. Thus, this project intended to develop a virtual reality (VR) activity that stimulates people who suffered a stroke to stimulate their physical and mental skills.

Virtual Reality is an exciting tool to help in rehabilitation, mainly because it is a more fun and motivating way of exercising. This document introduces the steps towards designing, prototyping and testing the RehabMarket, a CAVE-based activity targeting stroke survivors that allows stimulating limbs' movements while challenging their cognitive skills.

The development of the RehabMarket began with a focus group session involving several participants. This session aimed at gathering feedback and insights to inform the design and development of the game. Once the game was fully developed, user testing was conducted with 11 participants to ensure that it successfully achieved its intended goals. Results have shown that RehabMarket was a very useful tool to help in rehabilitation of people who are recovering from stroke.

Keywords: rehabilitation, stroke, CAVE, Virtual Reality, exercise, therapy, exergaming

This page was intentionally left blank

Resumo

Hoje em dia, há muitos casos de pessoas que sofreram um AVC e precisam de reabilitação para recuperar as funções motoras e cognitivas. Assim, com este projeto, pretendeu-se desenvolver uma atividade de realidade virtual (VR) que permita às pessoas que sofreram um AVC estimular as suas capacidades físicas e mentais.

A Realidade Virtual é uma ferramenta muito interessante para ajudar na reabilitação, principalmente por ser uma forma mais divertida e motivadora de exercitar. Neste documento são apresentados os passos realizados para o design, prototipagem e avaliação do RehabMarket, uma atividade baseada em CAVE direcionada a sobreviventes de AVC que permite estimular os movimentos dos membros enquanto desafia as suas habilidades cognitivas.

O desenvolvimento do RehabMarket começou com uma sessão de focus group, que envolveu vários participantes. O objetivo desta sessão foi recolher feedback e opiniões que pudessem informar o design e desenvolvimento do jogo. Uma vez que o jogo foi totalmente desenvolvido, foram realizados testes de utilizador com 11 participantes para garantir que o mesmo foi bem-sucedido em alcançar os objetivos pretendidos. Os resultados mostraram que o RehabMarket foi uma ferramenta muito útil para ajudar na reabilitação de pessoas que estão a recuperar de AVC.

Palavras-chave: reabilitação, AVC, CAVE, Realidade Virtual, exercício, terapia, exergaming

This page was intentionally left blank

Table of Contents

This page was intentionally left blank

1. Introduction

Nowadays, stroke has become a prevalent health issue, leading to the need for rehabilitation to recover limb movement, whether upper or lower. Therefore, this project aims to develop a technological activity that enables individuals undergoing post-stroke rehabilitation to improve their physical and mental capacities.

To obtain feedback and evaluate the success of the developed system, three crucial questions need to be answered at the end of the process:

1. Is the RehabMarket fully functional without user errors?
2. How is the game compared to the real-life activity of going to the supermarket?
3. How do participants evaluate their experience?

This document will provide information on the central theme of stroke, including its various types, factors leading to its occurrence, and common sequelae. Additionally, existing systems designed to assist post-stroke rehabilitation will be presented. To gather more opinions on the subject and identify various methods that could aid in rehabilitation, a focus group was conducted, whose results will also be presented in this document. After the completion of the focus group session, the implementation phase was initiated, which will be explicitly described in the paper along with the results of the user tests conducted at the end.

This document will also present my project idea and the justification for its implementation. As previously noted, individuals who experience a stroke often suffer from lifelong consequences, such as post-stroke depression (PSD), visual impairment, and others. Of all the sequelae, hemiparesis and hemiplegia stand out as the most common. As previously explained, these conditions significantly impact individuals' mobility.

Hemiplegia is a type of paralysis that affects an entire side of the body, completely impeding mobility. Another condition, hemiparesis, impedes the ability to move a part of the body but does not make it impossible. Both of these conditions represent the most frequent types of sequelae in individuals who have suffered a stroke, accounting for approximately 50% of cases.

For the selection of the project idea, two reasons served as motivation. First, rehabilitation is a critical factor for the well-being of individuals who have suffered a stroke. It is essential to improve their physical and mental condition, minimize damage, and regain independence. Second, the possibility of training in everyday activities is crucial. It is estimated that 26% of the 795,000 people who have suffered a stroke cannot perform daily activities, which significantly affects their independence. With proper stimulation and rehabilitation of the upper and lower limbs, it may be possible to perform desired activities of daily living (ADLs) without assistance.

When selecting this idea, it is essential to take into account the limitations observed in other researched systems. Previous studies have shown that users expressed dissatisfaction with the environment and graphics of the activities, which this project aims to address by creating a supermarket environment that is familiar and easily recognizable.

After a stroke, individuals may experience difficulties in performing daily activities, making rehabilitation essential for their recovery. While various rehabilitation activities exist, the majority of them are in the form of games. RehabMarket combines two concepts by providing a gamified activity that facilitates training of both motor and cognitive abilities in a familiar environment - the supermarket - that is closely related to daily activities.

2. State of the Art

In modern times, numerous diseases result in severe consequences, some of which are fatal. Worldwide, vascular problems are the leading cause of death, with stroke being the second most common. Together, these conditions caused the death of 15.2 million individuals in 2015 [1]. Stroke is also among the leading causes of long-term disability, primarily affecting the elderly population, which is more susceptible to the condition's incidence. Among 795,000 individuals who experience a first-time stroke, 26% cannot perform basic daily activities, and 50% experience reduced mobility due to hemiparesis, a condition characterized by partial paralysis on one side of the body [1]. With the advent of new therapies, the number of individuals who survive a stroke and continue to live with its consequences has increased worldwide. In the United States alone, 85% of stroke survivors are alive, and approximately four million individuals live with stroke sequelae [2].

Different stroke types can be identified using computed tomography (CT) or magnetic resonance imaging (MRI) scans. Although CT scans are commonly used for stroke diagnostics, MRIs provide more detailed and accurate information and allow for faster differentiation between bleeding and thrombosis [3].

Strokes can be categorized into two types: ischemic and hemorrhagic [1], [2], [3]. Ischemic stroke occurs due to an interruption of blood supply to a part of the brain, resulting in a sudden loss of functions [3]. It is the most common type of stroke, accounting for 80% of cases, although this varies by population [2]. A study in 22 countries supports this notion, with the authors concluding that Africa had 66% of hemorrhagic strokes and 34% of ischemic strokes, while more developed countries had 91% of ischemic strokes and 9% of hemorrhagic strokes [3]. Although it is the most common type, it is the least fatal [1].

There are several subtypes of ischemic strokes based on the TOAST (Trial of ORG 10172 in Acute Stroke Treatment) criteria, including cardioembolic, atherosclerotic, lacunar, other causes (genetics, vasculitis, dissections, etc.), and strokes with unknown causes [2].

Table 1 - Types of ischemic stroke based on toast classification [2]

Stroke type	Causes	Percentage
Large artery thrombotic strokes	Atherosclerotic plaques in the large blood vessels	20%
Small penetrating artery thrombotic stroke	Vessels are affected	25%
Cardiogenic embolic stroke	Valvular heart disease	15%
Cryptogenic stroke	Cause is unknown	5 – 10%
Stroke associated with other causes	Illicit drug use	20 – 25%

It is called a hemorrhagic stroke when there is a rupture of a blood vessel or an abnormal vascular structure. This type of stroke is less common, accounting for about 20% of cases. However, it results in a higher number of deaths. There are two types of hemorrhagic strokes: intracerebral and subarachnoid hemorrhage [2], [3].

Firstly, intracerebral hemorrhage is the most common non-traumatic intracranial hemorrhage, accounting for 80% of all hemorrhagic strokes. It is usually caused by uncontrolled hypertension that leads to the rupture of small blood vessels. This rupture can lead to the breakage of nearby vessels, resulting in the expansion of the hematoma in up to 40% of cases.

On the other hand, subarachnoid hemorrhage occurs mainly due to saccular aneurysms, arteriovenous malformation, intracranial neoplasia, and consumption of some anticoagulants. About 65% of patients with subarachnoid hemorrhage survive, but half of them are left with severe cognitive deficits, making it a highly dangerous type of hemorrhage [2], [3].

While anyone can suffer a stroke at any age and time, the chances of having a stroke increase with certain risk factors. Some of these risks can be controlled, while others cannot. These are known as modifiable and non-modifiable risk factors, respectively (see Table 2).

Table 2 Major nonmodifiable and modifiable stroke risk factors [3]

Nonmodifiable Risk Factors	Modifiable Risk Factors
Age	Hypertension
Gender	Diabetes
Race/Ethnicity	Physical Activity
Genetics	Alcohol consumption and smoking

2.1 Non-modifiable Risk Factors

Firstly, it is a well-established fact that the incidence of stroke increases with age, with the incidence doubling after 55 years of age, and rising by 25% among adults aged 20 to 64 in recent decades [1]. In 2005, the mean age of ischemic stroke was reported as 69 years, but the incidence of ischemic stroke has increased in age rates between 20 and 54 years, from 12.9% in 1993/1994 to 18.6% in 2005 [2].

The gender relationship with stroke is age-dependent. Women have a higher risk of suffering a stroke related to pregnancy and postpartum status in younger generations, while at older ages, the likelihood is higher for men. In general, stroke occurs more frequently in women than in men, which can be attributed to the higher life expectancy of women when compared to men [2]. The World Health Organization (WHO) reported a significantly higher number of stroke-related deaths in women than in men between 1990 and 2006, of which 60% were over the age of 75 [1].

Black people have twice the risk of stroke compared to whites and have a higher incidence of stroke-related deaths. Hispanics/Latinos also have a higher risk rate of stroke incidence. This discrepancy between ethnicities is primarily due to differences in access to healthcare, which is poorer in some areas compared to others [2]. Genetics also play an important role, as people with a family history of stroke are more likely to experience one [2].

2.2 Modifiable Risk Factors

One of the most significant modifiable risk factors for stroke is hypertension, as there exists a strong and direct association between blood pressure levels and stroke risk. Elevated blood pressure levels increase the likelihood of experiencing a stroke [2]. Individuals with diabetes also face a significantly increased risk of stroke, with stroke accounting for approximately 20% of diabetes-related deaths. Nonetheless, medical therapy and behavioral modifications, when employed together, can aid in reducing the risk of stroke in individuals with diabetes [2]. Insufficient physical activity has been linked to a variety of health concerns, including a higher incidence of stroke. Individuals who engage in regular physical activity are less likely to experience a stroke [2]. Moreover, alcohol consumption is associated with the occurrence of a particular type of stroke, known as haemorrhagic stroke. Alcohol

consumption can cause both high and low blood pressure, both of which can increase the likelihood of experiencing a stroke [2].

2.3 Sequelae

Many individuals who have survived a stroke often experience psychological and neuropsychiatric problems that can significantly impact their daily routines and activities [4]. These issues, referred to as sequelae, encompassing a range of conditions, such as Post Stroke Depression, Post Stroke Language Disorders, Visual Impairments, Hemiplegia, and Hemiparesis.

Post Stroke Depression is the most common mental health issue affecting stroke survivors, with an estimated prevalence of 33%. This condition negatively impacts the recovery of motor and cognitive deficits and increases the likelihood of experiencing other neurovascular events. Treatment with medication can improve quality of life, but the selection of medication should be based on the viability and reaction profile of individual patients [5]. Biological and psychological factors play a significant role in the development of Post Stroke Depression:

Table 3 - Biological and psychological factors in PSD [5]

Biological Factors	Psychological factors
Age	Social isolation
Gender	Low self-esteem
Alterations in neurotrophic factors	Stress
Disruption of neural networks	Dependability

As previously mentioned, stroke is associated with a variety of language disorders, including aphasia, alexia, agraphia, and acalculia [6].

- Aphasia is a language impairment that results from brain dysfunction and manifests as difficulties in verbal communication, comprehension of written or spoken language, repetition, and reading. It is a common consequence of lesions in the left hemisphere and is the most significant neuropsychological consequence of stroke.
- Alexia, on the other hand, causes the loss of the ability to read and it is divided into three categories: pure alexia with agraphia, pure alexia without agraphia, and aphasic alexia.
- Agraphia is defined as the loss of writing ability and can occur independently or be associated with aphasia or alexia.
- Acalculia, as the name suggests, affects mathematical ability and causes difficulty in mental or paper-based calculations.

In addition to language disorders, stroke survivors often experience visual impairments, including abnormalities in peripheral and central vision, eye movements, and other perception problems. These visual impairments can be especially dangerous if the eye or cortex is damaged. Visual impairment can significantly impact the performance of activities of daily living (ADLs) and the person's independence and overall quality of life [7].

Finally, hemiplegia and hemiparesis are common consequences of stroke. Hemiplegia is characterized by the loss of mobility in certain parts of the body, resulting in paralysis. Hemiparesis, on the other hand, makes movement of specific parts of the body more difficult without completely paralyzing them. The reduction of motor capacities due to hemiplegia and hemiparesis is the most significant factor affecting the performance of ADLs and socialization, especially when the damage occurs in the upper body. Therefore, it is essential to train affected areas to regain the ability to perform daily activities [8].

2.4 Stroke Rehabilitation

Currently, rehabilitation is recognized as an essential factor in recovering from physical or psychological injury. Traditional methods, such as exercises using weights and medicinal balls, remain the most widely used technique in clinical practice [16]. However, the evolution of technology has played an increasingly important role in home-based rehabilitation, improving patients' motor skills, providing equivalent rehabilitation quality as conventional therapies, enhancing activities of daily living, and providing patients with a sense of control over rehabilitation while offering the convenience of performing rehabilitation at home. Home-based technologies are often viewed as a more engaging and dynamic approach to rehabilitation than conventional therapies [15].

Various technologies are commonly used in home-based rehabilitation, including games, robotic devices, virtual reality, sensors, and tablets, each with its unique advantages. However, these methods also have limitations.

Table 4 - Benefits and limitations of home-based rehabilitation technologies [15]

Technology	Benefits	Limitations
Games	- More engaging and motivating. - Provide flexibility for patients that are dependent on caregivers.	- Less interaction between patients and therapists. - Require technical proficiency to use it
Robotic Devices	- Provide automatic therapies. - Deliver measurable intensity	- Require more space. - Need appropriate facilities for setup.
Virtual Reality	- Safe and controlled environment.	- Difficulty in validating the clinical outcome
Sensors	- Serve as a measuring device to quantify the accuracy of the movements.	- Difficulty in minimising the obtrusiveness of sensors
Tablets	- Commercially available. - Affordable technology.	- It might be challenging for stroke survivors who suffer from visual field loss.

There are various rehabilitation devices available, including the Kinect, a motion-capture device that can track the position of the human body's joints [10]. The natural interaction provided by Kinect makes it a popular tool in rehabilitation systems. For example, a system designed to retrain muscles and rejuvenate debilitated functions has demonstrated promising results for stroke patients [10]. Traditional exercises and rehabilitation can be unpleasant and painful, leading to decreased patient tolerance. Therefore, more natural and interactive ways of rehabilitation, such as using devices like the Kinect, may be more appealing to patients.

The Kinect has the potential to overcome barriers to in-home stroke rehabilitation as an engaging and accurate markerless motion capture tool and controller interface. Research has shown that the Kinect's ability to accurately capture upper extremity movements is sufficient for clinical use regarding elbow and wrist joint tracking. Rehabilitation exercises using the Kinect can be more enjoyable and

dynamic, but there are some limitations associated with the system. Initial Kinect-based comparisons with research-grade motion capture systems demonstrated highly correlated trends and reasonable accuracy, but evaluation studies have focused only on gross movements advantageous for Kinect. Evaluation of more realistic and specific diagnostic movement sets is still needed. The Kinect device is also unable to accurately assess internal joint rotations of the shoulder, and instead utilises a much less clinically viable single-point estimation. Therefore, the use of the Kinect for specific shoulder-based functionality requirements has yet to be shown to be clinically feasible. The system is also not suitable for severely disabled patients, as gross movements that remain extremely small in their entirety are difficult for the Kinect to capture accurately.

The Kinect is also used for full-body tracking and interaction in various software, such as the Cave Automatic Virtual Environment (CAVE) system. The CAVE system is a highly interactive virtual environment where patients can interact with any simulated environment, making it very useful in VR rehabilitation. The Kinect can track a patient's joints and represent it in the virtual world, requiring patients to move and reach into the virtual world to interact with objects and perform their exercises [17].

Virtual Reality (VR) and serious games have also been used for motor neurorehabilitation with stroke patients affected by upper limb paresis. For example, in [9], the authors combined two elements: i) movement observation with the intent to imitate and ii) visualization of the mirrored movements of the non-paretic limb [9]. The first element is based on the observation that mirror neurons (hand actions) discharge during goal-directed hand actions and the observation of another individual performing a similar move. The second element of the rehabilitation concept involves visualizing mirrored movements in the non-paretic arm, using mirrors placed along the center line of the patient's body so that viewing the reflection of the non-paretic arm in a mirror gives the patient the impression that their paretic arm can move [9]. The authors of [9] used a 3D digital compass (Honeywell) for arm position input and an 80 cm wide-screen LCD television for audiovisual output. The patient was seated in a regular chair or wheelchair at a table facing a monitor, with their arms on the table in front of them, showing on the monitor two components precisely with the same orientation and position (Figure 1), so that the real arms are represented as virtual arms on the screen, stimulating the patient to continue the exercise.

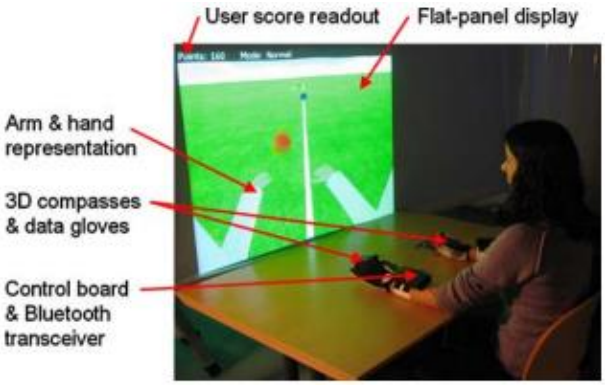


Figure 1 - Overview of the VR therapy system [9]

The authors performed a usability test with 19 healthy subjects aged between 25-36 years old from an academic conference and 5 patients from a children's hospital with various neurological diseases aged between 7-14 years old. Each participant played one or more games for 3 minutes per game. Participants were instructed to engage with the system in three different modes: hitting, catching, and grasping. In the hitting mode, the objective was to intercept virtual balls moving towards the participant. In the catching mode, the participant was required to open and close their hand to intercept the ball. In the grasping mode, the participant had to close their hand to intercept the ball, move it, and then release it at a different position. After completing the testing, participants were asked to provide feedback on their enjoyment of the system, the ease of use of the input devices, and suggestions for improving the

games. Results indicated that participants quickly learned how to use the system and perform the tasks, resulting in very good user acceptance. For the initial settings (one ball every 2.5 seconds, low speed), most participants were able to intercept between 70-100% of the balls. However, participants experienced more difficulties with different mapping parameters. Participants expressed a desire to continue using the system, which shows promise for providing rehabilitation sessions. Participants also requested background music, better graphics, and stronger storylines. The authors identified the need for further work in defining and calibrating standard tests and using the game infrastructure to ensure reproducible results.

Another example is PrimeSense [11], which is a technology consisting of four different games that patients can play to recover and improve their upper limbs. These activities utilized a PrimeSense 'Carmine' camera connected to a laptop via USB with graphics displayed on a TV screen. The camera uses a 3D depth sensor, similar to Kinect, and can detect a range of 0.8 to 3.5 meters. The objective of these games was to test the feasibility of using motion-controlled games in stroke survivors compared to conventional therapies. PrimeSense comprises four games that can be played in both sitting and standing positions. Three of these games are controlled by torso movements, while one is controlled by upper limb movements. As shown in Figure 2, the games include Ball Maze, Fridge Frenzy, Tentacle Dash, and Bubble Fish, each set in a different environment. In Ball Maze, Fridge Frenzy, and Tentacle Dash, the movements of the shoulders and hips are tracked, while in Bubble Fish, the wrist joint is tracked. All games require the patient to move to the right, left, forward, or backward to complete the levels, each with ten levels of increasing difficulty. Bubble Fish is unique in that the patient must use their arms as weapons to attack targets.

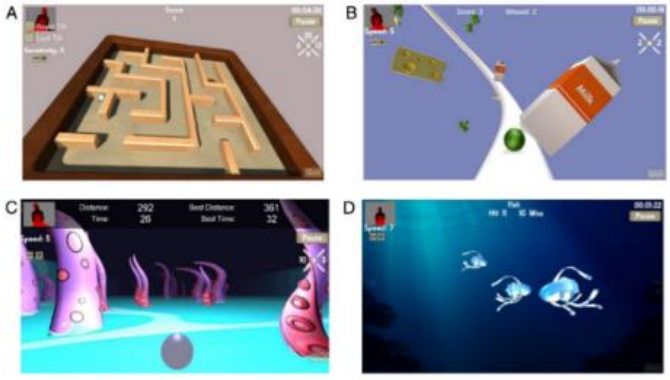


Figure 2 - The 4 different games: a) ball maze, b) fridge frenzy, c) tentacle dash, d) bubble fish [11]

The authors noted some limitations of the system, including the small number of games available in the current development phase, and the need to create more. Additionally, while 40 participants were included in Phase 1, only a small group of 16 participated in Phase 2, with only eight of these participants being evaluated for accessibility and viability. Therefore, more participants are needed to further evaluate the system. While patients enjoyed using the system, some reported experiencing minor pain before and after sessions, and expressed a desire for more challenging games.

In addition to PrimeSense, a CAVE system called Athene Software is also used for rehabilitation. The software is based on the Unity 3D game engine and includes various virtual environments with a free run option for users to explore the area freely or use different routes of varying length and difficulty. Currently, there are three prototypes in the Athene Exergaming product family: Athene asic, Athene Advanced, and Athene Premium. These prototypes differ in their setups, with Athene Premium being the most developed, featuring better lighting and three video projectors.

Figure 3 - The Athene Advanced [12]

Several studies were conducted with three objectives: to evaluate the suitability of the constructed exergaming simulator for different target groups, to identify the strengths and weaknesses of the prototypes, and to analyze the business prospects of various target markets. The Athene Software was used for Orienteering, where participants walked around the virtual city of Kajaani on a treadmill. It was also used for gym sessions, where participants could choose from four different routes at the Hukka Health Club in Finland. The software was tested on 56 participants, and they reported that it was a more enjoyable way to exercise.

Figure 4 - Athene exergaming for gym [12]

The software was also utilized for exercise testing, skiing with roller skis on a treadmill up to the top of Vuokatti hill, exploring an activity park, and rehabilitation exercises using a restorative bike for leg rehabilitation, with 12 elderly Japanese subjects participating. Through these different activities, it was concluded that exergames have the potential to motivate people of all ages to become more physically active. Exergaming, particularly with features that allow users to travel virtually, can provide fascinating experiences for many people, especially in areas that are too dangerous or uncomfortable for outdoor exercise.

The CAVE system was used to treat other diseases such as Rheumatoid Arthritis and Vestibular Disease. Rheumatoid Arthritis is a chronic, inflammatory autoimmune disorder that causes pain and swelling in the joints. The authors introduced the first steps in creating a virtual environment using a

CAVE System for physical therapy sessions, where the user will be engaged and motivated to complete the exercises prescribed by their doctor. Student volunteers simulated a patient by moving their arms in big circles while hitting specific target points along the circle's path. For Vestibular Disorder, the authors presented a theoretical justification for using a wide field of view (FOV) virtual reality display system for use in vestibular rehabilitation. A wide FOV environment offers some unique features that may benefit vestibular rehabilitation, just like in the CAVE. The participants tried to maintain balance while various patterns with some movements appeared on the walls around them. It was concluded that the tests were very successful, and the participants reported that it was a pleasant way to improve their movements. The data collected can be used by medical staff to track progress and make adjustments to the exercises. However, incorporating game-like activities in the future would help engage patients better. Adding locomotion within the virtual environment and incorporating a treadmill could enhance the functionality of rehabilitation. The fact that participants have to use sensors with cables may be a hindrance and should be addressed in future work.

3. Methodology

The project initial idea was to create a supermarket environment using the CAVE system. The supermarket will have empty shelves, with each shelf corresponding to a different category (drinks, bread, cereals, fruits, biscuits, etc.). Products will appear one by one on the floor of the wall, and the activity aims to pick up the products and drag them to the correct shelves to complete them. When the user places a product on the wrong shelf, it will return to the initial spot, allowing the user to try again. There will be shelves on the left and right walls with products, while the front wall will not have any interaction with the user but will inform the user about their progression in the activity, displaying a dynamic interface showing the number of completed shelves and the number of products required to complete a shelf. Figure 5 represents the interaction flow.

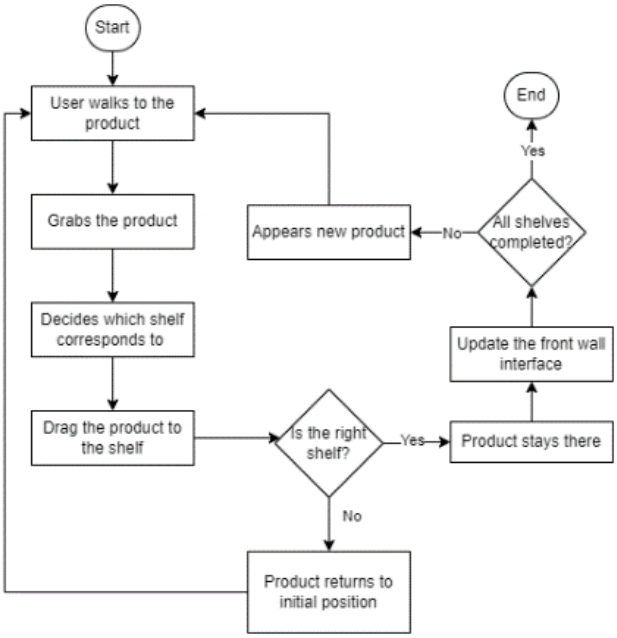


Figure 53 - Project Flow Diagram

After completing the development of the RehabMarket game, the next step is to conduct user testing sessions to gather feedback from participants. The goal of these sessions is to answer three main questions:

1. Is RehabMarket fully functional without any user errors?
2. How is the game compare to the actual experience of shopping at a supermarket?
3. How do participants evaluate their experience playing the game?

RehabMarket has the potential to offer several benefits to patients undergoing rehabilitation. Firstly, the game provides a more dynamic alternative to traditional rehabilitation methods. As a game, it can make the task more enjoyable and motivating for the patient. Secondly, the simulated supermarket environment is familiar to patients, facilitating their integration into the activity. The game also stimulates both the upper and lower limbs as patients must physically move to pick up and place products on the corresponding shelves. Additionally, the cognitive component is stimulated as patients must associate products with their correct category. Overall, RehabMarket has the potential to improve patients' rehabilitation outcomes by offering an engaging and familiar environment that stimulates multiple aspects of their physical and cognitive abilities.

3.1 Additions/Innovation/Contribution

Participation in this activity encourages individuals to stimulate both their motor and cognitive capacity to achieve the desired results. The simplicity of the task - filling the shelves - makes data collection during user testing intuitive, and all interactions will be recorded.

Furthermore, the inclusion of locomotion in the activity is important to stimulate the lower limbs, as the participant will need to walk to obtain and place the product on the correct shelf.

With these enhancements, RehabMarket is expected to meet all requirements and be accessible to users with varying preferences, providing a unique and immersive experience through the selection of products and environment.

3.2 Requirements collection

One of the most critical steps in project development is the collection of requirements. The process aims to understand the customer's needs before implementing the project [22]. This approach ensures that the system solves real problems by aligning with the customer's needs. Various techniques such as interviews, focus groups, questionnaires, and brainstorming sessions can be used to collect requirements [22].

To understand what should be implemented for the project, a focus group was organized, which is a widely used technique for gathering requirements in science and marketing [21]. This technique involves collecting data resulting from group-oriented conversations to analyze and receive feedback from participants about the project's ideas. Participants can include end-users of the system or people related to the project's theme. The goal of holding such meetings is to understand the customer experience and analyze various aspects of the project to improve it [21].

The focus group session was previously planned and structured, and underwent several reformulations [Appendix C]. The session was divided into two parts. The first hour of the focus group focused on presenting the context, goals, and possible system concept. The presentation included an introduction to stroke, followed by a question and answer session and a demonstration of the work completed up to that time. The second part of the session was reserved for a group activity named the Patient Journey Map, which will be detailed later in this document.

Since the project aims to rehabilitate people who have suffered a stroke using technology, people from different areas were present in the focus group session. Two psychologists, one occupational therapist, and two technology professionals participated in the meeting. The session also had a person responsible for taking notes of everything that would be mentioned by the participants for later summarization. The session was held on May 6, 2022, from 9:00 am to 11:00 am, and took place at Arditi.

Before commencing the initial presentation, participants were asked to sign an informed consent form that contained essential information about the focus group session. Additionally, they provided permission for the meeting to be audio recorded and photographed [Appendix A].

Following this, a brief introduction was given on stroke and its primary characteristics, focusing on the causes for its occurrence and the main sequelae associated with stroke, with a particular emphasis on hemiplegia and hemiparesis as they are the most relevant to the development of this project. Further, the presentation highlighted the limitations of existing technologies that were identified during the research and their potential to improve the final product.

The presentation also explained the main reasoning behind developing a game that would assist in post-stroke rehabilitation, specifically for individuals with hemiparesis and hemiplegia. The game was to be held in the CAVE system and would take place in a supermarket environment.

The subsequent section of the focus group presentation was designed to be more interactive, encouraging interaction between the moderator and participants. Participants were asked, "What kind of activities can we have in a supermarket that are associated with rehabilitation exercises?" The answers included picking up objects at different heights, selecting products from a shopping list, picking up and dropping objects, organizing purchases in a bag, budgeting for purchases, and having obstacles on the ground to make the activity challenging.

After receiving some ideas for the project's development, the initial concept of RehabMarket was presented, along with its advantages over existing systems. This was followed by a demo of the tasks that were created to simulate the activities of grabbing, dragging, and dropping objects in specific locations.

The first part of the focus group concluded with a question-and-answer session, where participants were asked to provide their opinions on three questions. The first question aimed to understand the advantages and disadvantages of playing these types of games in a virtual supermarket environment. The answers were relatively unanimous, with participants stating that the advantages included a familiar environment for most people and the ability to engage in various cognitive activities, such as categorization. The most commonly mentioned disadvantages were technical issues that can arise when using the CAVE system and difficulty in detecting fine motricity.

The second question aimed to understand how the game could provide feedback to the participant since it will have to be as autonomous as possible. The presenter highlighted the importance of audio feedback to make the game more dynamic, and participants suggested the use of a big virtual screen on the front wall of the CAVE to show instructions or a shopping list.

The third and final question aimed to understand which supermarket products should be used in the game. Participants agreed that basic everyday products, preferably healthy, should be used.

To conclude the focus group session, a group activity was conducted, with the aim of identifying how each difficulty level would be represented based on the suggested game variables. Participants were provided with a table that had three columns, with three different levels. In each column, the group needed to fill it with the variables they thought should be present [Appendix B]. They then had to explain how that variable would be parametrized on the different levels, giving examples.

Figure 6 - Group Activity

Figure 7 - Group Activity

By having these 3 columns filled with different variables, it was easier to think about the 10 levels that were supposed to exist, as it just needs an adaptation according to the participants' opinions.

Figure 8 - Group Activity

Figure 9 - Group Activity

The following tables represent how the 2 groups generated the associated variables in the different levels.

Table 5 - Group 1

Variable name	Level 1	Level 2	Level 3
Time	10 min	20 min	30 min
List size	10-15 objects	20-25 objects	30-35 objects
Number of a shelf set	1 left and 1 right	2 left and 2 right	3 left and 3 right
Number of shelves	3 to 4 shelves		
Position of products	Easiest as possible at the level of the upper limbs	Vary between top shelves	Freedom of movement between all existing shelves

Table 6 - Group 2

Variable name	Level 1	Level 2	Level 3
Time	Fixed time (10 min)		
List size	5 objects	10 objects	20 objects
Distance between product and destination	Always moves from the same shelf	Move products between shelves	Move products from shelf to basket
Budget	No	Budget set	
Junk Product	2 times	4 times	6 times
Promotions (audio)	Background music	Related audio promotions	Audio promotions related or not to the goal
Number of shelves	2	3	6

Products per shelf	10	20	30

Several concepts have been proposed to introduce different levels of complexity and dynamics to the game. In this activity, we have decided to define variables for three levels: the simplest, the average difficulty, and the most challenging. The decision to focus on three levels was made to simplify the task for the participants, but we plan to create ten additional levels with appropriate variable values.

The table presented above shows the initial distribution of variables across the levels. After conducting a focus group, we reviewed and summarized the information to simplify and clarify the task for the participants. The following key points emerged:

- The main task is to pick up an object and move it to another location;
- The game includes a shopping list of items to be collected and placed in a shopping cart;
- Players must select products based on a budget;
- The game includes interactions with the shelving units;
- All feedback, instructions, and shopping lists are displayed on a virtual screen;
- Audio/visual feedback will be provided through beeps and colors;
- Images will be used instead of text whenever possible.

With this information, we have defined ten levels of play.

Table 7 - Final Result

Variable name	Level 1	Level 2	Level 3
Time	Fixed time (10 min)		
List size	5 objects	10 objects	20 objects
Distance between product and destination	Always moves from the same shelf	Move products between shelves	Move products from shelf to basket
Budget	-	-	Budget set
Distractors	2 products	4 times	6 times
Number of a shelf set	1 left and 1 right	2 left and 2 right	3 left and 3 right
Number of shelves	3 to 4 shelves		
Products per shelf	10	20	30
Position of products	Easiest possible at the level of the upper limbs	Vary between top shelves	Freedom of movement between all existing shelves

The interaction flow will be as follows:

1. Drag products on the same shelf;

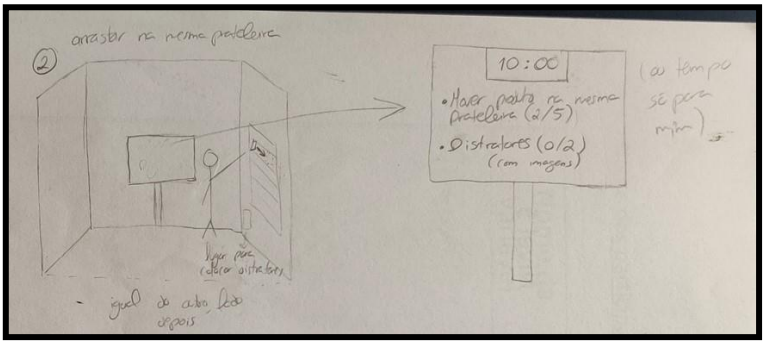


Figure 10 - Level 1

2. Drag products between shelves;

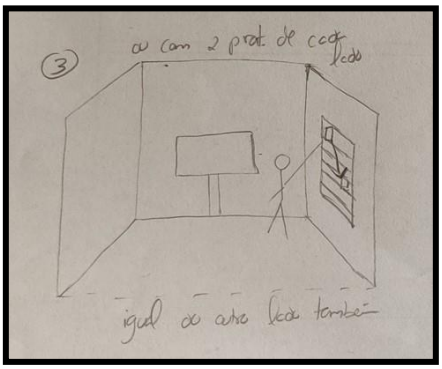


Figure 11 - Level 2

3. Drag product from shelf to shopping cart;

4. Drag distractors to the right place;

5. Reconcile activity with products and distractors;

6. With the shopping list, identify products on the shelves by touching them;

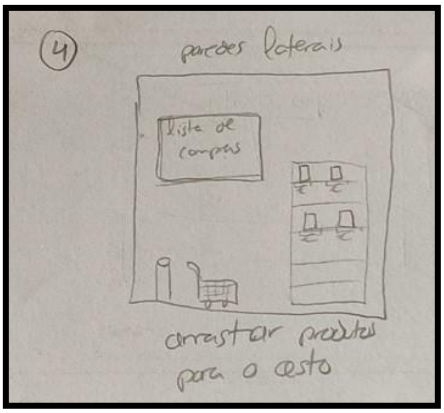


Figure 12 - Level 6

7. With the shopping list, drag products to the shopping cart;

8. The shopping list appears for a while and then disappears;

9. A budget appears and taking into account the prices of the products, the person drags them into the cart, with the budget noticeably decreasing;

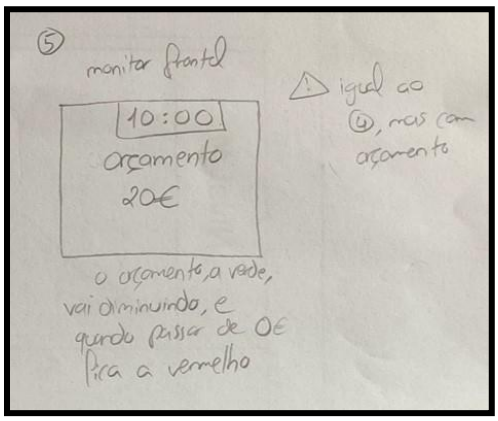


Figure 13 - Level 9

10. Similar to 9 but does not appear the budget to decrease.

After thinking about the structuring of the game and its levels, it was necessary to map more clearly the structure of the system to be developed by modelling its classes, attributes, and relationships between objects, so that, a Class Diagram was created.

Those Class Diagrams are very useful because allow for modelling, in a declarative way, the static structure of an application domain, such as concepts and the relations between them. [23].

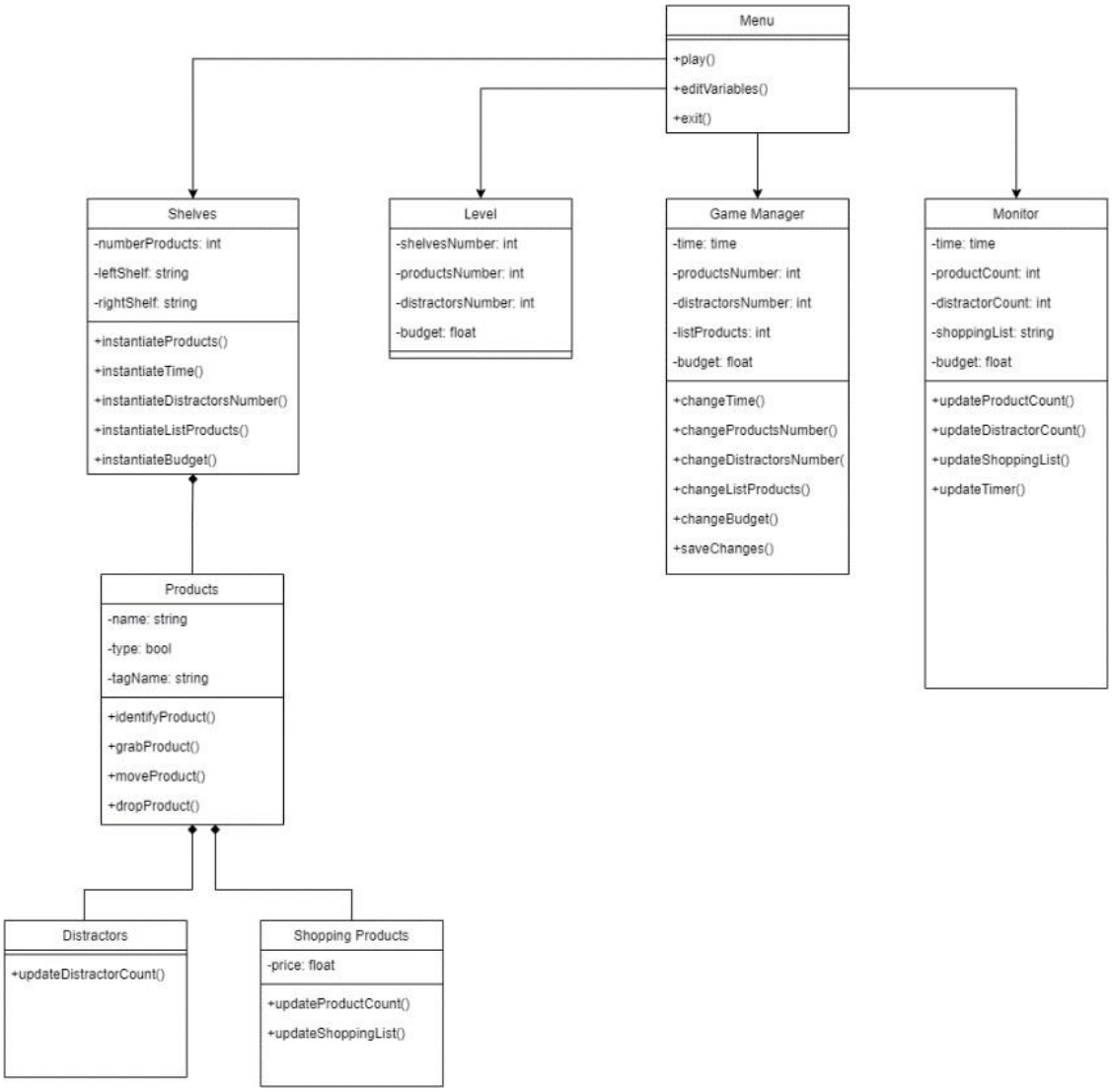


Figure 14 - Class Diagram

Figure 14 illustrates the distribution of classes and objects in the system. The main class, "Menu," will be the first scene presented to the participant upon launching the game. The user can interact with the system by selecting one of three options: starting the game using the play() method, modifying game variables with the editVariables() method, or exiting the game using the exit() method.

Within the game scene, there will be four main entities: Shelves, Level, Game Manager, and Monitor. Shelves represent all shelves in the game, with each shelf containing a specific number of products (numberProducts) and divided into left and right shelves (leftShelf and rightShelf). The shelves will have associated methods, such as instantiateProducts(), which will place the products on the shelves that the participant can interact with. Depending on the number of products chosen for each shelf, the corresponding number of products will appear.

Time-limited levels will require the instantiateTime() method to use the time defined by the participant. Additionally, three other essential methods will instantiate the number of distracting objects, the products that will appear in the shopping lists, and the budget that the person will have to spend on products. These methods will be named instantiateDistractorsNumber(), instantiateListProducts(), and instantiateBudget(), respectively.

To associate products with shelves, a relationship between Products and Shelves is established. Each product will have a name and belong to a specific type, which can be either Distractor or Shopping Product. Additionally, each product will have a tagName, which will identify it within the Unity software. There will also be several methods associated with each product, such as identifyProducts(), which identifies whether the product is in the shopping list; grabProduct(), responsible for getting the participant to grab the product; moveProduct(), responsible for the product's movement from one side to the other; and dropProduct(), responsible for dropping the product when it reaches the correct final destination.

As previously mentioned, products in the system are categorized into two types: Distractors or Shopping Products. Distractors are products that do not match the items in the shopping list, while Shopping Products are those that are included in the list with an associated price. Each product type has its own set of methods. Distractors have only one method, updateDistractorCount(), which updates the distractor counter. On the other hand, Shopping Products have two methods: updateProductCount() and updateShoppingList(), which respectively update the product counter and the shopping list.

Another important entity is the Level, which has several attributes including the number of shelves (shelvesNumber), number of products (productsNumber), number of distractors (distractorsNumber), and budget. The Game Manager is responsible for defining all game variables and their representations. Its attributes include time, number of products, number of distractors, number of products in the shopping list, and budget. Its methods include functions to modify each attribute, as well as saveChanges(), which saves all changes made to the variables.

The Monitor entity is responsible for displaying various information on the virtual screen, such as game time, product and distractor counters, shopping list, budget, and menu. Its methods include updateProductCount(), updateDistractorCount(), updateShoppingList(), and updateTimer(). These methods allow the virtual screen to be updated according to the actions performed by the participant.

Flow diagrams are another tool that can be used to optimize the system's processes. They provide a detailed explanation of each step in the process and can help optimize the paths of people, objects, or information through the system.

The image in Appendix P shows all the interactions that the participant will have during the game. The game starts with a menu that allows the player to start the game, change the game variables, or exit. If the player wishes to modify the variables, a submenu is displayed where changes can be made and saved. Once the game is started, the player begins at level 1, where the variables are instantiated and the activity is performed. Successful completion of the tasks results in a sound effect and the score is updated on the virtual screen. The player then advances to the next level, where the variables are instantiated and the tasks are performed again. This process continues until all levels are completed, at which point a successful final menu is displayed, and the game ends.

3.4 Setup

The game was created using the Unity platform, and participant movements were tracked through the use of a Kinect camera. The KAVE KinectForWindows_UnityPro plugin [25] was utilized to enable interaction within the CAVE environment, which features game projection on all four walls through the use of four projectors. This plugin permits the visualization of all CAVE walls within the Unity Editor, simplifying the process of creating the game environment and programming all necessary interactions.

3.5 Implementation

The development phase commenced subsequent to the thorough analysis of all information discussed in the focus group chapter. Following the successful installation of the plugin, the subsequent step entailed creating a game environment, namely a supermarket. The primary objective at this stage was to identify objects that, when organized, could simulate a realistic supermarket.

To this end, the lateral walls featured shelves with various products, while the front wall had a monitor displaying information about the level, accompanied by feedback to guide the participant in the game. A comprehensive asset, the Supermarket Interior [27], created by Daniil Demchenko and sourced from the Unity Asset Store [26], had all the necessary components to make the environment resemble a supermarket as closely as possible. The asset included wallpapers to simulate the walls and floors of a supermarket, shopping carts, television monitors, cash registers, refrigerators, shelves, and a wide range of supermarket products (see Figure 15). I extend my gratitude to Daniil Demchenko for generously donating the entire package to support this project.

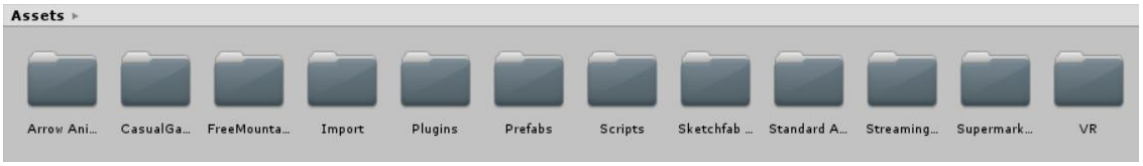


Figure 15 - Assets

After conducting multiple experiments on how the supermarket should be represented, several results were obtained. The iterative process involved adjusting the size of certain products that were found to be disproportionate to others until the final outcome was achieved. The front wall is uniformly represented in all levels, and the only change observed is the information displayed on the monitor, which varies from level to level, as elaborated later. In contrast, the elements on the side walls change frequently as the levels progress, as will be demonstrated subsequently.

Once all the necessary information was gathered, the next step was to learn Unity and gain practical experience since it was a relatively new platform for me. Concurrently, I generated some ideas for the final project, as previously discussed.

After completing the research phase, I was eager to test the feasibility of my idea for the final project. The initial step involved creating a basic Unity project, wherein the objective was to select an object and place it in a preferred location solely using the mouse. To grab the object, the individual needed to press and hold the left mouse button until they placed it in a different position.

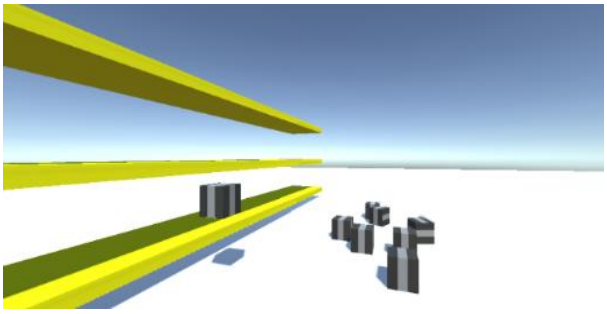


Figure 16 - Game Environment

As Figure 16 shows us, there are many black boxes, and the player just needs to drag them into any yellow shelf. After completing this grab-and-drop test, it would be necessary to perform the same experiment, but this time using the Kinect (Figure 17) and tracking the wrist joint was possible to grab and drop the object using the user's hand movement.

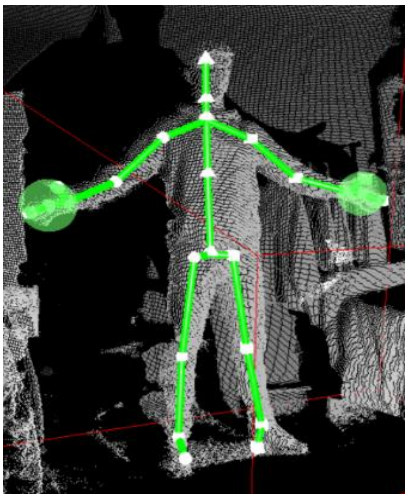


Figure 17 - Body tracking with Kinect

By successfully implementing this idea, the functionality of the final concept became more robust. Subsequently, the grab-and-drop feature was integrated with the KAVE Unity plugin, which allowed it to operate seamlessly within a CAVE system [17]. By using the Kinect and CAVE, the user could grab a red capsule with their hand, walk to the drop point, and place the capsule in that spot. The implementation was designed to detect and permit only the red capsule to remain in that location. This is the initial step towards filtering various objects and subsequently linking them with specific supermarket categories. The following sequence of images demonstrates how this feature works (see Figure 18 – Figure 20).

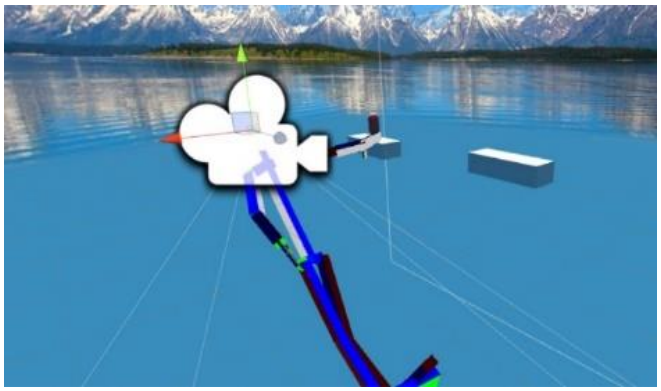


Figure 18 - Initial positions

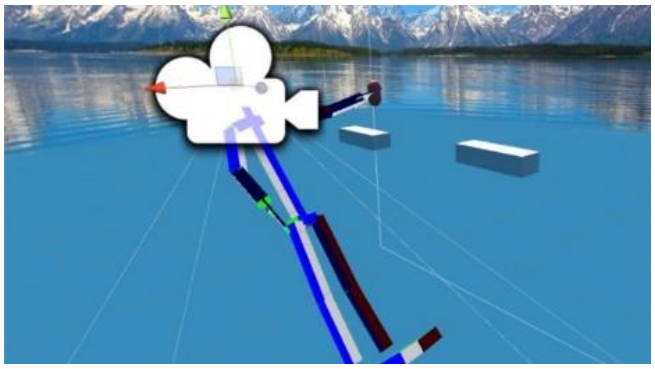


Figure 19 - Dragging the capsule

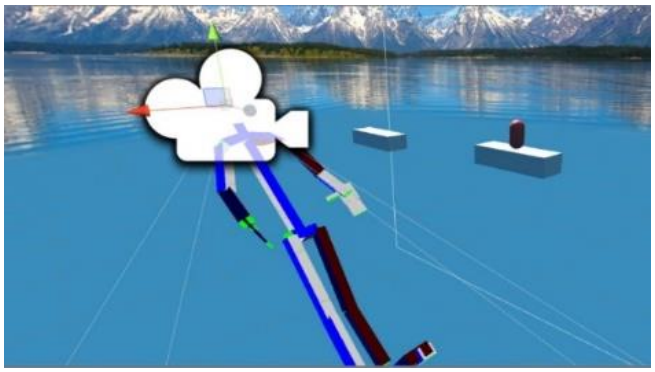


Figure 20 - Capsule stays in the correct spot

This experience was quite beneficial for future work, since the main idea of drag and drop was successfully made, however, the most important would be to test in the CAVE, since it would be where the final work would be carried out.

The first trial in the CAVE was not successful, because the capsule´s box collider was too big for what was needed. After correcting that box collider, it worked perfectly, but just needed a better calibration, as represented in the next sequence of figures (Figure 21 – 23).

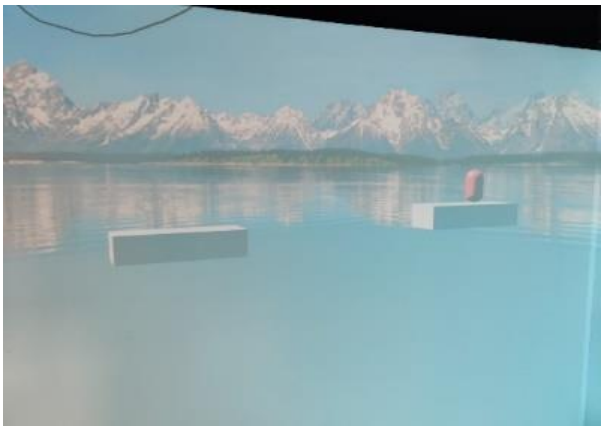


Figure 21 - Initial environment

Figure 22 - Dragging the capsule

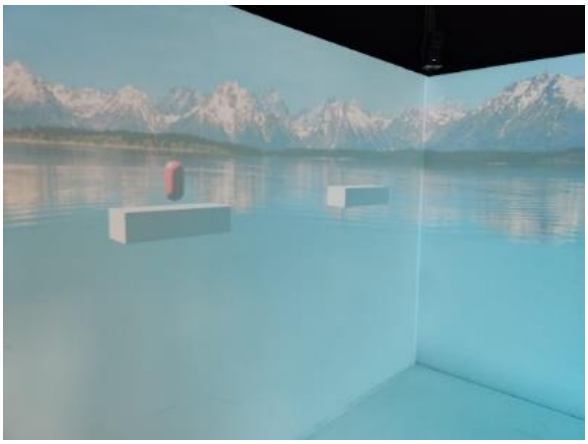


Figure 23 - Capsule stays in the correct spot

After conducting all of these prototype tests, it became evident that the final project could be achieved as originally envisioned. This was an excellent starting point for realizing the project's potential and limitations. For example, it was discovered that the front wall does not track the body and movements of a person when they are in close proximity, which is not an issue with any of the side walls.

To manage the Unity project's version control, BitBucket and SourceTree will be utilized.

3.6 Levels

In this chapter, every level present in the game will be displayed. For each level, an explanation will be provided along with images. To start the game, the user must initiate the first interaction, which is achieved by clicking on the "Jogar" button, as shown in Figure 24, which appears on the main menu page.

Figure 24 - MainMenu

The first level of the game requires the participant to drag the presented products to a specific target location on the same shelf. The target location is marked with a different texture and an arrow to indicate the appropriate place for the participant to drag the product. The side walls of the game feature shelves with products and the final location where the participant must interact, as shown in Figure 25.

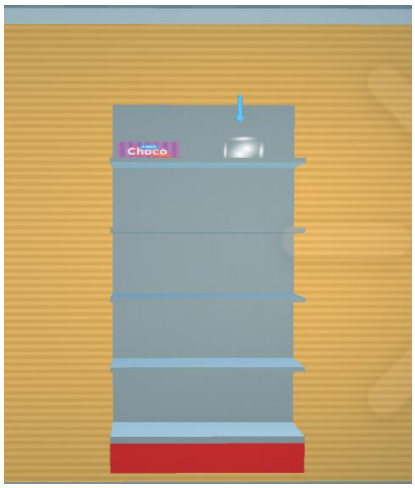


Figure 25 - Level 1

As can be observed, the designated area for product placement is indicated by a white texture and a dynamic arrow, providing guidance to the user. In order to successfully complete the level, both products, located on the right and left sides, must be placed in their respective target areas.

Moving on to Level 2, it closely resembles Level 1 in terms of interactions and objectives. The user must drag the product to the target location indicated by a white texture and arrow, as in Level 1. The only difference between the two levels is that in Level 2, the target area is located on a different shelf from the product's starting position, as opposed to Level 1 where everything remains the same (as depicted in Figure 26).

This change requires the user to execute movements distinct from those performed in the previous level, particularly involving the upper limbs in vertical movements.

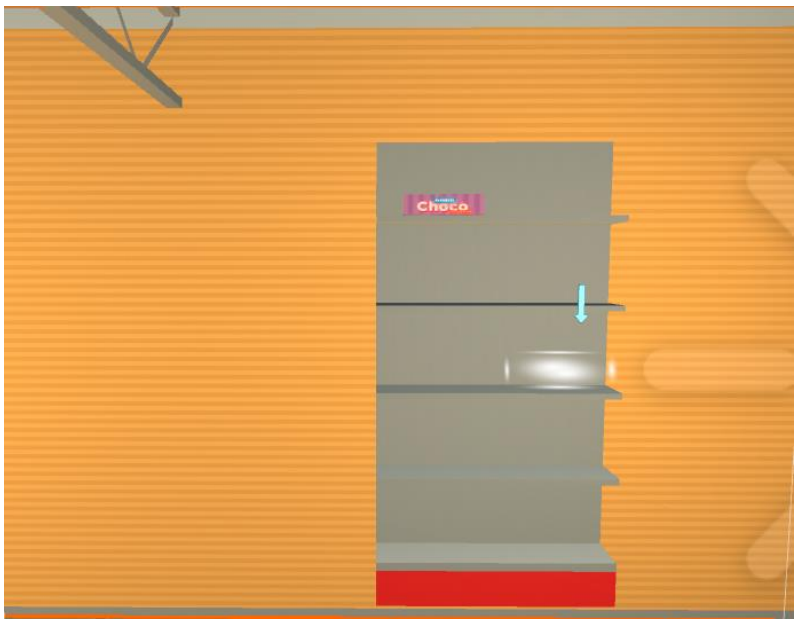


Figure 26 - Level 2

As presented thus far, Levels 1 and 2 primarily involve simple movements and interactions intended to help the participant grasp the dynamics of the RehabMarket environment. In these levels, the target positions were depicted as transparent boxes with a white texture, accompanied by a blue arrow that moved vertically to indicate where the product should be placed. However, starting from Level 3, targets will no longer be represented in this manner, but instead will be in the form of shopping baskets or large crates, and the objectives and interactions will vary depending on the level.

In Level 3, the objective of the game is simply to drag a product from each side of the shelves to a shopping basket located nearby, as illustrated in Figure 27.

Figure 27 - Level 3 Instructions

As already said, the *targets* will become shopping baskets and crates, and at this level will be shopping baskets.

Figure 28 - Level 3

The objective of this level is to drag the product into a shopping basket, with the understanding that all products must be placed in these baskets from this level and onward.

Up until this point, all levels have featured products that could be found in a supermarket setting. However, in Level 4, the concept is altered, as objects will no longer be represented as products, but rather as boxes.

Figure 29 - Level 4

The use of boxes as objects necessitates a different type of interaction compared to products. While products could be dragged into shopping baskets, boxes must be placed in crates located near the shelves, as shown in Figure 30.

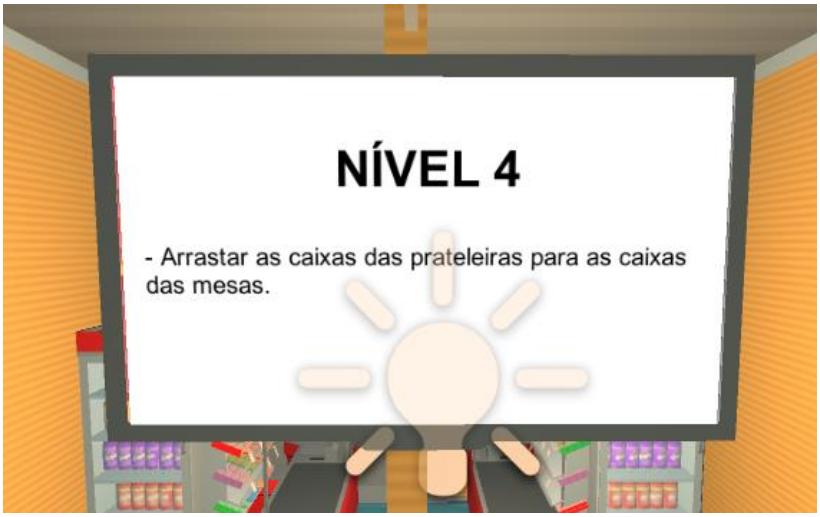


Figure 30 - Level 4 Instructions

After the user has completed the activities in Levels 3 and 4, which involved placing products in baskets and boxes in crates respectively, the game progresses automatically to Level 5. At this stage, users are required to perform a task that integrates everything presented in the preceding two levels. Specifically, Level 5 entails the random distribution of products and boxes on the shelves, with shopping baskets and crates also present.

Figure 31 - Level 5

Thus, the user must be able to distinguish between the products and the boxes since the products have to be placed in the shopping baskets and the boxes in the crates.

Each time the user places a product in a crate, or a box in the shopping basket, an error sound is heard, and the object is placed back in its original position.

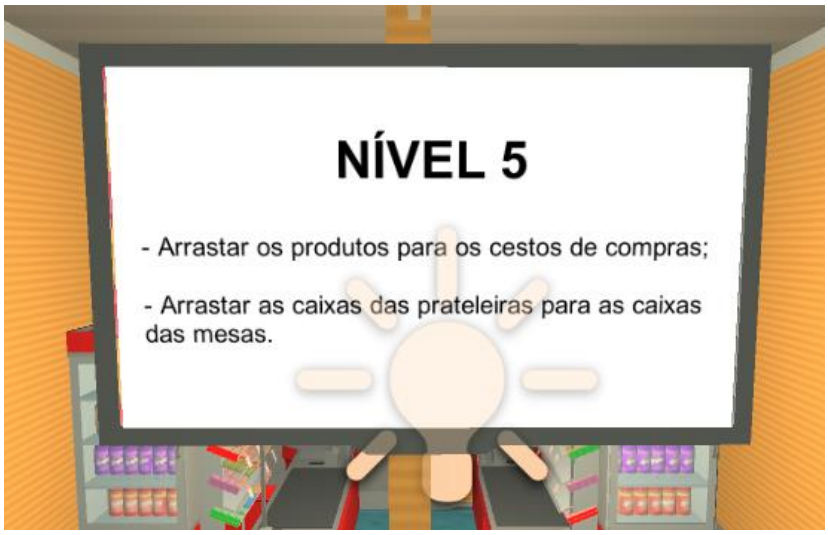


Figure 32 - Level 5 Instructions

As mentioned above, there is one level of the game in which the user does not need to drag products from one location to another but keeps the hand near the product for 1 second, and this level is the 6th.

Figure 33 - Level 6 Instructions

In Level 5, shopping lists are present on both the left and right sides, in addition to products that need to be located by the user. These lists display the products that must be found. If a product is not selected within 1 second, the corresponding square on the shopping list turns red (Figure 34). Once the product is accurately detected, the square turns green, indicating that the task has been successfully completed (Figure 35).

Figure 35 - Level 6 Product I

The level is completed once the user successfully finds all the products on both the left and right sides. Starting from Level 7, the game becomes increasingly more challenging and demands higher levels of concentration and cognitive effort from the user.

At Level 7, there are two shopping lists with various products listed and two shelves with products displayed. The objective of the level is for the participant to identify the products listed on the shopping lists and drag them to the corresponding shopping baskets.

Figure 36 - Level 7 Instructions

On each side of the wall, there are shelves containing products arranged in random positions. Some of these products appear on the shopping lists visible to the user, while two of them do not. It is the responsibility of the participant to identify which products appear on the lists and place them in their shopping basket, thereby simulating a purchase.

Figure 37 - Level 7

While a product is on the shelf, the square allocated to it, present in the shopping list, presents the color red. This changes to green as soon as the product is placed in the shopping basket.

Figure 38 - Level 7 Square

During gameplay, there may be opportunities for the user to attempt to select a product that is not on the designated shopping list. In the event of such an occurrence, the product will remain stationary and cannot be added to the shopping basket. The level concludes once all products listed on the designated shopping lists are successfully placed in the corresponding baskets.

Level 8 follows a similar design to that of Level 7, with products placed on shelves on both sides and two designated shopping lists. However, this level features a reduced number of items on each list and the lists themselves disappear after a period of 10 seconds.

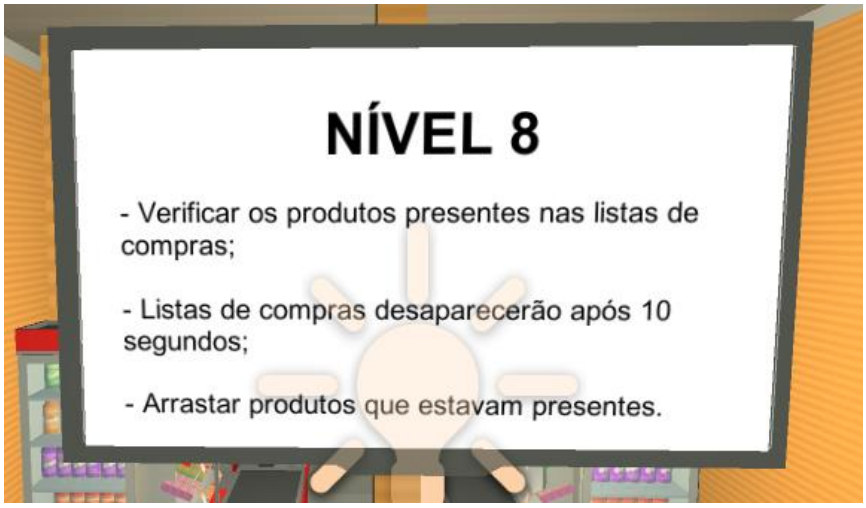


Figure 39 - Level 8 Instructions

The shopping lists are defined as having only 2 products each for the user to memorize. Thus, when the lists disappear, the user must remember which products were on the lists.

Figure 40 - Level 8 Visible list

Figure 41 - Level 8 list not visible

As in previous levels, the level ends when all products on the lists are placed in the displayed shopping baskets.

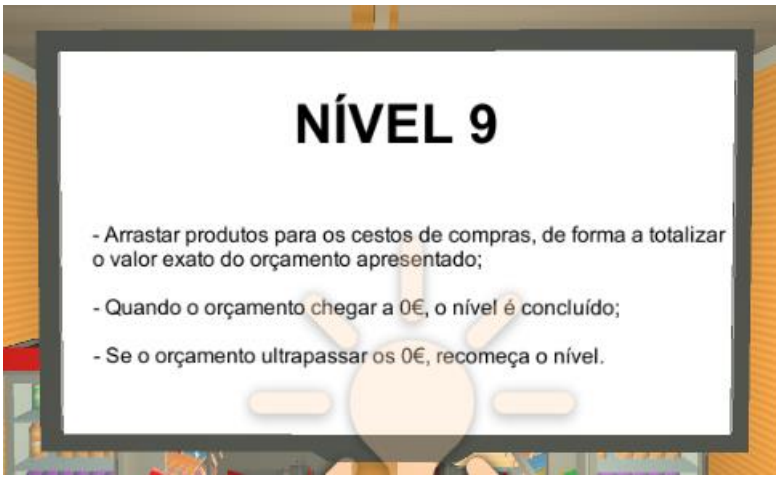


Figure 43 - Level 9 Instructions

Starting from Level 9, a budgetary concept is introduced in the game. A specific value is randomly assigned as the level budget, and each product on the shelves has an associated price. The objective is to place the products in the shopping baskets in a manner that reduces the budget to its total value.

For instance, in Figure 45 shown below, the budget was set at 25€, with each product having a corresponding price. It is important to note that for this level and the next one, interactions with products on both sides of the wall contribute towards the same budgetary goal. Therefore, the budget remains the same for both sides, and the user can add products to the baskets from either side.

Figure 43 - Level 9 Shelves

If the participant happens to add the red product, which is priced at 3€, to their shopping basket, the budget will be adjusted accordingly, subtracting the price of the product and updating the total to 22€. This process continues until the budget reaches 0€.

Figure 44 - Level 9 Product price

The level concludes once the budget reaches 0€. If the budget exceeds this value, the level will resume, providing the participant with another opportunity to complete it.

Level 10 is considered the most challenging level for the participant, as it requires more effort, involving calculations and memory. The objective is similar to that of Level 9, with the goal being to reduce the budget to its total value. However, the budget monitor disappears after 10 seconds, adding an additional level of difficulty to the game.

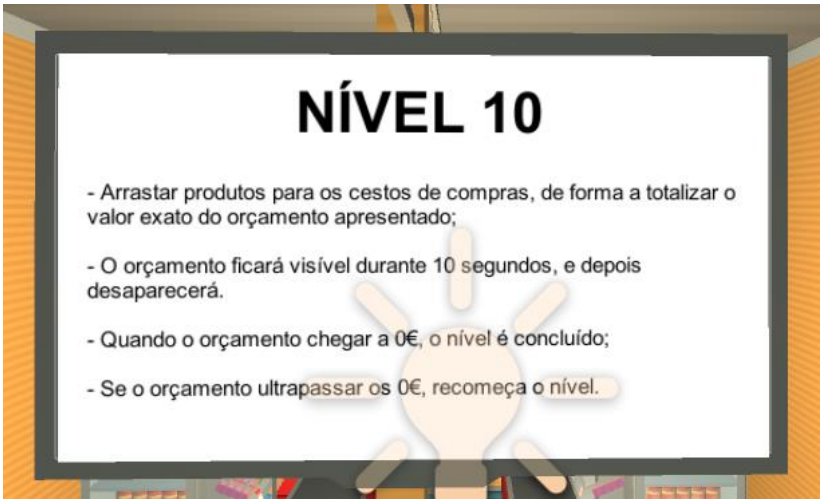


Figure 45 - Level 10 Instructions

To conclude the game, a level was designed to inform the player that they have reached the end. This level does not contain any shelves or products, only a monitor displaying information.

Figure 46 displays what appears on the monitor, with the game congratulating the participant by saying "Parabéns" and informing them that they have successfully completed the game. Additionally, there is a "Jogar de novo" button, allowing the player to replay the game by simply clicking on it.

Figure 46 - Last

Once the whole environment was created, it then went to the programming of the 10 different game levels. To this end, a set of *scripts* were used to guide the interactions and dynamics of the game.

3.7 Scripts

As it is typical for this project, an iterative process is required, meaning that the ideas and structure are in constant flux to achieve the best possible outcome. Initially, a Class Diagram was created to represent the project. However, after some modifications, the diagram underwent changes, and all the classes are presented below. The updated Class Diagram can be found in Appendix O.

The first script to be presented is NextScene.cs. It is responsible for loading scene number 1 when the user clicks the "Play" button in the "Main Menu," which will be shown later. It utilizes the Input.GetMouseButtonDown(0) function on the aforementioned button and loads the scene using the SceneManager.LoadScene(1) function from the UnityEngine.SceneManagement library.

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

0 references
public class NextScene : MonoBehaviour {


    0 references
    private void Update()
    {
        if (Input.GetMouseButtonDown(0))
        {

            SceneManager.LoadScene(1);
        }
    }
}
```

Figure 47 - NextScene Code

The "Item.cs" script represents the product class, which is characterized by several attributes, including "name," which represents the product's name; "targetName," which specifies the name of the location where the product should be placed; "price," which indicates the cost of the product for use in later levels; "dragging," a boolean that indicates whether the product has been placed correctly; "rightProduct" and "leftProduct," which determine whether the product is on the right or left shelf, respectively; "rightTarget" and "leftTarget," which are activated depending on the wall's side when a product is grabbed; "position," which stores the product's location at the start of the level, and "square," which denotes a specific square for each product. The square is initially red (Figure 48), and when the product is placed correctly, it turns green, indicating that it has been dragged to the appropriate location successfully.

Figure 48 - Product square

```
public class Item : MonoBehaviour
{
    public string name;
    public string targetName;
    public float price;
    public bool dragging = false;
    public bool rightProduct;
    public bool leftProduct;
    public bool rigthTarget = false;
    public bool leftTarget = false;
    public Vector3 position;
    public SpriteRenderer square;
```

Figure 49 - Item.cs, Product characteristics

To accurately detect and store a product's initial position in the "position" variable, a "setPosition" function was created. This function accepts the "pos" variable, represented as a Vector3, as an argument. The "position" variable, which was defined in Figure 49, is then assigned to "pos", as illustrated in Figure 50.

```
public void setPosition( Vector3 pos)
{
    position = pos;
}
```

Figure 50 - SetPosition function

All of the variables created can be viewed in the Unity editor. This is where the values of some of these variables are set, which allows for checking their changes as the activities progress, as demonstrated in Figure 51.

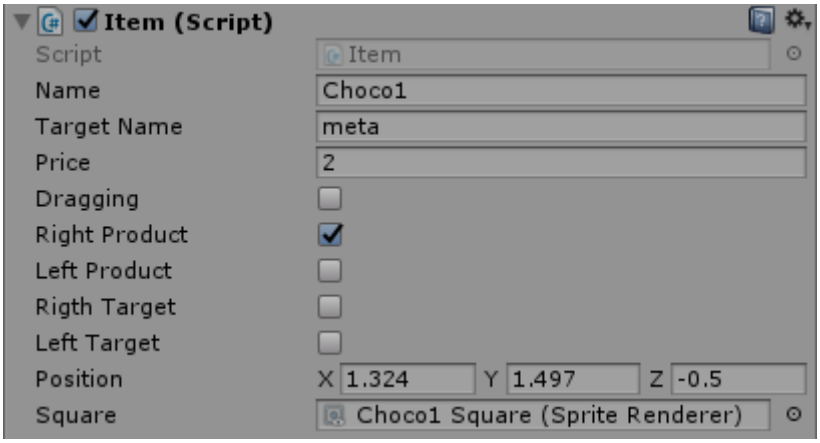


Figure 51 - Item.cs in unity editor

Regarding the "Targets.cs" scripts, each target, or location where products are placed, must be identified. This script sets the side to which each target belongs by creating two boolean variables: "rightTargetOn" and "leftTargetOn". Both variables are initially set to false, and as soon as the level begins, the targets on the right wall activate "rightTargetOn," and those on the left side activate "leftTargetOn". This distinction between the sides of the targets simplifies the process of checking and placing products in the appropriate targets, as explained later.

```
public class targets : MonoBehaviour {
    public bool rightTargetOn = false;
    public bool leftTargetOn = false;
```

Figure 52 - Targets.cs

As previously mentioned, the difficulty of the game increases as levels progress. One of the contributing factors to this increase in difficulty is the positioning of products and targets. The required interaction forces participants to move their arms in multiple directions, which is more challenging than simple linear movements. The same applies to lower limb movements, with some levels requiring individuals to move from side to side more frequently, which can be more complex than levels where the participant remains static.

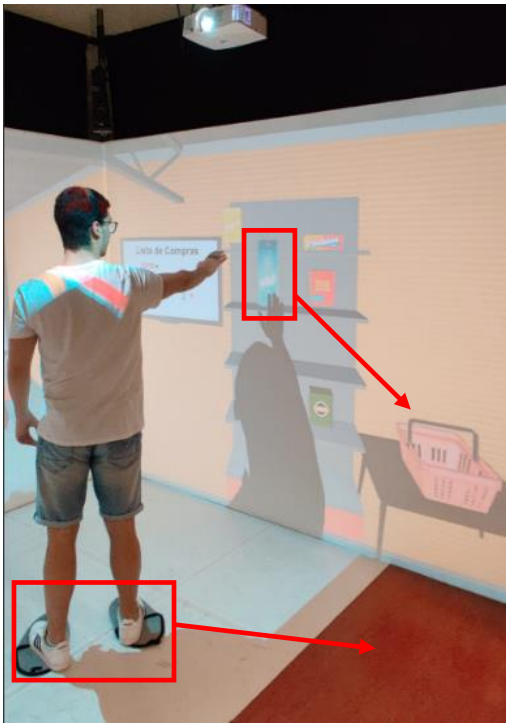


Figure 53 - Body movement

One important aspect that needed to be taken into account is the randomness of the positions of each object in the game. This allows each level to be different every time it is played, making the game more dynamic and unpredictable.

To achieve such randomness of positions, scripts named "Shelf.cs", "Shelf2.cs", and "Shelf5.cs" were created. These scripts create two Vector3 lists, called "rightPoslist" and "leftPoslist", which represent all the positions that the level products can occupy. The size of the lists is defined, which represents the number of positions that can be occupied by the products and targets. In this case, only two positions were defined for each list.

```
public class Shelf : MonoBehaviour
{
    public List<Vector3> rightPoslist = new List<Vector3>();
    public List<Vector3> leftPoslist = new List<Vector3>();
```

Figure 54 - Shelf.cs Lists

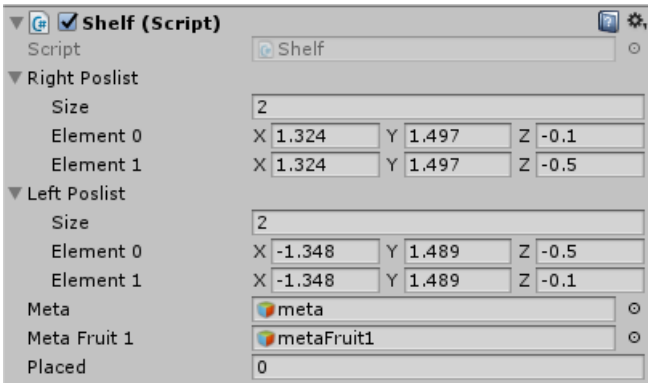


Figure 55 - List size

To randomly define the position of the *targets*, it was necessary to invoke the "GameObjects" that represent them, in this case, the "meta", which is the *target* on the right side, and the "metaFruit1", which is the *target* on the left side.

```
public GameObject meta;
public GameObject metaFruit1;
```

Figure 56 - Shelf.cs targets

A variable was also created that would allow counting of how many products were placed in the correct place. This variable is called "placed". This becomes quite useful because it allows us to know how many products are already placed in the *target*, and how many are left before the level reaches the end.

```
public int placed;
```

Figure 57 - Shelf.cs placed

```
20      void Start()
21      {
22          placed = 0;
23          GameObject[] prod = GameObject.FindGameObjectsWithTag("Product");
24          foreach (GameObject g in prod)
25          {
26              if (g.GetComponent<Item>().rightProduct == true)
27              {
28                  g.transform.position = rightPoslist[Random.Range(0, rightPoslist.Count)];
29                  g.GetComponent<Item>().setPosition(g.transform.position);
30                  rightPoslist.Remove(g.transform.position);
31                  meta.transform.position = rightPoslist[Random.Range(0, rightPoslist.Count)];
32                  rightPoslist.Remove(meta.transform.position);
33              }
34
35              else if (g.GetComponent<Item>().leftProduct == true )
36              {
37
38                  g.transform.position = leftPoslist[Random.Range(0, leftPoslist.Count)];
39                  g.GetComponent<Item>().setPosition(g.transform.position);
40                  leftPoslist.Remove(g.transform.position);
41                  metaFruit1.transform.position = leftPoslist[Random.Range(0, leftPoslist.Count)];
42                  leftPoslist.Remove(metaFruit1.transform.position);
43              }
44          }
45      }
```

Figure 58 - Shelf.cs

In this script, the level is initiated with the variable "placed" set to 0, indicating that no product has yet been correctly placed (line 22). All products that need to be placed have the same tag, "Product", which identifies them as objects of the same type (line 23). When the level is loaded, products are randomly positioned on the shelves by finding all objects with the "Product" tag and adding them to a list called "prod" (line 23).

Once all products have been added to the "prod" list, it is necessary to determine which products are on the left and right sides of the shelves (lines 26 and 35). For each element in the "prod" list, if the "rightProduct" variable is true (line 26), it means that the product is on the right side and its position corresponds to one of the positions defined in the "rightPoslist" (lines 28 and 29). Once a product is in a defined position, that position is removed from the list (line 30).

Similarly to the products, the target position on the shelf is also randomly assigned by selecting a position from the "rightPoslist" and removing it from the list (lines 31 and 32). The target object, "meta", represents the target on the right side of the shelf.

For products on the left side of the shelves (line 35), the same random positioning is applied, except that the list "leftPoslist" and the target name "metaFruit1" are used instead. Initially, only fruits were meant to be placed in this target, so the name remained the same.

At the beginning of each level, the front monitor displays the level number to the user (Figure 59). Using the script "WaitForTime.cs", the user is allowed 3 seconds to view this information before it disappears and is replaced with instructions on how to successfully complete the level.

Figure 59 - WaitForTime.cs, Level information

```
5   public class WaitForTime : MonoBehaviour {
6
7       // Use this for initialization
        0 references
8       void Start () {
9           StartCoroutine(Test());
10      }
11
12      // Update is called once per frame
        0 references
13      void Update () {
14
15      }
        1 reference
16      IEnumerator Test()
17      {
18          yield return new WaitForSeconds(3);
19
20          Destroy(gameObject);
21      }
22  }
```

Figure 60 - WaitForTime.cs

To create the timer, a Coroutine was utilized, requiring an IEnumerator function, named "Test". This function specifies the duration of an action, which in this case is to display the level being played for 3 seconds (line 18) followed by its disappearance (line 20).

To enable the function, it was invoked in Start(), which begins the 3-second count as soon as the level starts, initiating the Coroutine (line 9).

The Move.cs script was the first to be developed and is essential for game interaction. It allows the user to move objects in the game.

To enable physical collisions between objects, the size of the colliders of the products present in the level had to be defined. The collider can be customized by the developer to facilitate or hinder the ability to collide with the object.

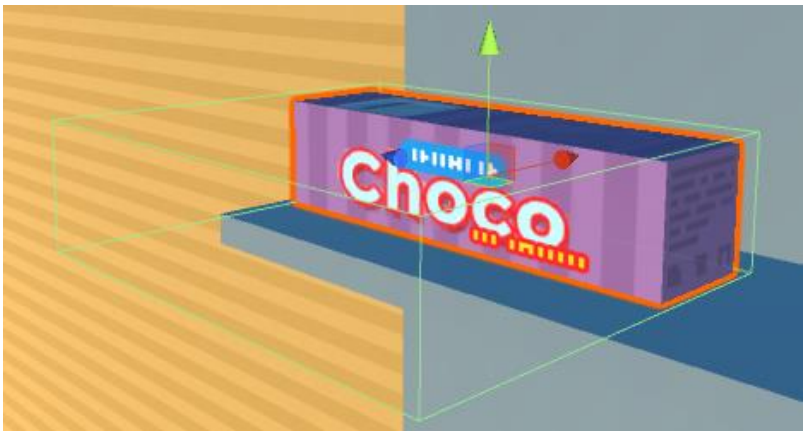


Figure 61 - Product Collider

As depicted in Figure 61 above, the product collider is represented by a box with green borders that encloses the object, which enables user interaction upon contact with said collider. All products, shopping baskets, and boxes have colliders, as previously mentioned.

The "Move.cs" script is primarily responsible for allowing participants to drag objects from one location to another. This is achieved through the "OnTriggerEnter()" function, which receives a Collider argument named "other" (line 104). In this context, "other" represents the upper limbs of the participant, which, upon contact with an object, trigger an action.

Subsequently, a check is implemented to determine which objects can be moved. Specifically, all non-target objects are deemed movable, which is achieved by restricting objects to those without the "target" tag (line 106). A check is also performed to determine if the participant has already positioned the object correctly by setting the "dragging" variable to true (line 108).

When the participant approaches an object to grasp it, intercepting the object's collider causes the object's position to align with the participant's hand (the "other" argument), thereby enabling the object to be carried throughout the created environment (line 114).

Once the participant has successfully grasped the object, the system distinguishes whether the object is on the right or left wall to determine the appropriate target location. If the object is on the right side (line 115), the product variables "rightTarget" and "rightTargetOn" (in this case, "meta") are set to true, indicating that the system should transport the product to the target on the right side (lines 117 and 118). Conversely, if the object is on the left side of the environment, the variables reference the left side, such as "leftProduct," "leftTarget," and "leftTargetOn" (lines 121 to 124), and the target is named "metaFruit1.

```
104     public void OnTriggerEnter(Collider other)
105     {
106         if (other.tag != "target")
107         {
108             if (gameObject.GetComponent<Item>().dragging == true)
109             {
110                 gameObject.transform.position = new Vector3(13.643f, 2.831644f, -1.637f);
111             }
112             else
113             {
114                 gameObject.transform.position = new Vector3(other.gameObject.transform.position.x, other.gameObject.transform.position.y, other.gameObject.transform.position.z);
115                 if (gameObject.GetComponent<Item>().rightProduct == true)
116                 {
117                     gameObject.GetComponent<Item>().rigthTarget = true;
118                     meta.GetComponent<targets>().rightTargetOn = true;
119                 }
120
121                 else if (gameObject.GetComponent<Item>().leftProduct == true)
122                 {
123                     gameObject.GetComponent<Item>().leftTarget = true;
124                     metaFruit1.GetComponent<targets>().leftTargetOn = true;
125                 }
126             }
127         }
128     }
```

Figure 62 - Move.cs, OnTriggerEnter()

Given that the "OnTriggerEnter()" function is used to ensure that the participant's hand is in the same position as the collider of the object as soon as it is touched, it is crucial to prevent the user from grabbing multiple objects at the same time. In order to achieve this, the "OnTriggerStay()" function takes the argument "Collider other" (line 37) and verifies whether the object is already in the correct position as specified in "OnTriggerEnter()" (line 41).

If the object is not yet in the correct position, it is important to ensure that it remains in the same position as the participant's hand (line 47). As mentioned earlier, it is imperative to prevent the user from grabbing more objects while one is already being carried. To accomplish this, the system identifies and places all products in the current level in a list called "objs" (line 49), and all distractors such as boxes and products that cannot be transported in a list called "dist" (line 50). Both lists are then merged into a single list called "final_array" by concatenating them (line 51).

Having a list of all the objects present in the level, the next step is to loop through the list and analyze each element (line 52). A check is then performed to determine which list element is already being transported by comparing the name of each object in the list with the "gameObject", which represents the name of the object that has already been selected (lines 54 and 55). For all cases where the names are different, the "Collider" of these objects is set to false, ensuring that only the object that is currently being moved is set to true, thereby preventing simultaneous movement (line 57).

```
37  public void OnTriggerStay(Collider other)
38  {
39      if (other.tag != "target")
40      {
41          if (gameObject.GetComponent<Item>().dragging == true)
42          {
43              gameObject.transform.position = new Vector3(13.643f, 2.831644f, -1.637f);
44          }
45          else
46          {
47              gameObject.transform.position = new Vector3(other.gameObject.transform.position.x, other.gameObject.transform.position.y, other.gameObject.transform.position.z);
48
49              GameObject[] objs = GameObject.FindGameObjectsWithTag("Product");
50              GameObject[] dist = GameObject.FindGameObjectsWithTag("Distractor");
51              GameObject[] final_array = objs.Concat(dist).ToArray();
52              foreach (GameObject g in final_array)
53              {
54                  nome = g.name;
55                  if (g != gameObject)
56                  {
57                      g.GetComponent<Collider>().enabled = false;
58                  }
59
60              }
61          }
```

Figure 63 - Move .cs, OnTriggerStay()

With the implementation of the "Move.cs" script, it became possible to pick up any object and move it to the appropriate location. The first two levels feature highlighted zones with a distinct white texture, which serve as the designated locations for object placement. Subsequent levels require objects to be placed in shopping baskets or crates, depending on the specific scenario.

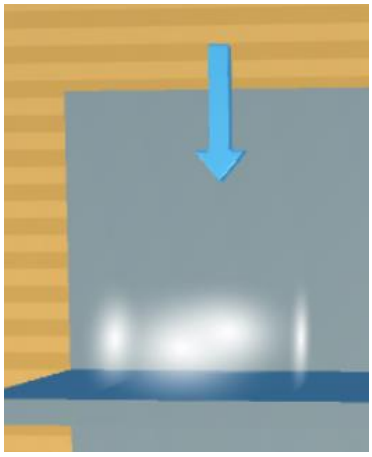


Figure 64 – Target in the first 2 levels

Figure 65 - New targets

The "Collide.cs" script is responsible for detecting whether an object has been correctly placed in the target zone. This script is intended to be called exclusively on target objects, with no need to invoke it for other objects in the environment.

Prior to creating the functions for the script, we began by defining several variables that would be utilized later on, as demonstrated in Figure 66.

```
11      public GameObject rightTarget;
12      public GameObject leftTarget;
13      public GameObject meta;
14      public GameObject FrontScreenInfo;
15      public float price;
16      public float itemPrice;
17      public float newPrice;
18      public GameObject TextBoxL;
19      public GameObject TextBoxR;
20      private int new_placed;
21      private int new_placed1;
22      private int new_placed2;
23      public ParticleSystem collisionParticleSystem;
24      public AudioSource sound;
25      public AudioSource wrong_sound;
```

Figure 66 - Collide.cs, variables

The first two variables, "rightTarget" and "leftTarget", serve to differentiate between the target objects on the right and left sides, respectively. The appropriate GameObject corresponding to each target is then assigned to its respective variable slot. In Figure 67, for example, "rightTarget" refers to the "meta" GameObject, while "leftTarget" refers to the "metaFruit1" GameObject.

Figure 67 - Collide.cs, targets

Then the variable "meta" appears. This is only used to identify the name of the *target* on which *the* "Collider.cs" script is acting. In this way, it is possible, further ahead, to define the right *targets* that objects must collide with.

Figure 68 - Collider.cs, goal

The following variables, namely "FrontScreenInfo", "price", "itemPrice", and "newPrice", are directly related to each other due to their relevance to prices, either for products or for the total purchase value. The connection of "FrontScreenInfo" with the other variables is established through the script "RandomPriceGenerator", which resides in the GameObject "FrontScreenInfo" and contains the variable "TheNumber" indicating the total budget value of the level.

"TextBoxL" and "TextBoxR" variables are solely required for the last two levels of the game, levels 9 and 10, representing the text that will appear on each of the side monitors indicating the budget.

The variables "new_placed", "new_placed1", and "new_placed2" are utilized in three different scripts, namely "Shelf.cs", "Shelf2.cs", and "Shelf5.cs", which aim to randomize the position of each object in the level. The first three variables are responsible for updating the number of correctly placed objects, considering the variable "placed" in each "Shelf.cs" script.

To enhance the dynamism and enjoyment of the game for the participant, some visual and sound effects were added, providing feedback on their progress throughout the game. Thus, three variables were created, one for visual effects and two for sound effects. The first variable utilizes a "ParticleSystem", causing an explosion-like effect with green particles when activated. The remaining sound effect variables are implemented for different scenarios; one for successful participant actions and the other for mistakes made during gameplay.

Once the variables were defined, the programming of object collisions was initiated. In the "Move.cs" script, the "OnTriggerEnter()" function receives the "gameObject" argument that will collide.

```
51      public void OnTriggerEnter(Collider gameObject)
```

Figure 69 - Collide.cs, OnTriggerEnter()

In levels where the user interacts with product budgets and prices, it is essential that the values displayed are constantly updated. Firstly, the "price" variable is assigned the value of the "TheNumber" variable from the "RandomPriceGenerator" script (line 54), which represents the total budget that the user must consider in levels 9 and 10 and is a random number between 15 and 27. The price of each product, defined in the "Item.cs" script for each object, is saved in the "itemPrice" variable (line 55).

As the user places products in the shopping baskets, the "price" variable should decrease, taking into account the price of these products. Therefore, a new variable, "newPrice", is created to update the total budget by subtracting the product price, "itemPrice", from the "price" variable (line 56). To display the budget value in real-time, two "GameObjects" are created to represent the side monitors, "TextBoxL" for the left side, and "TextBoxR" for the right side, which receive the value indicated by the "newPrice" variable (lines 57 and 58). The "price" variable is then updated with the value of "newPrice", and the "TheNumber" variable is set to the value of "price" (lines 59 and 60).

When an object collides with the target, the primary functionality is to remove the object from the environment to prevent the user from interacting with it again. The solution found was to move the object to a position in the environment where it is not visible, as shown in Figure 70.

```
54          price = FrontScreenInfo.GetComponent<RandomPriceGenerator>().TheNumber;
55          itemPrice = gameObject.GetComponent<Item>().price;
56          newPrice = price - itemPrice;
57          TextBoxL.GetComponent<Text>().text = "" + newPrice + "" + "€";
58          TextBoxR.GetComponent<Text>().text = "" + newPrice + "" + "€";
59          price = newPrice;
60          FrontScreenInfo.GetComponent<RandomPriceGenerator>().TheNumber = (int)price;
```

Figure 70 - Collide.cs, prices

When an object is placed in the right place, the counter responsible for updating the number of right actions must be updated to reach the total number of objects present at the level to advance to the next level. Depending on the active level, each time there was a collision, the variable that updates changes.

In the case of level 1 (lines 76 and 77), the previously created variable "new_placed1" was updated, based on the variable "placed" present in the *script* "Shelf.cs", which is initialized to 0, starting to sum 1 each time an object is dropped correctly, as shown in Figure 72 (line 79).

```
74        gameObject.transform.position = new Vector3(13.643f, 2.831644f, -1.637f);
```

Figure 71 - Collide.cs, new object position

If this is level 2, however, the variable that updates the number of objects correctly placed in the *target* is called "new_placed2" and saves the value of the variable "placed" present in the *script* "Shelf2.cs" (lines 80 and 82).

If the level being played is any other level, except for 1 and 2, the variable "placed" is in the "Shelf5.cs" *script* and its value is updated in "new_placed" (line 84).

```
76        Scene scene = SceneManager.GetActiveScene();
77        if (scene.name == "Level 1")
78        {
79            new_placed1 = FrontScreenInfo.GetComponent<Shelf>().placed + 1;
80        }
81        else if (scene.name == "Level 2")
82        {
83            new_placed2 = FrontScreenInfo.GetComponent<Shelf2>().placed + 1;
84        }
85        else new_placed = FrontScreenInfo.GetComponent<Shelf5>().placed + 1;
```

Figure 72 - Collide.cs, Placed

After a collision is made between the object and the *target*, a check is immediately made to compare the *tag* name of the object. If it is "Product" it means that it is a product to put in the shopping basket. On the other hand, if it is "Distractor", it represents the boxes that have to be dropped into crates.

```
91    if (gameObject.tag == "Product") {
92
93        if ((gameObject.GetComponent<Item>().rigthTarget == true) && meta.gameObject.name == "meta" )
94        {
95            gameObject.GetComponent<Item>().dragging = true;
96            gameObject.GetComponent<Item>().square.color = Color.green;
97
98            ColliderOn(gameObject);
99
100
101       }
102       else if ((gameObject.GetComponent<Item>().leftTarget == true) && meta.gameObject.name == "metaFruit1")
103       {
104
105           gameObject.GetComponent<Item>().dragging = true;
106           gameObject.GetComponent<Item>().square.color = Color.green;
107           ColliderOn(gameObject);
108
109       }
110       else
111       {
112           gameObject.GetComponent<Collider>().enabled = false;
113
114
115           ColliderOnWrong(gameObject);
116           gameObject.transform.position = gameObject.GetComponent<Item>().position;
117
118       }
119   }
```

Figure 73 - Collide.cs, product

Once verification has been completed, it is necessary to determine whether the product that collided with the target is located on the correct side of the environment. This is to ensure that a product on the right side cannot be dropped onto a target on the left side. If the product variable "rightTarget" is true and the variable "meta," which indicates the placement location, is "meta," then there is no error (line 93). The same principle applies to the left side; if the product's "leftTarget" variable is true and the variable "meta" is "metaFruit1," it is also correct (line 102). When the checks indicate that there are no incorrect collisions, the "dragging" variable is set to true (lines 95 and 105), indicating that a product has been dropped in the correct position. The variable "square," which is also associated with the products, changes as well, and its color changes from red to green (lines 96 and 106).

If the object that the participant is grabbing is a "Product," the same process occurs, with the only difference being the names of the targets. For products, the targets on the right and left sides are called "meta" and "metaFruit1," respectively.

When it comes to boxes, the target on the right is named "metaBox2," and the one on the left side is called "metaBox," as shown in Figure 74.

```
161         if (gameObject.tag == "Distractor")
162         {
163             if ((gameObject.GetComponent<Item>().rigthTarget == true) && meta.gameObject.name == "metaBox2")
164             {
165                 gameObject.GetComponent<Item>().dragging = true;
166                 gameObject.GetComponent<Item>().square.color = Color.green;
167                 ColliderOn(gameObject);
168 
169             }
170             else if ((gameObject.GetComponent<Item>().leftTarget == true) && meta.gameObject.name == "metaBox")
171             {
172 
173 
174                 gameObject.GetComponent<Item>().dragging = true;
175                 gameObject.GetComponent<Item>().square.color = Color.green;
176                 ColliderOn(gameObject);
177             }
178             else
179             {
180 
181 
182                 gameObject.GetComponent<Collider>().enabled = false;
183                 ColliderOnWrong(gameObject);
184                 gameObject.transform.position = gameObject.GetComponent<Item>().position;
185 
186             }
187         }
188         }
```

Figure 74 - Collide.cs, Distractor

After such variables change, the "ColliderOn()" function is invoked, which receives as an argument the "gameObject" that is colliding.

```
216     void ColliderOn(Collider gameObject)
217     {
218         Scene scene = SceneManager.GetActiveScene();
219         var em = collisionParticleSystem.emission;
220         GameObject[] prod = GameObject.FindGameObjectsWithTag("Product");
221         GameObject[] dist = GameObject.FindGameObjectsWithTag("Distractor");
222         foreach (GameObject g in prod )
223         {
224             g.GetComponent<Collider>().enabled = true;
225             gameObject.transform.position = new Vector3(13.643f, 2.831644f, -1.637f);
226 
227             em.enabled = true;
228             collisionParticleSystem.Play();
229             sound.Play();
230 
231             if (scene.name == "Level 1")
232             {
233                 FrontScreenInfo.GetComponent<Shelf>().placed = new_placed1;
234             }
```

Figure 75 - Collide.cs, ColliderOn()

When the "ColliderOn()" function is called, the intention is for the product to disappear, ensuring that its position becomes hidden and is not visible within the displayed environment. Once this occurs, sound and visual effects are triggered.

In this function, we first detect the name of the active scene (line 218) and define a variable named "em," which stands for "emission," as the particle system starts playing when the action is successfully performed (line 219).

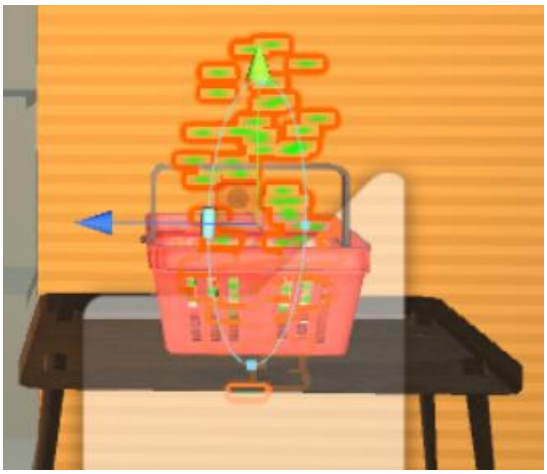


Figure 76 - Particle System

As previously mentioned, when the participant interacts with a product, they cannot move any others simultaneously. However, after placing a product in the correct position, they can grab another. To facilitate this, all objects with the "Product" tag are placed in a list called "prod" (line 220), and all objects tagged "Distractor" are placed in a list called "dist" (line 221). For each element in these lists, their "Collider" becomes true (lines 222 and 224), and the position of the product placed in the correct position becomes hidden (line 225).

Once this is done, the sound and visual effects are triggered. The previously created "em" variable is set to true (line 227), and the particle system "collisionParticleSystem" is initiated using the command ".Play" (line 228), along with the hit sound "sound" (line 229).

To update the number of objects placed in the correct position, the scene name is used to update the variable "placed" in the "Shelf.cs" script to the value of the new variable "new_placed1" for level 1, and the same for other levels, with the variable names changed accordingly.

As shown in Figure 73, line 110, there is also the case where the participant places a product meant for the left side on a target on the right side. In this case, when a product is put in the wrong position, its "Collider" becomes false (line 112), and a new function called "ColliderOnWrong()" is invoked (line 115).

```
265     void ColliderOnWrong(Collider gameObject)
266     {
267         Scene scene = SceneManager.GetActiveScene();
268         var em = collisionParticleSystem.emission;
269         GameObject[] prod = GameObject.FindGameObjectsWithTag("Product");
270         GameObject[] dist = GameObject.FindGameObjectsWithTag("Distractor");
271         foreach (GameObject g in prod)
272         {
273
274             g.GetComponent<Collider>().enabled = true;
275             gameObject.transform.position = new Vector3(13.643f, 2.831644f, -1.637f);
276             wrong_sound.Play();
277
278
279         }
280         foreach (GameObject g in dist)
281         {
282
283             g.GetComponent<Collider>().enabled = true;
284             gameObject.transform.position = new Vector3(-13.643f, 2.831644f, -1.637f);
285             wrong_sound.Play();
286
287         }
288     }
```

Figure 77 - Collide.cs, ColliderOnWrong()

The function "ColliderOnWrong()" bears significant resemblance to "ColliderOn()" which was previously explained, albeit with some differences. Similar to "ColliderOn()", this function also populates a list with all objects tagged as "Product" (line 269), and another list with objects tagged as "Distractor" (line 270). For each element present in the "Product" list, its "Collider" attribute is set to true (line 274), and the object is moved to a location outside the environment (line 275). The difference here is the sound effect that is played - in this case, as it signifies an incorrect interaction by the participant, a sound of failure is played (line 276), which is also played if the object is a "Distractor".

When the function completes its interactions, it returns to the "else" condition (presented in Figure 73, line 116). If the object is placed in a location that is not the correct one, it returns to its initial position so that the participant can have another chance to move it to the correct target. Each object has a variable that stores its initial position in the level, labeled "position". The new position of an object that was incorrectly placed goes back to the position indicated in that variable.

As previously explained, when all objects in the level are correctly placed, the level ends and advances to the next. To enable this functionality, the script "LevelEnding.cs" was created. This script creates a canvas that displays a message indicating the completion of the level and informs the participant that the next level will start in 3 seconds, as shown in Figure 78.

Figure 78 - LevelEnding Canvas

This *Canvas* that was created is used in this *script*, and a variable called "Level1Ending" is defined, and this is where this *Canvas* is called to appear in the game.

Another variable that was created was the "prod_number". This will simply indicate the total number of objects that are present at the level, to subsequently compare with the number of objects being placed in the *targets* to see if all objects have already been moved.

For this comparison, the variable "placed" is retrieved, present in the GameObject "FrontScreenInfo", so we can compare how many objects are placed in the correct target.

```
12      public GameObject Level1Ending; // Assign in inspector
13      private int prod_number;
14      public GameObject FrontScreenInfo;
```

Figure 79 - LevelEnding.cs, Variables

First, this script looks for how many objects there are at the level being played. Thus, in the function "Start()", 2 lists called "objs" and "dist" were created, to find all objects at the level that had the *tags* "Product" and "Distractor", respectively (lines 19 and 20). These last objects tagged “Distractor" will only be used at later levels. Once all the objects are in each of the created lists, lists are merged to obtain a single one with all the objects that were at the level, as is done in other *scripts* previously mentioned. To do so, the 2 lists were concatenated in a "final_array" (line 21).

Once there was only 1 list with all the objects, the only thing missing was knowing how many elements existed in that same list. For this, the variable "prod_number" corresponds to the size of the "final_array" list (line 22).

```
17      void Start()
18      {
19          GameObject[] objs = GameObject.FindGameObjectsWithTag("Product");
20          GameObject[] dist = GameObject.FindGameObjectsWithTag("Distractor");
21          GameObject[] final_array = objs.Concat(dist).ToArray();
22          prod_number = final_array.Length;
23      }
```

Figure 80 - LevelEnding.cs, Start()

Once the number of objects present in the level is known, a comparison can be made with the number of objects that have been placed in the targets thus far. To accomplish this, a check is performed to determine which scene the game is in (Figure 81, lines 31 and 32). If the participant is in level 1, the variable "placed" in the "Shelf.cs" script (which only exists for level 1) is accessed. If the number indicated by the "placed" variable is greater than or equal to "prod_number", it indicates that the level has ended (line 34). In other words, if the number of objects placed in the targets is equal to or greater than the total number of objects in the level.

If the game is at Level 2, the same comparison as in Level 1 is performed, but the variable "placed" is located in the script "Shelf2.cs" on line 45. The same reasoning applies to Level 6, but the variable "placed" is located in the script "Shelf5.cs".

Starting from Level 7, interactions become more distinct. In Level 7 itself, the check to determine if the level has been completed differs from the previous levels. As depicted in Figure 82 below, for the level to end, the value of the variable "placed" must be equal to or greater than "prod_number - 4" on line 62. This is because Level 7 has several products, four of which should not be placed in the

```
29      void Update()
30      {
31          Scene scene = SceneManager.GetActiveScene();
32          if (scene.name == "Level 1")
33          {
34              if (FrontScreenInfo.GetComponent<Shelf>().placed >= prod_number)
35              {
36
37
38                  StartCoroutine(Time());
39                  Level1Ending.SetActive(true);
40              }
41          }
42
43          else if (scene.name == "Level 2")
44          {
45              if (FrontScreenInfo.GetComponent<Shelf2>().placed >= prod_number)
46              {
47                  StartCoroutine(Time());
48                  Level1Ending.SetActive(true);
49              }
50          }
51          else if (scene.name == "Level 6")
52          {
53              if (FrontScreenInfo.GetComponent<Shelf5>().placed >= prod_number)
54              {
55                  StartCoroutine(Time());
56                  Level1Ending.SetActive(true);
57              }
58          }
```

Figure 81 - LevelEnding.cs

shopping baskets, and therefore cannot be counted, hence the subtraction of four.

If the game is at Level 8, the same concept applies as in Level 7. If the variable "placed" is equal to or greater than "prod_number - 10", the level ends on line 71. This is because there are 14 products in the level, but only four are to be dragged, leaving the remaining ten without effect.

In Levels 9 and 10, the concept of budget is introduced. The participant's goal is to place the products in the shopping baskets based on their prices until the budget decreases to zero. This is precisely the check made to advance the level. If the value of the variable "TheNumber" in the "RandomPriceGenerator" script is equal to zero, the game proceeds to the next level. However, if the value is less than zero due to participant miscalculations, the level resumes on lines 87 and 100.

If the level is not one of those mentioned above, the game behaves the same as in Level 6 on line 104.

```
60          else if (scene.name == "Level 7")
61          {
62              if (FrontScreenInfo.GetComponent<Shelf5>().placed >= prod_number-4)
63              {
64                  StartCoroutine(Time());
65                  Level1Ending.SetActive(true);
66              }
67          }
68
69          else if (scene.name == "Level 8")
70          {
71              if (FrontScreenInfo.GetComponent<Shelf5>().placed >= prod_number-10)
72              {
73                  StartCoroutine(Time());
74                  Level1Ending.SetActive(true);
75              }
76          }
77
78          else if (scene.name == "Level 9")
79          {
80              if (FrontScreenInfo.GetComponent<RandomPriceGenerator>().TheNumber == 0)
81              {
82                  StartCoroutine(Time());
83                  Level1Ending.SetActive(true);
84              }
85              else if (FrontScreenInfo.GetComponent<RandomPriceGenerator>().TheNumber < 0)
86              {
87                  SceneManager.LoadScene("Level 9");
88              }
89          }
90
91          else if (scene.name == "Level 10")
92          {
93              if (FrontScreenInfo.GetComponent<RandomPriceGenerator>().TheNumber == 0)
94              {
95                  StartCoroutine(Time());
96                  Level1Ending.SetActive(true);
97              }
98              else if (FrontScreenInfo.GetComponent<RandomPriceGenerator>().TheNumber < 0)
99              {
100                 SceneManager.LoadScene("Level 10");
101             }
102         }
103         else
104             if (FrontScreenInfo.GetComponent<Shelf5>().placed >= prod_number)
105             {
106                 StartCoroutine(Time());
107                 Level1Ending.SetActive(true);
```

Figure 82 - LevelEnding.cs, Levels

To make the level automatically advance to the next, a "Coroutine" was used, where there was a 5-second wait until it is switched to the next level of the game (Figure 83). After the "StartCoroutine()" function is invoked in Update(), the *Canvas* that indicates that the level has been completed appears to inform the participant of the same.

```
116     IEnumerator Time()
117     {
118
119         yield return new WaitForSeconds(5);
120
121         SceneManager.LoadScene(SceneManager.GetActiveScene().buildIndex + 1);
122     }
```

Figure 83 - IEnumerator Time()

The context in which "Shelf2.cs" is used is the same as “Shelf.cs”, where the goal is to organize products and *targets* so that their position on the shelves is random. The difference between "Shelf.cs" and "Shelf2.cs" is the arrangement of objects. While in the first, the products and *targets* would have to be on the same shelf, in the second, these are on different shelves, so the code implemented is different.

```
6   public class Shelf2 : MonoBehaviour
7   {
8       public List<Vector3> rightPoslistUp = new List<Vector3>();
9       public List<Vector3> leftPoslistUp = new List<Vector3>();
10      public List<Vector3> rightPoslistDown = new List<Vector3>();
11      public List<Vector3> leftPoslistDown = new List<Vector3>();
12      public int placed;
13      public GameObject meta;
14      public GameObject metaFruit1;
```

Figure 84 - Shelf2.cs, Variables

Starting with the variables created, it is evident that there are more variables compared to the script "Shelf.cs" of level 1. The script defines position variables for the products, namely "rightPoslistUp" and "leftPoslistUp", and for the targets, namely "rightPoslistDown" and "leftPoslistDown". The variables "rightPoslistUp" and "leftPoslistUp" indicate positions relative to the top shelf, while the variables "rightPoslistDown" and "leftPoslistDown" correspond to positions on shelves further down.

Similar to other scripts, the variable "placed" is used to count the number of products placed in the right target. The last two variables, "meta" and "metaFruit1", represent the existing targets on the right and left sides, respectively.

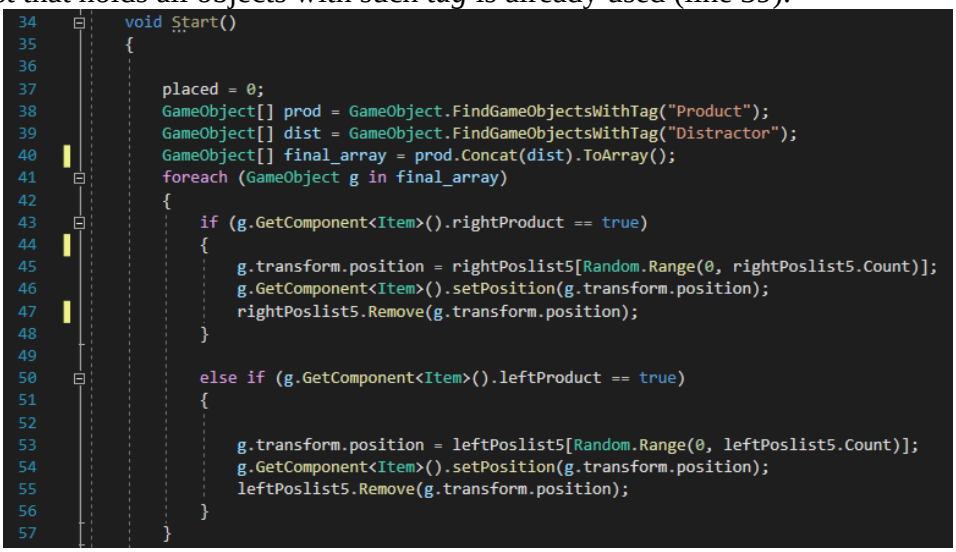
After the variables were created, the arrangement of products on shelves was implemented. As in level 1, all objects with the tag "Product" are placed in a list named "prod" (line 23). Once all objects are identified, each product is checked to determine if it belongs to the right or left side. If the variable "rightProduct" associated with the product is true, the product belongs to the right side (line 26). The product is then randomly placed on one of the predefined positions on the top shelf, as indicated by the "rightPoslistUp" list (lines 28 and 29). After placing the product, the new position is removed from the list of available positions (line 30). Similarly, the "meta" target is placed in a random position on another shelf below, as defined in the "rightPoslistDown" list (line 31).

If the product belongs to the left side (line 34), the code is the same, except for the variable names. Instead of using "right" in their names, the variables use "left". Similarly, the target is renamed as "metaFruit1".

```
21      void Start()
22      {
23          GameObject[] prod = GameObject.FindGameObjectsWithTag("Product");
24          foreach (GameObject g in prod)
25          {
26              if (g.GetComponent<Item>().rightProduct == true)
27              {
28                  g.transform.position = rightPoslistUp[Random.Range(0, rightPoslistUp.Count)];
29                  g.GetComponent<Item>().setPosition(g.transform.position);
30                  rightPoslistUp.Remove(g.transform.position);
31                  meta.transform.position = rightPoslistDown[Random.Range(0, rightPoslistDown.Count)];
32              }
33
34              else if (g.GetComponent<Item>().leftProduct == true)
35              {
36
37                  g.transform.position = leftPoslistUp[Random.Range(0, leftPoslistUp.Count)];
38                  g.GetComponent<Item>().setPosition(g.transform.position);
39                  leftPoslistUp.Remove(g.transform.position);
40                  metaFruit1.transform.position = leftPoslistDown[Random.Range(0, leftPoslistDown.Count)];
41              }
42          }
```

Figure 85 - Shelf2.cs, Start()

Shelf5.cs is implemented in the same way that it was used in the "Shelf.cs" and "Shelf2.cs" but in this case, it will already be used at levels where there are boxes, i.e. objects with the *tag* “Distractor", where the list that holds all objects with such t*ag* is already used (line 39).



```
34      void Start()
35      {
36
37          placed = 0;
38          GameObject[] prod = GameObject.FindGameObjectsWithTag("Product");
39          GameObject[] dist = GameObject.FindGameObjectsWithTag("Distractor");
40          GameObject[] final_array = prod.Concat(dist).ToArray();
41          foreach (GameObject g in final_array)
42          {
43              if (g.GetComponent<Item>().rightProduct == true)
44              {
45                  g.transform.position = rightPoslist5[Random.Range(0, rightPoslist5.Count)];
46                  g.GetComponent<Item>().setPosition(g.transform.position);
47                  rightPoslist5.Remove(g.transform.position);
48              }
49
50              else if (g.GetComponent<Item>().leftProduct == true)
51              {
52
53                  g.transform.position = leftPoslist5[Random.Range(0, leftPoslist5.Count)];
54                  g.GetComponent<Item>().setPosition(g.transform.position);
55                  leftPoslist5.Remove(g.transform.position);
56              }
57          }
```

Figure 86 - Shelf5.cs

Allow me to present the Identify.cs script, which is utilized solely in level 6 and eliminates the need for the user to move objects from one location to another. Its primary function is to recognize products upon the user's hand being held near them for a duration of 1 second. This script is exclusively invoked in products and not in targets.

The Identify.cs script comprises several variables. The first variable corresponds to a "SpriteRenderer" in the shape of a "square" (line 9), which later changes color depending on the outcome of the activities. The timer is set at 1 second (line 10), and a bool "countdown" is initialized as false (line 11). Similar to other scripts, particle systems (line 12), and two sounds (lines 13 and 14) are created. Additionally, a variable named "new_place" is established (line 16) to update the number of dropped products as they are placed correctly.

To update the "new_placed" variable, we must work with the "placed" variable present in FrontScreenInfo. Thus, we refer to the game object as FrontScreenInfo.

```
7   public class Identify : MonoBehaviour
8   {
9       public SpriteRenderer square;
10      float timer = 1f;
11      bool countdown = false;
12      public ParticleSystem collisionParticleSystem;
13      public AudioSource sound;
14      public AudioSource wrong_sound;
15      public GameObject FrontScreenInfo;
16      private int new_placed;
```

Figure 87 - Identify.cs, Variables

When the user approaches his hand to a product, the boolean "countdown" becomes *true (line* 21), *indicating* that *the timer* can start counting, because while *false*, the timer does not take action. After this happens, the value of finished objects at the level is incremented (line 27).

```
19      void OnTriggerStay(Collider other)
20      {
21          countdown = true;
22
23
24      }
        0 references
25      void OnTriggerEnter(Collider other)
26      {
27          new_placed = FrontScreenInfo.GetComponent<Shelf5>().placed + 1;
28      }
29
```

Figure 88 - Identify.cs, Collisions

As soon as the person *approaches the* product, the particle system is activated (line 32). If the countdown is *true* (line 33), *the* 1-second timer starts to countdown (line 35) and a sound is heard to imply that the product has been found (line 37).

When the *timer* equals or exceeds 0 (line 39), it means that the product has been correctly identified and its texture disappears, becoming transparent (line 41). As this product has been correctly discovered, the color of the square changes to green (line 42) and the sound of success is heard (line 45). The variable "placed" is then updated with the value in the "new_placed" that increases, as has already been said, for each product that is successfully discovered (line 46).

```
30      void Update()
31      {
32          var em = collisionParticleSystem.emission;
33          if (countdown)
34          {
35              timer -= Time.deltaTime;
36
37              wrong_sound.Play();
38          }
39          if (timer <= 0)
40          {
41              gameObject.GetComponent<MeshRenderer>().enabled = false;
42              square.color = Color.green;
43
44
45              sound.Play();
46              FrontScreenInfo.GetComponent<Shelf5>().placed = new_placed;
47
48          }
49
50      }
```

Figure 89 - Identify.cs

If the user is close to the product, but suddenly, moves away and it is no longer found, the function "OnTriggerExit()" sets the *countdown* to *false* again (line 54) and *resets* the *timer* (line 55), waiting for the user to approach the product again to resume counting.

```
52      void OnTriggerExit(Collider other)
53      {
54          countdown = false;
55          timer = 1f;
56
57      }
```

Figure 90 - Identify.cs, OnTriggerExit()

As previously mentioned, certain levels incorporate a budget and product pricing, requiring the use of the RandomPriceGenerator.cs script. This script controls the pricing dynamics of such levels. Additionally, these levels feature two side monitors, one on the left and one on the right, which display text represented by variables "TextBoxL" and "TextBoxR" (lines 8 and 9), respectively.

The "TheNumber" variable (line 10) corresponds to the budget set at the onset of the level, with the participant's gameplay relying on this value.

```
6   public class RandomPriceGenerator : MonoBehaviour {
7
8       public GameObject TextBoxL;
9       public GameObject TextBoxR;
10      public int TheNumber;
```

Figure 91 - RandomPriceGenerator.cs, Variables

The "RandomGenerator" function sets the value of "TheNumber" by generating a random number between 15 and 27 (line 20). To display this value in the environment, the two variables are set to showcase the budget on the side screens (lines 21 and 22). The budget range is specifically set between 15 and 27, as numerous trials indicate that the minimum value that an individual could engage in the activity using the two lateral walls was 15. On the other hand, the maximum budget is set at 27 since, upon analyzing the prices of the products and the required participant movements, a budget of 27 allowed for multiple hypotheses of products to be incorporated.

```
13      void Start()
14      {
15          RandomGenerator();
16
17      }
        1 reference
18      public void RandomGenerator()
19      {
20          TheNumber = Random.Range(15, 27);
21          TextBoxL.GetComponent<Text>().text = "" + TheNumber + "" + "€";
22          TextBoxR.GetComponent<Text>().text = "" + TheNumber + "" + "€";
23
24      }
25  }
```

Figure 92 - RandomPriceGenerator.cs

As the levels pass, the difficulty increases. There are those where the information and shopping lists are visible to the participant and there are others where that information is visible for a certain time until they disappear and force the user to further strive their cognitive ability.

The *script* WaitForTimeList.cs is responsible for controlling how long the information is visible to the participant until it disappears, so the only function used is a Coroutine that waits 22 seconds (line 21) until the "GameObject" for the information is no longer visible (line 23).

```
5   public class WaitForTimeList : MonoBehaviour
6   {
7
8       // Use this for initialization
        0 references
9       void Start()
10      {
11          StartCoroutine(Test());
12      }
13
14      // Update is called once per frame
        0 references
15      void Update()
16      {
17
18      }
        1 reference
19      IEnumerator Test()
20      {
21          yield return new WaitForSeconds(22);
22
23          gameObject.SetActive(false);
24      }
25  }
26
```

Figure 93 - WaitForTimeList.cs

4. User tests

To accurately evaluate the quality of the developed project and determine its usability, usability tests are essential. These tests involve having participants play the developed game to assess whether they can complete it and identify any errors made during the experiment.

The methodology used for the test sessions, as well as the profiles of the participants and the results of the tests, are presented below.

4.1 Methodology

To conduct the test sessions, participants were required. Priority was given to those who attended the focus group session held previously (Chapter XX). These individuals, along with other members of the NeuroRehabLab group and classmates, were invited to participate.

Once enough volunteers were secured, sessions were scheduled between November 23 and December 6, 2022, based on the availability of each participant.

4.2 Participants

The usability tests involved 11 participants, 7 of whom were male. Demographic information about the participants is presented in Table 8. The participants' ages ranged from 20 to 33 years (Mean= 25, SD= 4.22). Among the participants, 63% had a Master's degree, 28% had completed secondary education, and only 9% had an undergraduate degree. Participants included students, researchers, research assistants, and psychologists. The table demonstrates that there was a diverse range of participants in terms of profession, educational degrees, and gender.

Table 8 - User Information

ID	Gender	Age	Dominant hand	Education	Job
1	Male	20	Right	Secondary	Student
2	Male	20	Rigth	Secondary	Student
3	Male	20	Rigth	Secondary	Student
4	Male	23	Rigth	Master's degree	Research Assistent
5	Female	29	Rigth	Master's degree	Research Assistent
6	Female	24	Rigth	Graduation	Student
7	Female	33	Rigth	Master's degree	Researcher
8	Male	25	Left	Master's degree	Research
9	Male	29	Rigth	Master's degree	PhD Student/Researcher
10	Female	29	Rigth	Master's degree	Psychologist
11	Male	23	Rigth	Master's degree	Student/research assistant

4.3 Procedure

A script was created to guide all the user test sessions, where it explained everything that would happen during it (Appendix D).

This document was divided into 6 parts:

1- Informed Consent

This corresponds to a document where all the conditions for the course of user test sessions are described, and where the user can accept or not the collection of photographs during the study (Appendix E).

2- Context Description

This part only explained the context in which the project was developed, stating that it is a game to rehabilitate people who have suffered a stroke while stimulating the person's physical and cognitive abilities.

3- Game Description

In this part are explained the game mechanics and how to play. It is mentioned that the game has a supermarket environment and that it will have 10 levels with an increase in difficulty as they progress.

Then the participant is given information about the objective of each level so that he/she has already an idea of what is going to happen during the play session.

4- User plays the game

At this time, the participant plays the 10 levels of the game, which lasts approximately 20 minutes.

5- Distribution of the questionnaires

After completing the game, participants were required to complete a series of questionnaires aimed at providing feedback on their experience. Six questionnaires were utilized

in total. The first questionnaire collected user demographic information, which was used to compare and analyze data between participants, as presented in Table 8 [Appendix J].

A Customized Questionnaire was then created specifically for this project, containing questions about the participant's opinions on various aspects of the game. This questionnaire was tailored to the project's specific needs and is distinct from the subsequent questionnaires that are commonly used. It was based on the three main questions established at the beginning of the project's development, as mentioned in the "5.2 The Idea" chapter of the document [Appendix K].

In addition, the USEQ - User Satisfaction Evaluation Questionnaire [28] was administered to evaluate the participants' satisfaction with the game [Appendix H]. This questionnaire consists of 6 questions with 5 possible answers each, rated on a Likert Scale. The minimum score for the USEQ is 0 and the maximum is 30. The objective of the USEQ is to assess participants' satisfaction with the game.

The Presence Questionnaire [29] was also utilized to evaluate the level of immersion and the participants' ability to interact with the game [Appendix F]. The aim was to assess the participant's sense of presence during the game, specifically, whether the participant felt involved in the virtual environment, whether they were able to complete the required tasks successfully, and whether the activities in the game were relevant to real-life situations. The Presence Questionnaire consists of 24 questions with answer options corresponding to a 7-point scale. The 24 questions are divided into several themes, as illustrated below.

« Realism » : Items 3 + 4 + 5 + 6 + 7 + 10 + 13
« Possibility to act » : Items 1 + 2 + 8 + 9
« Quality of interface » : Items (all reversed) 14 + 17 + 18
« Possibility to examine » : Items 11 + 12 + 19
« Self-evaluation of performance » : Items 15 + 16
« Sounds* » : Items 20 + 21 + 22
« Haptic* » : Items 23 + 24

With this division it is possible to see in more detail the aspects of presence in the game in which the participants were more and less comfortable. In this case, the themes "Sounds" and "Haptic" were not considered, since these 2 are not considered in some versions of the questionnaire to analyze.

The System Usability Scale (SUS) [30] which is one of the best known and simplest methods to ascertain the usability of the system [Appendix G]. This questionnaire consists of a set of 10 questions with 5 answer options in the form of scale, from Strongly Disagree to Strongly Agree.

After the participant answers all the questions present in the SUS, a total score is calculated based on the score given to the odds and even questions.

SUS Score	Grade	Adjective Rating
> 80.3	A	Excellent
68 – 80.3	B	Good
68	C	Okay
51 – 68	D	Poor
< 51	F	Awful

Figure 94 – SUS classification

Figure 94 illustrates how the scoring system works and it's ranking. In ideal cases, a system should have a score above 80.3 to achieve the highest possible level of classification in this questionnaire.

The last questionnaire that the participant had to answer was the Virtual Environment Verisimilitude Questionnaire, a custom questionnaire created by the NeuroRehabLab team. With this questionnaire, the goal is to assess the degree of realism or similarity with a real environment, that is, how real the game itself was represented, taking into account the environment and tasks to be performed [Appendix I]. This questionnaire consists of 14 questions with 5 options in the form of a scale, such as the other questionnaires, and in the end, by summing the answers to all the questions for each user, a certain score is obtained.

The maximum score that can be achieved is 70, thus corresponding to the perfect value, meaning that the system is completely adequate with reality.

6- Distribution of the compensation

Chocolates were distributed to the participants by finishing the user testing sessions in the form of a thank you session.

4.4 Game Testing

After completing the necessary procedures, which included providing an explanation of the game's context and mechanics and obtaining informed consent from the participant, the game test began.

Participants had the opportunity to experience the game at their own pace, taking as much time as necessary to complete the levels. Feedback was also provided through questionnaires following the completion of all levels.

Figure 95 – Game Interaction

Above we can see a photograph that was taken during the test sessions of a participant, where we can understand how the interactions work. There, the participant is grabbing a product and moves it to the selected *target.*

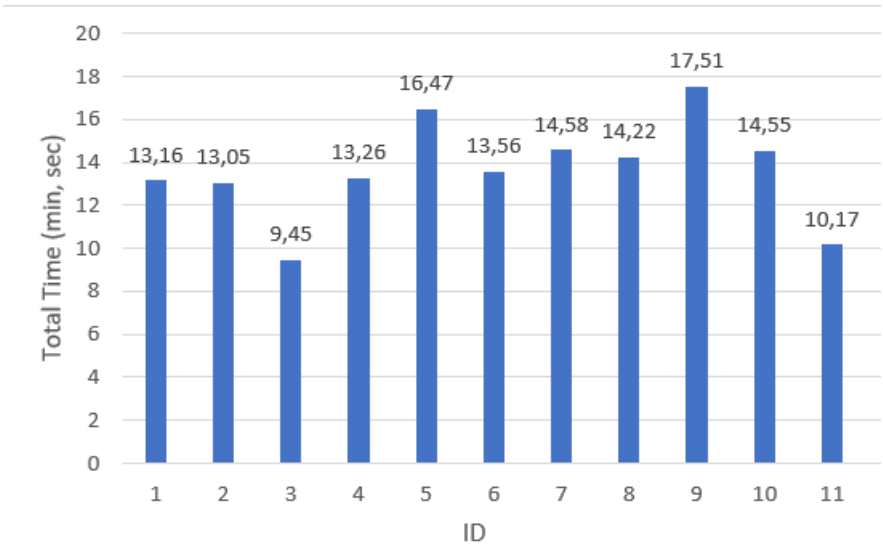
4.5 Results

As previously mentioned, each test session concluded with the participant providing feedback via questionnaires. Once all participants had completed the questionnaires, the results were analyzed using Microsoft Excel. This software enabled the generation of tables and graphs depicting the participants' answers.

It is worth noting that some of the questionnaires used in this project had specific metrics for measuring results, which was important for our analysis. The resulting statistics are presented below.

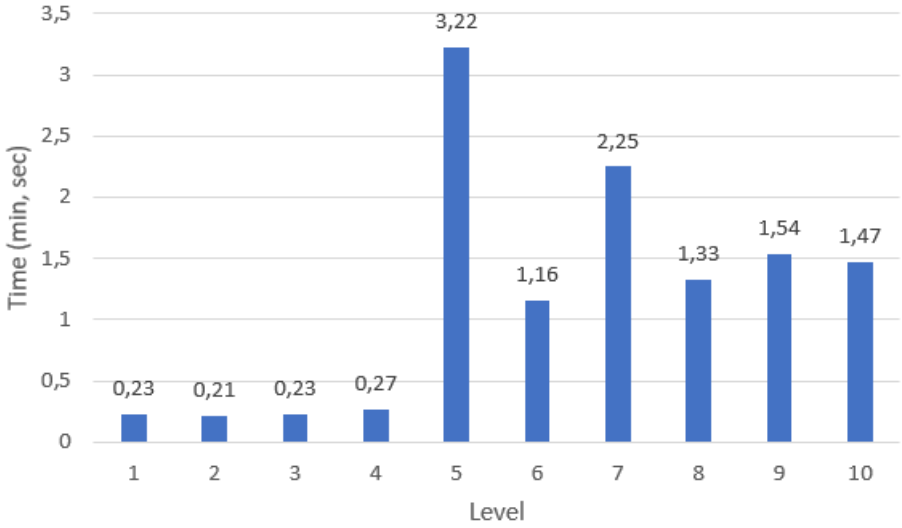
The time it took for each participant to complete the game was influenced by various factors, including motor and cognitive abilities, as well as age. In our test sessions, participants' ages ranged from 20 to 33 years old, but there was still a small degree of variability. Therefore, the time it took for each individual to play the game depended on their speed in performing actions and comprehending each level.

Table L in Appendix L presents the times it took for each participant to complete each of the 10 levels of the game.



Graph 2 – Game time per participant

From Graph 1, we can observe the time that each participant took to complete the game. The fastest participant completed the game in 9 minutes and 45.34 seconds, while the slowest one took 17 minutes and 51.10 seconds to finish. On average, the 11 participants took 13 minutes and 53.07 seconds to complete the game, which falls within the expected range of game completion time.

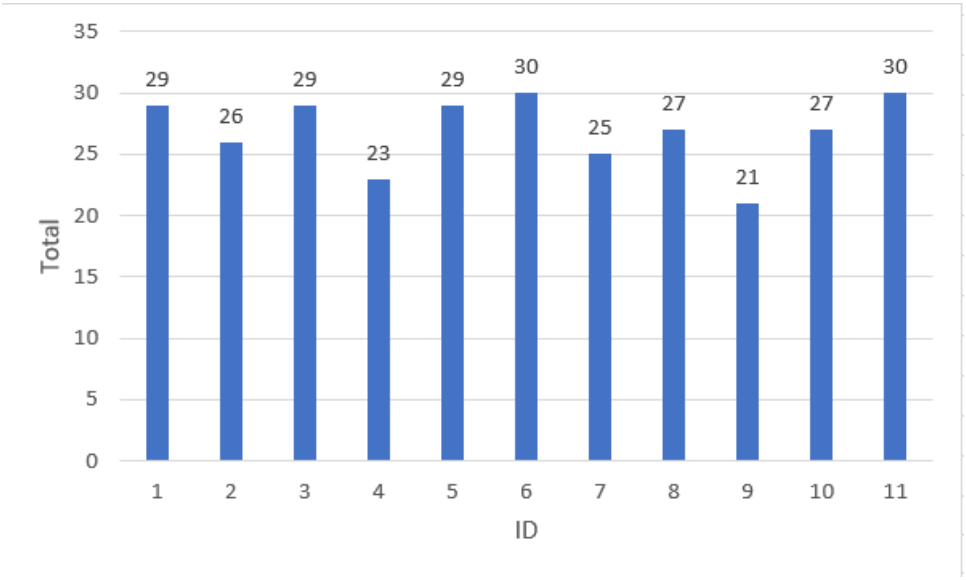


Graph 1 – Mean time per level

The duration of each participant's performance on each level was recorded, enabling us to determine that the simplest level for users was level 2, which had the lowest average time of 00:21.52s. This finding is supported by Graph 2, which illustrates the average completion time of each level by the 11 participants. Level 2 had the shortest completion time because it required participants to only place two products in a crate, making it a straightforward and quick task.

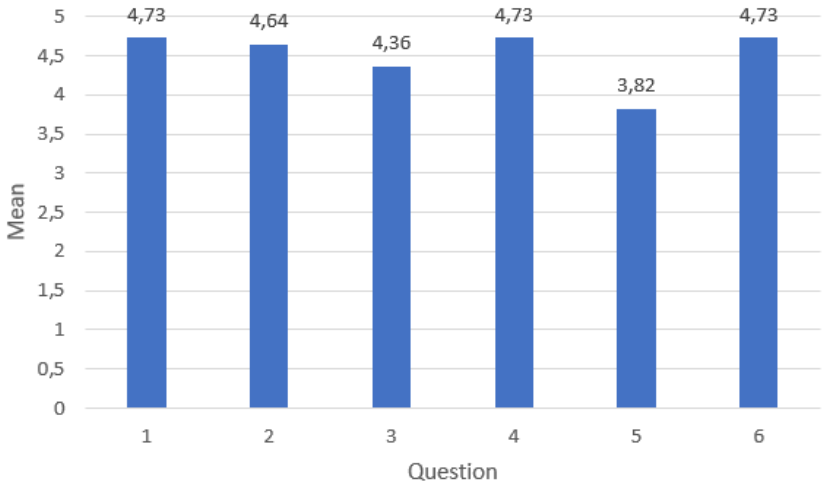
Conversely, level 5 was the most challenging level and took the longest time to complete, with an average time of 3:22.05s. This level took significantly longer because it involved numerous products on shelves that needed to be placed either in crates or shopping baskets

The first questionnaire completed by the participants was the USEQ.



Graph 3 – USEQ results for each participant

Overall, the scores obtained by the participants were high, with the lowest score being 21 and two participants achieving a maximum score of 30. The average score was 26.91, indicating that users experienced a satisfactory level of enjoyment while playing the game. The questionnaire utilized a Likert Scale of 5 hypotheses and consisted of six questions. Out of the six questions, only the fifth question received negative responses, which required a distinct approach to analyzing the results. To derive the results presented in Graph 3, the values of each question's responses were added. For instance, if a user evaluated a question with a 5, then the value added to the total score was 5. However, in the case of question 5, the calculation was different. Here, the value to add to the total score corresponded to 6 minus the score obtained in question 5 (e.g., if the user's response was 2, then the value added to the total score was 4). Furthermore, the means of the responses to each question were also examined and are displayed in the following graph.



Graph 4 - USEQ score's mean for each question

Graph 4 displays that the average responses to all questions, except question 5, were consistently close to 5, which indicates high satisfaction among the participants. Question 5 had an average score of 3.82, suggesting a lower level of satisfaction regarding that particular question.

In general, the analysis of the participants' responses indicates that the game successfully satisfied their expectations and provided a satisfying experience while performing the tasks.

Following each session, the participants completed the Presence Questionnaire, and the results are presented below.

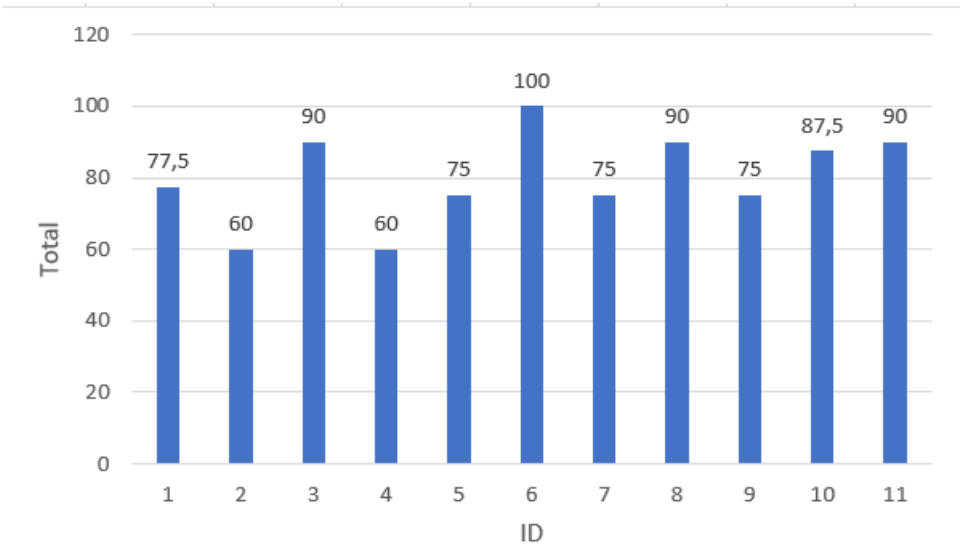
Table 9 - Presence Questionnaire results

ID	Total + Sounds + Haptics	"Realism"	"Possibility to act"	"Quality of interface"	"Possibility to examine"	"Self-evaluation of perfomance"
1	133	41	23	7	19	13
2	123	37	23	14	16	10
3	136	37	26	12	17	12
4	103	29	17	11	15	9
5	133	40	27	4	21	13
6	148	49	27	3	21	13
7	122	34	21	10	15	11
8	139	47	25	12	11	14
9	106	31	18	10	17	9
10	111	31	23	7	17	11
11	129	44	24	13	13	13

Based on Table 9, we can observe the obtained results. The highest possible score for the total score was 168, which represents a perfect value corresponding to an excellent sense of presence. The highest score achieved was 148, while the lowest was 103. The mean score attained was approximately 126, indicating a positive outcome.

In addition to the total score, Table 9 also displays the scores obtained for each theme presented in the questionnaire, except for "Sounds" and "Haptics", which were previously mentioned.

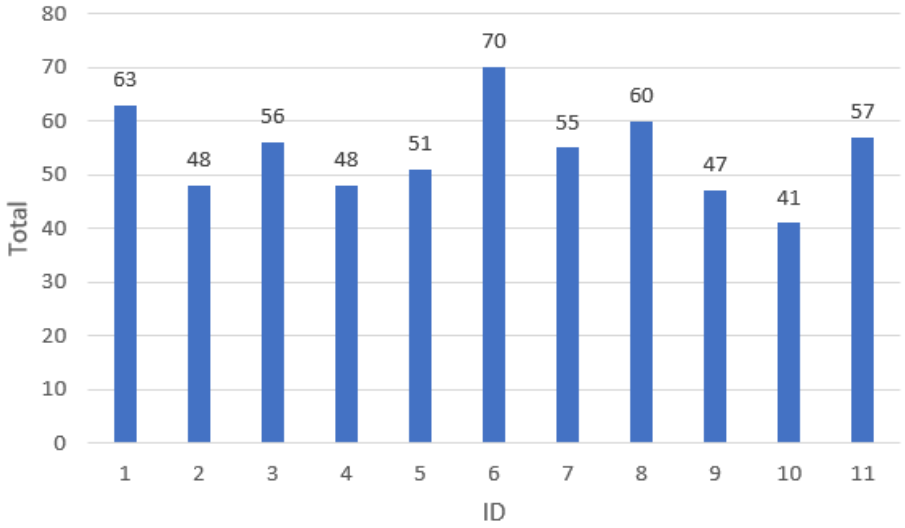
The System Usability Scale (SUS) is a reliable tool and is widely used to evaluate the usability of a system. The participants' results were analyzed by completing the System Usability Scale, and the outcomes are presented below.



Graph 5 – SUS score for each participant

The obtained scores were highly positive, with most participants achieving a rating above 68, corresponding to the "Good" classification. Only two participants scored below this value, obtaining a score of 60. The average score in this questionnaire was 79.2, favoring the success of the project in terms of usability, since this score falls within the "Good" range, very close to "Excellent," which starts at 80.3.

Through this questionnaire, it was concluded that the system's usability is well developed and, according to the participants' opinion, practically ready to be used in real rehabilitation situations. In game development, the degree of similarity to reality is a critical factor that needs consideration. To evaluate the game's truthfulness or similarities with reality, the Virtual Environment Verisimilitude Questionnaire was employed.

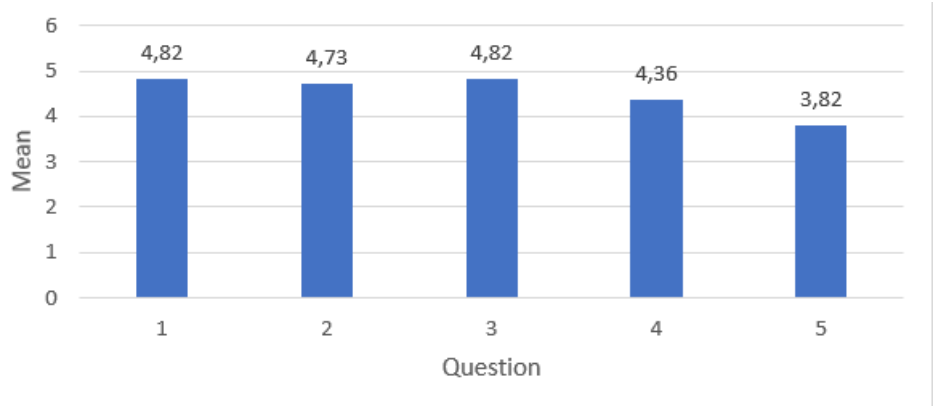


Graph 6 - Virtual Environment Verisimilitude Questionnaire scores for each participant

Above, in Graph 6, the scores of all participants in the test sessions are presented. The highest score achieved was precisely 70, with only one participant attaining this value, while the lowest score was 41.

Regarding the average score, it was calculated to be 54.18. This value indicates that, in general, the system simulates reality to some extent. In other words, the tasks are similar to real-life situations, and the environment resembles a supermarket in this specific case.

To collect additional feedback on various aspects related to the developed game, we utilized the Customized Questionnaire. This questionnaire includes five multiple-choice questions and five open-ended questions. The goal of the questionnaire is to gather diverse feedback to help us further improve the game.



Graph 7 – Score's mean for each question of the Customized Questionnaire

These firsts 5 questions were:

1. How appropriate do you consider the fact that the game is in a supermarket environment?
2. How helpful do you consider the use of the CAVE system?
3. How helpful do you consider the requirement of natural user interface to play?
4. How operational do you think the game is?
5. How ready do you think the game is to be played without user errors?

With the answers presented in these first 5 multiple-choice questions, it was possible to get the mean of each one of them, which is represented in Graph 7. Through the answers, it is possible to assess that the fact that the game is represented in a supermarket environment and is developed at CAVE can be advantageous and helpful for post-stroke rehabilitation. Most participants also think that the game is operational and that it is almost ready to be played without errors by users.

All the answers that the 11 participants gave in each question, from 0 to 5, are shown in Appendix M.

Regarding the open answer questions, these were the following ones:

6. How do you compare this game to the real-life activity of going to a supermarket?
7. What are the positive aspects of using gamification elements (eg. correct and incorrect feedback, different levels of difficulty) in this activity?
8. What are the negative aspects of using gamification (eg. correct and incorrect feedback, different levels of difficulty) in this activity?
9. How did the different levels of difficulty affect you?
10. Any comment about this experience?

As a result, users indicated that the game was quite similar to reality, as the moves required in the game were similar to as those made in real life.

When asked about the positive aspects of using gamification in this project, many respondents highlighted the dynamic nature of the game and its ability to transform mundane tasks such as shopping into enjoyable experiences. Participants also appreciated the accuracy of the feedback, which helped them understand their performance and identify areas for improvement.

However, when asked about the negative aspects of gamification and feedback, some participants mentioned that elderly people might find it challenging to recognize the gamification elements or take the activity seriously.

Another important topic for gathering feedback was the increasing level of difficulty throughout the game. Participants reported that they felt the difficulty of the levels increased progressively, as their arm and leg muscles became increasingly sore, requiring greater physical and mental effort to complete the levels.

Finally, an optional open-ended question allowed participants to leave additional comments about the project. The responses were unanimous, emphasizing that the game was enjoyable to play and that the project had potential for future development.

5. Discussion

The results of the tests and feedback provided valuable information about the success of the stroke rehabilitation project developed. However, it is important to note that these results need to be analyzed and interpreted within the context of the project's limitations. This chapter will reflect on the development process, discuss the limitations encountered, and offer recommendations for future work.

Based on the questionnaires completed by participants at the end of each user test session, the developed game appears to meet the necessary principles to be considered successful for post-stroke rehabilitation. Prior to the development of RehabMarket, research was conducted to understand the existing technologies for post-stroke rehabilitation. In the section "5.4 Additions/Innovation/Contribution," the limitations of these technologies are analyzed, and the factors that set RehabMarket apart are discussed. For instance, participants in other systems did not always feel comfortable with the game environment, but in this case, the supermarket environment was advantageous.

One of the important limitations that the project aimed to address was the fact that most existing systems only stimulate the upper limbs. In RehabMarket, both the upper and lower limbs are stimulated, and the feedback from participants during user tests was positive. However, as the levels increased, the difficulty also increased, leading to muscle soreness, which was perceived as a positive sign.

While the project successfully addressed gaps in existing technologies, it also encountered limitations. For example, since CAVE is a fixed system, the project could only be tested in a single physical space. This made the development process more time-consuming and complicated, as each functionality had to be tested in the ARDITI laboratories where CAVE is installed. This limitation also had an impact on the user tests, as participants had to travel to ARDITI to test the game.

Another limitation was that the user tests did not include stroke survivors. None of the participants had physical or cognitive disabilities resulting from stroke, due to the short time frame to find appropriate participants and the logistical difficulties they may have faced in traveling to ARDITI. Therefore, the tests were performed with people from the NeuroRehabLab group and classmates.

In conclusion, the RehabMarket project successfully addressed gaps in existing post-stroke rehabilitation technologies. However, limitations such as the fixed physical space and the lack of participation from stroke survivors in user tests should be considered in future work. Overall, the project shows potential for further development and implementation.

In practical terms in the development of RehabMarket, it is important to note that it was necessary to learn all about Unity software because it was the first time I had contact with it. As I didn't have the basis for developing a project in Unity, it became a bit difficult to start using it, but it turned out to be an aspect that was improving over time, as was daily research and practice work.

As with all projects, some aspects need to be improved and created for the future. In this case, the main objective that should be met in future work is in line with what was previously mentioned, which would be to do user tests with people who have had a stroke and who need rehabilitation. Only then the results would be as correct as possible and would eventually prove RehabMarket's ability to assist these same people.

In the future, it will be important to enhance the flexibility of the RehabMarket system, particularly in terms of its playability. As previously mentioned, currently the only space for testing RehabMarket is in ARDITI laboratories. However, to increase accessibility, the system should be made playable in a more portable environment, allowing for a wider range of experiences.

Furthermore, in future iterations of the game, certain aspects could be improved. For instance, participants should be given the option to enter their name, and the time taken to complete each level

should be automatically saved and displayed at the end of the game. This would enable individuals to compare their results with future tests.

Additionally, it would be beneficial to introduce a feature that allows participants or their helpers to customize the number of shelves and products for each level. One approach could be to add a "Customize Levels" button to the main menu, enabling users to make desired changes before the game begins. Furthermore, these variables could also be modified within the game environment, with a button on each level allowing users to change any variable at any time.

6. Conclusion

The present thesis aimed to develop and evaluate a technological solution, RehabMarket, for post-stroke rehabilitation. This work presents an alternative approach to existing systems, with unique features that address specific challenges.

The development process was carried out systematically, beginning with extensive research on the topic. Subsequently, an initial concept for the game was formulated and presented in a focus group session. The feedback gathered from this session was analyzed and incorporated into the development of RehabMarket. After the implementation phase, user tests were conducted to assess the effectiveness of the system.

The results from the user tests confirm the success of the development process, fulfilling the project's objectives. This includes the initial research, conceptualization, and implementation of RehabMarket, as well as its validation through user testing. Overall, the work presented in this thesis provides a valuable contribution to the field of post-stroke rehabilitation technology.

7. References

[1] M. Katan e A. Luft, «Global Burden of Stroke», Semin. Neurol., vol. 38, n. 02, pp. 208–211, Abr. 2018, doi: 10.1055/s-0038-1649503.

[2] A. K. Boehme, C. Esenwa, e M. S. V. Elkind, «Stroke Risk Factors, Genetics, and Prevention», Circ. Res., vol. 120, n. 3, pp. 472–495, Fev. 2017, doi: 10.1161/CIRCRESAHA.116.308398.

[3] E. S. Donkor, «Stroke in the 2 1 s t Century: A Snapshot of the Burden, Epidemiology, and Quality of Life», Stroke Res. Treat., vol. 2018, pp. 1–10, Nov. 2018, doi: 10.1155/2018/3238165.

[4] «WHO EMRO | Stroke, Cerebrovascular accident | Health topics». http://www.emro.who.int/health-topics/stroke-cerebrovascular-accident/index.html (acessed october 13, 2021).

[5] J. Das e R. G.K., «Post stroke depression: The sequelae of cerebral stroke», Neuroscience & Biobehavioral Reviews, vol. 90, pp. 104–114, Jul. 2018, doi: 10.1016/j.neubiorev.2018.04.005.

[6] O. Sinanović, Z. Mrkonjić, S. Zukić, M. Vidović, e K. Imamović, «Post-Stroke Language Disorders», Acta Clin. Croat., vol. 50, n. 1, pp. 79–93, Mar. 2011.

[7] L. Hepworth et al., «Post-stroke Visual Impairment: A Systematic Literature Review of Types and Recovery of Visual Conditions», OR, vol. 5, n. 1, pp. 1–43, Jan. 2016, doi: 10.9734/OR/2016/21767.

[8] S. Park e J.-Y. Park, «Grip strength in post-stroke hemiplegia», J. Phys. Ther. Sci., vol. 28, n. 2, pp. 677–679, 2016, doi: 10.1589/jpts.28.677

[9] K. Eng et al., «Interactive visuo-motor therapy system for stroke rehabilitation», Med. Biol. Eng. Comput., vol. 45, n. 9, pp. 901–907, Set. 2007, doi: 10.1007/s11517-007-0239-1

[10] D. Webster e O. Celik, «Systematic review of Kinect applications in elderly care and stroke rehabilitation», J. NeuroEngineering Rehabil., vol. 11, n. 1, p. 108, 2014, doi: 10.1186/1743-0003-11-108

[11] K. J. Bower, J. Louie, Y. Landesrocha, P. Seedy, A. Gorelik, e J. Bernhardt, «Clinical feasibility of interactive motion-controlled games for stroke rehabilitation», J. NeuroEngineering Rehabil., vol. 12, n. 1, p. 63, Dez. 2015, doi: 10.1186/s12984-015-0057-x.

[12] V.-M. Nurkkala, J. Kalermo, e T. Jarvilehto, «Development of Exergaming Simulator for Gym Training, Exercise Testing and Rehabilitation», p. 10.

[13] S. N. Gieser, E. Becker, e F. Makedon, «Using KAVE in physical rehabilitation exercises for rheumatoid arthritis», em Proceedings of the 6th International Conference on PErvasive Technologies Related to Assistive Environments - PETRA '13, Rhodes, Greece, 2013, pp. 1–4. doi: 10.1145/2504335.2504367.

[14] P. J. Sparto, J. M. Furman, S. L. Whitney, L. F. Hodges, e M. S. Redfern, «Vestibular rehabilitation using a wide field of view virtual environment», em The 26th Annual International Conference of the IEEE Engineering in Medicine and Biology Society, San Francisco, CA, USA, 2004, vol. 4, pp. 4836–4839. doi: 10.1109/IEMBS.2004.1404338.

[15] Y. Chen, K. T. Abel, J. T. Janecek, Y. Chen, K. Zheng, e S. C. Cramer, «Home-based technologies for stroke rehabilitation: A systematic review», International Journal of Medical Informatics, vol. 123, pp. 11–22, Mar. 2019, doi: 10.1016/j.ijmedinf.2018.12.001.

[16] E. Pedroli, S. Serino, P. Cipresso, F. Pallavicini, e G. Riva, «Assessment and rehabilitation of neglect using virtual reality: a systematic review», Front. Behav. Neurosci., vol. 9, p. 226, 2015, doi: 10.3389/fnbeh.2015.00226.

[17] Afonso Gonçalves, Sergi Bermúdez i Badia. (2018). KAVE: Building Kinect Based CAVE Automatic Virtual Environments, Methods for Surround- Screen Projection Management, Motion Parallax and Full-Body Interaction Support. PACM on Human-Computer Interaction, 2, EICS. https://doi.org/10.1145/3229092

[18] J. R. Lewis and J. Sauro, 'Item benchmarks for the system usability scale', J. Usability Stud., vol. 13, no. 3, pp. 158–167, May 2018.

[19] C. Dantas, A. L. Jegundo, J. Quintas, A. I. Martins, A. Queirós, and N. P. Rocha, 'European Portuguese Validation of Usefulness, Satisfaction and Ease of Use Questionnaire (USE)', in Recent Advances in Information Systems and Technologies, 2017, pp. 561–570.

[20] «Explore NeuroRehabLab - UMa in 3D», Matterport. https://my.matterport.com/show/?m=kZHMXWXesoE (acessed december 13, 2021).

[21] «IJSRM4-9.pdf». Acessed: June, 20, 2022. Available in: https://www.circleinternational.co.uk/wp-content/uploads/2021/01/IJSRM4-9.pdf#page=46

[22] M. Ali Ramdhani, D. Sa'adillah Maylawati, A. Syakur Amin, e H. Aulawi, «Requirements Elicitation in Software Engineering», Int. J. Eng. Technol., vol. 7, n. 2.29, p. 772, mai. 2018, doi: 10.14419/ijet.v7i2.29.14254

[23] «doi:10.1016/j.artint.2005.05.003 | Elsevier Enhanced Reader». https://reader.elsevier.com/reader/sd/pii/S0004370205000792?token=0DBA8F64DB4D13B578F04130AB3FA6E780A15CCBC8ADD5AC4EFCC8E0A1DB4B71217BF22149F2C9CDD398A5A902B8E9FB&originRegion=eu-west-1&originCreation=20221010121407 (accessed october 10, 2022).

[24] «What is a Flow Diagram & Flowchart? (7 Types + Definitions)», Slickplan. https://slickplan.com/diagram/what-is-a-flow-diagram (accessed october 10, 2022).

[25] hickeys, «Kinect for Windows - Windows apps». https://learn.microsoft.com/en-us/windows/apps/design/devices/kinect-for-windows (accessed october 21, 2022).

[26] «Unity Asset Store - The Best Assets for Game Making». https://assetstore.unity.com/ (accessed november 7, 2022).

[27] «Supermarket Interior | 3D Urban | Unity Asset Store». https://assetstore.unity.com/packages/3d/environments/urban/supermarket-interior-38178 (accessed november 7, 2022).

[28] J.-A. Gil-Gómez, P. Manzano-Hernández, S. Albiol-Pérez, C. Aula-Valero, H. Gil-Gómez, e J.-A. Lozano-Quilis, «USEQ: A Short Questionnaire for Satisfaction Evaluation of Virtual Rehabilitation Systems», *Sensors (Basel)*, vol. 17, n.º 7, p. 1589, jul. 2017, doi: 10.3390/s17071589.

[29] «PRESENCE QUESTIONNAIRE (Witmer & Singer, Vs. 3.0, Nov. 1994)* Revised by the UQO Cyberpsychology Lab (2004) - PDF Free Download». https://docplayer.net/52991659-Presence-questionnaire-witmer-singer-vs-3-0-nov-1994-revised-by-the-uqo-cyberpsychology-lab-2004.html (accessed december 1, 2022).

[30] J. Brooke, «SUS: A quick and dirty usability scale», *Usability Eval. Ind.*, vol. 189, nov. 1995.

This page was intentionally left blank

8. Appendix

Appendix A. Informed consent focus group – Portuguese version

Investigador Principal:

[Guilherme Spínola Freitas, Universidade da Madeira, 2034917@student.uma.pt]

Objetivo do Estudo

No âmbito da cadeira de mestrado Preparação de Projeto do Mestrado em Engenharia Informática, realizou-se uma atividade para reabilitação pós-AVC, utilizando o sistema CAVE, onde o objetivo será estimular os movimentos dos membros superiores, inferiores e estimular a capacidade cognitiva. Este estudo tem o objetivo de realizar a atividade desenvolvida, como sendo testes de utilizador.

Procedimento

Nesta atividade, serão apresentados 10 níveis de jogo, onde o objetivo para a conclusão de cada um deles é diferente entre si e com diferentes graus de dificuldade, sendo que será explicado antes de começar cada nível. Após a realização da atividade, serão fornecidos questionários que terão como objetivo perceber o grau de satisfação e para caracterizar o nível da experiência.

Nesta sessão de focus group, terá a possibilidade de ver detalhadamente mais informações sobre o projeto e visualizar todo o trabalho que foi realizado até agora. Ao longo desta sessão vamos discutir vários temas e vamos pedir a sua opinião e feedback acerca de ideias novas que possam surgir e que já existem, com o objetivo de melhorar o trabalho.

Critérios de Inclusão

Será considerado elegível para participar neste estudo qualquer pessoa que se mostre interessada em discutir opiniões sobre reabilitação pós-AVC.

Riscos

A sua participação não tem quaisquer riscos.

Benefícios

Este estudo não terá um benefício imediato para si, mas a informação recolhida permitirá contribuir para o desenvolvimento de tecnologias para apoio à reabilitação.

Compensações & Custos

Não haverá compensação. Participar neste estudo não implica quaisquer custos, será apenas necessária uma deslocação à Universidade da Madeira.

Confidencialidade

A confidencialidade dos dados será mantida das seguintes formas: serão recolhidos dados identificáveis. Os dados recolhidos serão utilizados apenas para fins de investigação.

Direitos

A sua participação é voluntária. Você é livre de interromper a sua participação em qualquer momento. A recusa em participar ou interrupção da participação não resultará em qualquer penalização, ou perda de eventuais benefícios ou direitos. O investigador principal poderá decidir, de forma fundamentada, interromper a sua participação neste estudo. Caso se verifique esta situação, esta não resultará em qualquer penalização, ou perda de eventuais benefícios ou direitos.

Esclarecimento de Dúvidas e Contactos

Se você tem dúvidas sobre este estudo em qualquer momento, poderá contactar o investigador responsável através do contacto acima referido. Também poderá em qualquer momento da sessão do *focus group* colocar questões ao investigador.

Consentimento Informado Voluntário

Ao concordar participar neste estudo, você confirma que leu a informação acima descrita sobre o mesmo, que todas as suas dúvidas foram esclarecidas e que aceita a gravação em áudio da sessão e a recolha de fotografias, mantendo-se o anonimato nas mesmas:

☐ Aceito a gravação da sessão

☐ Aceito a recolha de fotografias

_____ _____

ASSINATURA DO PARTICIPANTE DATA

Investigador que Obtém o Consentimento

Como membro da equipa de investigação, confirmo que expliquei ao participante acima referido a natureza e finalidade deste estudo de investigação, e que esclareci quais os potenciais benefícios e eventuais riscos da participação no estudo. Todas as perguntas foram respondidas e estou disponível para esclarecer quaisquer dúvidas que possam surgir ao longo do estudo.

_____ _____

ASSINATURA DO INVESTIGADOR DATA

Appendix B. Focus group final activity table

Participants' names: _____

Variable Name	Level 1	Level 2	Level 3

Appendix C. Focus Group Plan – Portuguese version

Primeira parte

- Ler e assinar consentimento informado

- PowerPoint

-> Primeira parte da apresentação (até slide 8 do ambiente) – **7 min**

-> Pergunta aberta (Que atividades podemos ter num supermercado?) – **7 min**

-> Trabalho já feito (slide 10 ao 15) – **5 min**

-> Perguntas e respostas (4 perguntas do slide 16) – **40 min**

-> Vantagens e desvantagens de ser um supermercado e de ser este tipo de jogo?

-> Como pode haver interação com o jogo num contexto de reabilitação?

-> Como é que o jogo pode dar feedback ao participante?

-> Que tipo de produtos são prioritários?

Segunda parte

- Discutir a criação de variáveis que possam complementar o jogo, de maneira a torná-lo mais complexo. – **10 min**

- tempo
- energia
- progresso
- prateleiras
- numero de produtos
- objeto aleatório no ecrã (distratores ou prémios)
- objeto lixo
- distancia entre objeto e destino

- Fazer atividade em que devem associar as variáveis a cada nível do jogo e explicar o porquê e como iria ser integrada essa variável no nível. (levar material pronto para a atividade) – **20 min**

- Agradecer e terminar.

Appendix D. User tests Plan

Study plan for user tests

1. Read and sign the informed consent;
2. Brief description of the context in which the activity will be held;
    a. Esta atividade que vai ser realizada será um jogo, em que o objetivo principal é reabilitar pessoas que tenham sofrido um AVC. Aqui será possível estimular membros superiores e inferiores, e até estimular as capacidades cognitivas da pessoa.
3. Brief description of the game;
    a. Este jogo tem 10 níveis diferentes, sendo que a dificuldade do mesmo aumenta com o avançar dos níveis. A atividade tem como ambiente um supermercado, onde o objetivo principal do jogo é interagir com diferentes produtos que estão, habitualmente, presentes num supermercado, dependendo de nível para nível.
    b. Neste jogo existem 4 modos de interação: arrastar produto para uma posição, arrastar produto para o local correto, selecionar produtos ao aproximar a mão do mesmo e recolha de produtos tendo em conta um orçamento.
    c. Os primeiros níveis vão servir para o participante se habituar ao ambiente que foi criado e para perceber como serão as interações em níveis futuros.
    d. Os 2 primeiros níveis serão apenas para arrastar um produto para outra posição, na mesma prateleira e em prateleiras diferentes;
    e. Os níveis 3 e 4 são para arrastar os produtos para os cestos de compras e as caixas das prateleiras para as caixas das mesas;
    f. O nível 5 mistura produtos e caixas;
    g. O nível 6 terá como objetivo aproximar as mãos, sem arrastar, aos produtos que estão presentes nas listas de compras para os identificar;
    h. O nível 7 tem como objetivo identificar os produtos que estão presentes nas listas de compras e arrastá-los para os cestos de compras, sendo que o nível 8 é igual, no entanto, a lista de compras desaparecerá após 10 segundos;
    i. O nível 9 obriga a que a pessoa tenha em conta o preço de cada produto e o arraste para o cesto de compras, de maneira a atingir o valor exato do orçamento total que é apresentado, sendo que o nível 10 é igual, mas o orçamento desaparecerá após 10 segundos.
    j. Cada nível terá as suas instruções, que estarão presentes no monitor frontal.
    k. Para interagir com a parede do lado direito, deve-se utilizar o braço direito, e vice-versa.
    l. Eu vou estar presente para ajudar em certos movimentos que não estejam corretos, ou para auxiliar em qualquer dúvida durante a realização da atividade.
4. User plays the 10 levels of the game (~20 minutes);
5. Distribution and completion of questionnaires (SUS, USEQ, Presence Questionnaire, Customized Questionnaire);
6. Distribution of the compensation.

Appendix E. Informed Consent User Tests – Portuguese version

Consentimento Informado para Testes de Utilizador

Investigador Principal:

[Guilherme Spínola Freitas, Universidade da Madeira, 2034917@student.uma.pt]

Objetivo do Estudo

No âmbito da cadeira de mestrado Preparação de Projeto do Mestrado em Engenharia Informática, desenvolveu-se uma atividade para reabilitação pós-AVC, utilizando o sistema CAVE, onde o objetivo será estimular o movimento do membro superior, inferior e estimular a capacidade cognitiva. Este estudo tem o objetivo de realizar a atividade desenvolvida e estudar a usabilidade do sistema.

Procedimento

Nesta atividade, serão apresentados 10 níveis de jogo, onde o objetivo para a conclusão de cada um deles é diferente entre si e com diferentes graus de dificuldade, sendo que será explicado antes de começar cada nível. Após a realização da atividade, serão fornecidos questionários que terão como objetivo perceber o grau de satisfação e caracterizar o nível da experiência.

Critérios de Inclusão

Será considerado elegível para participar neste estudo qualquer pessoa, com mais de 18 anos, que se mostre interessada em realizar a atividade.

Riscos

A sua participação não tem quaisquer riscos.

Benefícios

Este estudo não terá um benefício imediato para si, mas a informação recolhida permitirá contribuir para o desenvolvimento de tecnologias para apoio à reabilitação.

Custos

Participar neste estudo não implica quaisquer custos, será apenas necessária uma deslocação à Universidade da Madeira.

Confidencialidade

A confidencialidade dos dados será mantida das seguintes formas: serão recolhidos dados identificáveis. Os dados recolhidos serão utilizados apenas para fins de investigação.

Direitos

A participação é voluntária. É livre de interromper a participação em qualquer momento. A recusa em participar ou interrupção da participação não resultará em qualquer penalização, ou perda de eventuais benefícios ou direitos. O investigador principal poderá decidir, de forma fundamentada, interromper a sua participação neste estudo. Caso se verifique esta situação, esta não resultará em qualquer penalização, ou perda de eventuais benefícios ou direitos.

Esclarecimento de Dúvidas e Contactos

Se tiver dúvidas sobre este estudo em qualquer momento, pode contactar o investigador responsável através do contacto acima referido. Também pode em qualquer momento da realização da atividade, colocar questões ao investigador.

Consentimento Informado Voluntário

Ao concordar participar neste estudo, confirmo que li a informação acima descrita sobre o mesmo, que todas as suas dúvidas foram esclarecidas e que aceito a recolha de fotografias, mantendo-se o anonimato nas mesmas:

☐ Aceito a recolha de fotografias/vídeos

_____ _____

ASSINATURA DO PARTICIPANTE DATA

Investigador que Obtém o Consentimento

Como membro da equipa de investigação, confirmo que expliquei ao participante acima referido a natureza e finalidade deste estudo de investigação, e que esclareci quais os potenciais benefícios e eventuais riscos da participação no estudo. Todas as perguntas foram respondidas e estou disponível para esclarecer quaisquer dúvidas que possam surgir ao longo do estudo.

_____ _____

ASSINATURA DO INVESTIGADOR DATA

Appendix F. Presence Questionnaire

PRESENCE QUESTIONNAIRE

(Witmer & Singer, Vs. 3.0, Nov. 1994)*

Revised by the UQO Cyberpsychology Lab (2004)

Characterize your experience in the environment, by marking an "X" in the appropriate box of the 7-point scale, in accordance with the question content and descriptive labels. Please consider the entire scale when making your responses, as the intermediate levels may apply. Answer the questions independently in the order that they appear. Do not skip questions or return to a previous question to change your answer.

WITH REGARD TO THE EXPERIENCED ENVIRONMENT

1. How much were you able to control events?

|_____|_____|_____|_____|_____|_____|_____| NOT

AT ALL SOMEWHAT COMPLETELY

2. How responsive was the environment to actions that you initiated (or performed)?

|_____|_____|_____|_____|_____|_____|_____| NOT
MODERATELY COMPLETELY RESPONSIVE RESPONSIVE
RESPONSIVE

3. How natural did your interactions with the environment seem?

|_____|_____|_____|_____|_____|_____|_____|
EXTREMELY BORDERLINE COMPLETELY ARTIFICIAL NATURAL

4. How much did the visual aspects of the environment involve you?

|_____|_____|_____|_____|_____|_____|_____| NOT
AT ALL SOMEWHAT COMPLETELY

5. How natural was the mechanism which controlled movement through the environment?

|_____|_____|_____|_____|_____|_____|_____|
EXTREMELY BORDERLINE COMPLETELY ARTIFICIAL NATURAL

6. How compelling was your sense of objects moving through space?

|_____|_____|_____|_____|_____|_____|_____| NOT
AT ALL MODERATELY VERY COMPELLING COMPELLING

7. How much did your experiences in the virtual environment seem consistent with your real world experiences?

|_____|_____|_____|_____|_____|_____|_____| NOT
MODERATELY VERY CONSISTENT CONSISTENT CONSISTENT

8. Were you able to anticipate what would happen next in response to the actions that you performed?

|_____|_____|_____|_____|_____|_____|_____| NOT
AT ALL SOMEWHAT COMPLETELY

9. How completely were you able to actively survey or search the environment using vision?

|_____|_____|_____|_____|_____|_____|_____| NOT
AT ALL SOMEWHAT COMPLETELY

10. How compelling was your sense of moving around inside the virtual environment?

|_____|_____|_____|_____|_____|_____|_____| NOT
MODERATELY VERY COMPELLING COMPELLING COMPELLING

11. How closely were you able to examine objects?

|_____|_____|_____|_____|_____|_____|_____| NOT
AT ALL PRETTY VERY CLOSELY CLOSELY

12. How well could you examine objects from multiple viewpoints?

|_____|_____|_____|_____|_____|_____|_____| NOT
AT ALL SOMEWHAT EXTENSIVELY

13. How involved were you in the virtual environment experience?

|_____|_____|_____|_____|_____|_____|_____| NOT
MILDLY COMPLETELY INVOLVED INVOLVED ENGROSSED

14. How much delay did you experience between your actions and expected outcomes?

|_____|_____|_____|_____|_____|_____|_____| NO
DELAYS MODERATE LONG DELAYS DELAYS

15. How quickly did you adjust to the virtual environment experience?

|_____|_____|_____|_____|_____|_____|_____| NOT
AT ALL SLOWLY LESS THAN

ONE MINUTE

16. How proficient in moving and interacting with the virtual environment did you feel at the end of the experience?

|_____|_____|_____|_____|_____|_____|_____| NOT
REASONABLY VERY PROFICIENT PROFICIENT PROFICIENT

17. How much did the visual display quality interfere or distract you from performing assigned tasks or required activities?

|_____|_____|_____|_____|_____|_____|_____| NOT AT ALL
INTERFERED PREVENTED SOMEWHAT TASK PERFORMANCE

18. How much did the control devices interfere with the performance of assigned tasks or with other activities?

|_____|_____|_____|_____|_____|_____|_____| NOT AT ALL INTERFERED INTERFERED SOMEWHAT GREATLY

19. How well could you concentrate on the assigned tasks or required activities rather than on the mechanisms used to perform those tasks or activities?

|_____|_____|_____|_____|_____|_____|_____| NOT AT ALL SOMEWHAT COMPLETELY

IF THE VIRTUAL ENVIRONMENT INCLUDED SOUNDS:

20. How much did the auditory aspects of the environment involve you?

|_____|_____|_____|_____|_____|_____|_____| NOT AT ALL SOMEWHAT COMPLETELY

21. How well could you identify sounds?

|_____|_____|_____|_____|_____|_____|_____| NOT AT ALL SOMEWHAT COMPLETELY

22. How well could you localize sounds?

|_____|_____|_____|_____|_____|_____|_____| NOT AT ALL SOMEWHAT COMPLETELY

IF THE VIRTUAL ENVIRONMENT INCLUDED HAPTIC (SENSE OF TOUCH): 23. How well could you actively survey or search the virtual environment using touch?

|_____|_____|_____|_____|_____|_____|_____| NOT AT ALL SOMEWHAT COMPLETELY

24. How well could you move or manipulate objects in the virtual environment?

|_____|_____|_____|_____|_____|_____|_____| NOT AT ALL SOMEWHAT EXTENSIVELY

Validation of the French-Canadian version developed by the UQO Cyberpsychology Lab:

- 101 participants completed the questionnaire following an immersion in a virtual environment;
- Cronbach's Alpha = .84
- Now 19 items (for VEs without sound/touch) et 24 items (for VEs with sounds/touch)

Scoring :

Total : Items 1 to 19 (reverse items 14, 17, 18)

· « Realism » : Items 3 + 4 + 5 + 6 + 7 + 10 + 13

· « Possibility to act » : Items 1 + 2 + 8 + 9

· « Quality of interface » : Items (all reversed) 14 + 17 + 18

· « Possibility to examine » : Items 11 + 12 + 19

· « Self-evaluation of performance » : Items 15 + 16

· « Sounds* » : Items 20 + 21 + 22

· « Haptic* » : Items 23 + 24

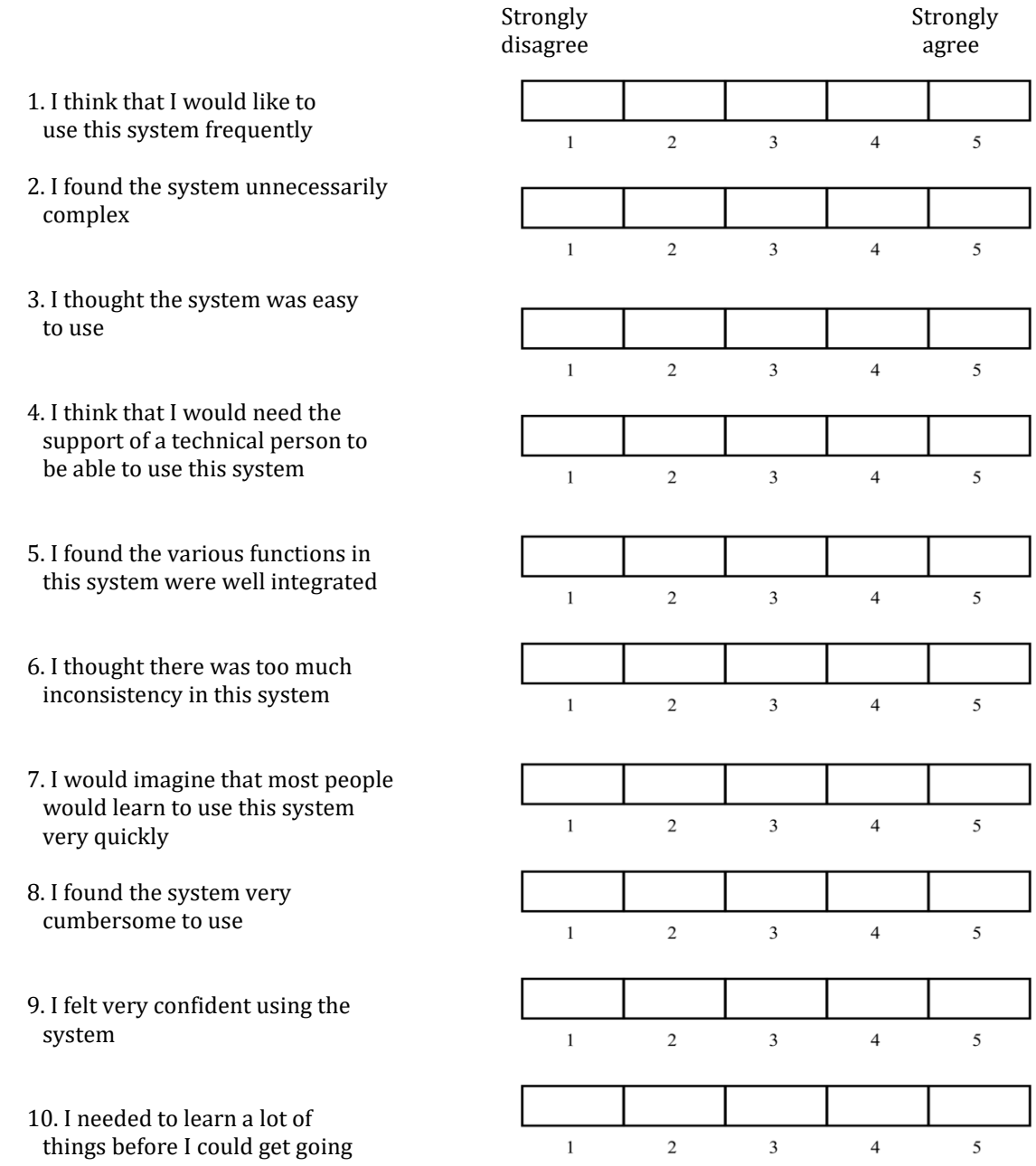
* NOTE : Scoring of « *sounds* » and « *haptic* » are not part of the factor analysis of the French version.

Norms (French version) :

	Moyenne	**Écart type**
Total	104.39	18.99
« Realism »	29.45	12.04
« Possibility to act »	20.76	6.01
« Quality of interface »	15.37	5.15
« Possibility to examine»	15.38	4.90
« Auto-évaluation de la performance »	11.00	2.87

Appendix G. System Usability Scale – SUS

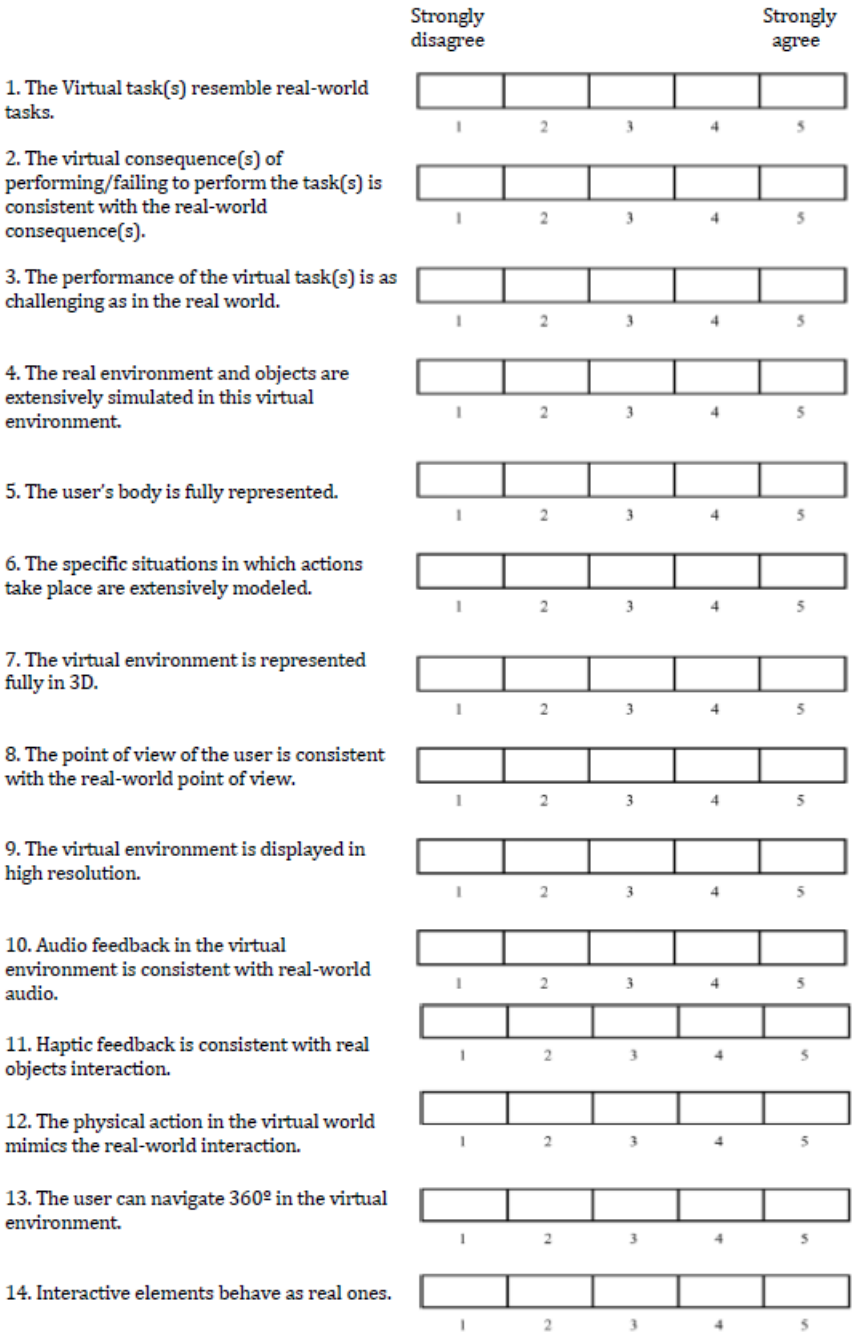
System Usability Scale



Appendix H. USEQ

Question	Response
	Not at All–Very Much
Q1. Did you enjoy your experience with the system?	1 2 3 4 5
Q2. Were you successful using the system?	1 2 3 4 5
Q3. Were you able to control the system?	1 2 3 4 5
Q4. Is the information provided by the system clear?	1 2 3 4 5
Q5. Did you feel discomfort during your experience with the system?	1 2 3 4 5
Q6. Do you think that this system will be helpful for your rehabilitation?	1 2 3 4 5

Appendix I. Virtual Environment Verisimilitude Questionnaire



Appendix J. User Information

User Information

Gender

☐ Male

☐ Female

Name: _____

Age: _____

Dominant hand

☐ Right

☐ Left

Education

☐ Graduation

☐ Master's Degree

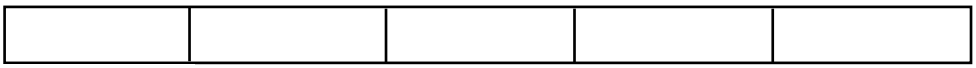
☐ Doctorate

☐ Other

Job: _____

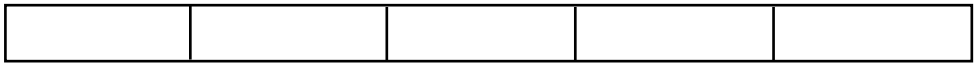
Appendix K. Customized Questionnaire

1. How appropriate do you consider the fact that the game is in a supermarket environment?



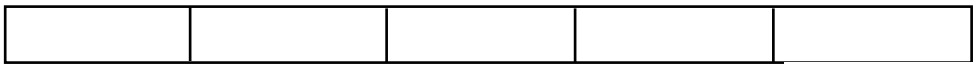
Not appropriate Very appropriate

2. How helpful do you consider the use of the CAVE system?



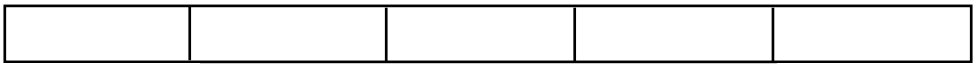
Not helpful Very helpful

3. How helpful do you consider the requirement of natural user interface to play?



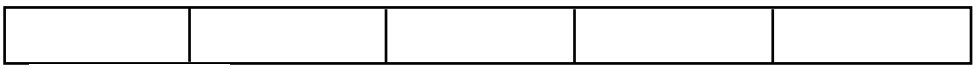
Not helpful Very helpful

4. How operational do you think the game is?



Not operational Very operational

5. How ready do you think the game is to be played without user errors?



Not ready Completely ready

6. How do you compare this game to the real life activity of going to a supermarket?

   Answer: _____

7. What are the positive aspects of using gamification elements (eg. correct and incorrect feedback, different levels of difficulty) in this activity?

   Answer: _____

8. What are the negative aspects of using gamification (eg. correct and incorrect feedback, different levels of difficulty) in this activity?

Answer: _____

9. How did the different levels of difficulty affect you?

   Answer: _____

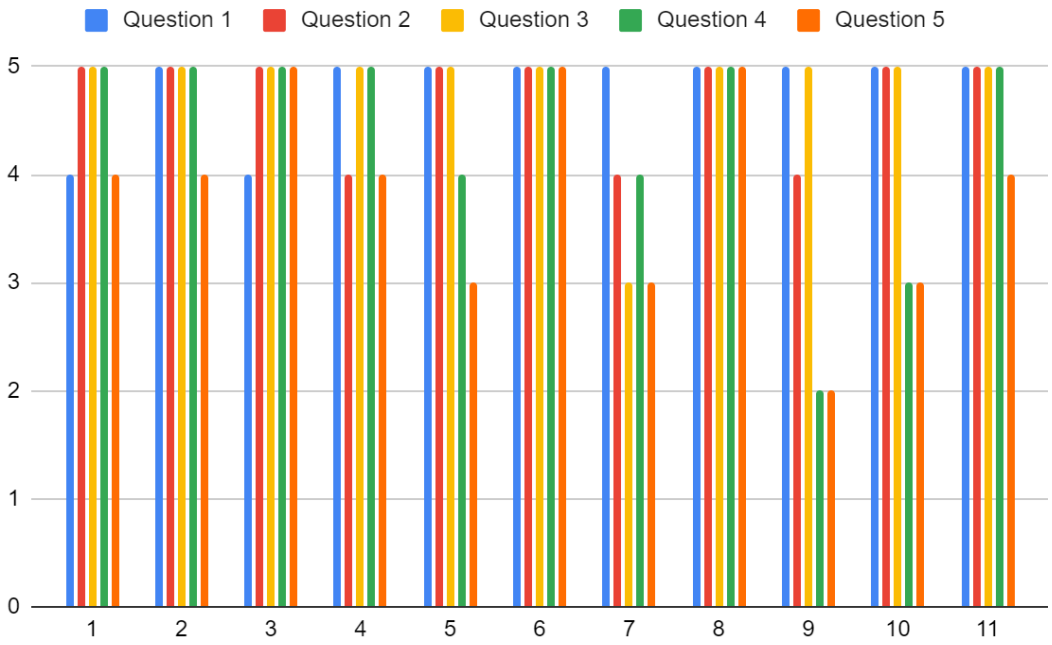
10. Any comment about this experience?

    Answer: _____

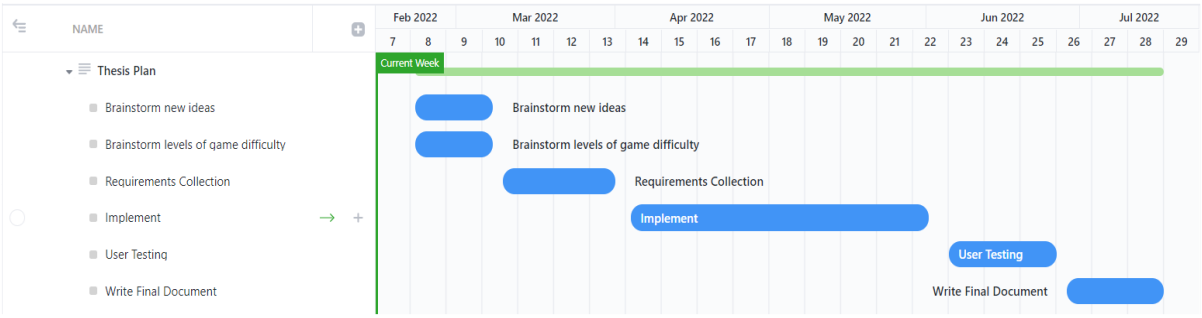
Appendix L. Level times per participant

ID	Level 1	Level 2	Level 3	Level 4	Level 5	Level 6	Level 7	Level 8	Level 9	Level 10	Total
1	00:11.82	00:22.11	00:22.30	00:19.55	04:28.96	01:24.42	01:49.29	01:19.33	00:52.98	02:04.42	13:16.07
2	00:26.23	00:24.05	00:18.65	00:17.68	02:32.62	01:20.99	03:12.21	01:05.44	01:53.65	01:33.66	13:05.23
3	00:31.07	00:19.08	00:19.47	00:20.31	02:04.57	00:56.65	01:31.22	01:01.76	01:31.55	01:09.64	09:45.34
4	00:30.59	00:16.70	00:16.75	00:24.63	03:31.95	01:11.03	02:33.39	01:02.07	02:51.20	00:48.23	13:26.59
5	00:19.27	00:15.23	00:17.11	00:39.40	04:27.10	00:49.27	02:18.20	04:04.54	02:17.56	01:19.88	16:47.61
6	00:24.94	00:14.89	00:24.43	00:31.08	02:45.96	02:44.62	02:08.99	00:45.47	02:28.08	01:27.63	13:56.13
7	00:24.07	00:24.30	00:21.39	00:46.71	04:36.36	01:18.51	02:09.15	01:06.55	02:14.64	01:36.37	14:58.10
8	00:23.92	00:32.49	00:23.10	00:37.29	02:51.36	01:22.77	02:33.26	01:26.30	01:46.12	02:25.74	14:22:40
9	00:28.58	00:14.64	00:43.01	00:20.08	03:41.35	00:54.72	04:52.54	03:00.03	01:26:11	02:10:01	17:51.10
10	00:21.40	00:33.16	00:26.56	00:27.65	03:36.08	01:07.47	02:11.19	01:00.07	02:07.51	03:04.49	14:55.62
11	00:14.87	00:20.86	00:19.79	00:18.80	02:23.02	00:44.97	01:20.24	01:09.50	01:25.93	01:59.63	10:17.62
Average	00:23.41	00:21.52	00:23.14	00:27.53	03:22.05	01:16.20	02:25.39	01:33.04	01:54.27	01:47.35	13:53.07

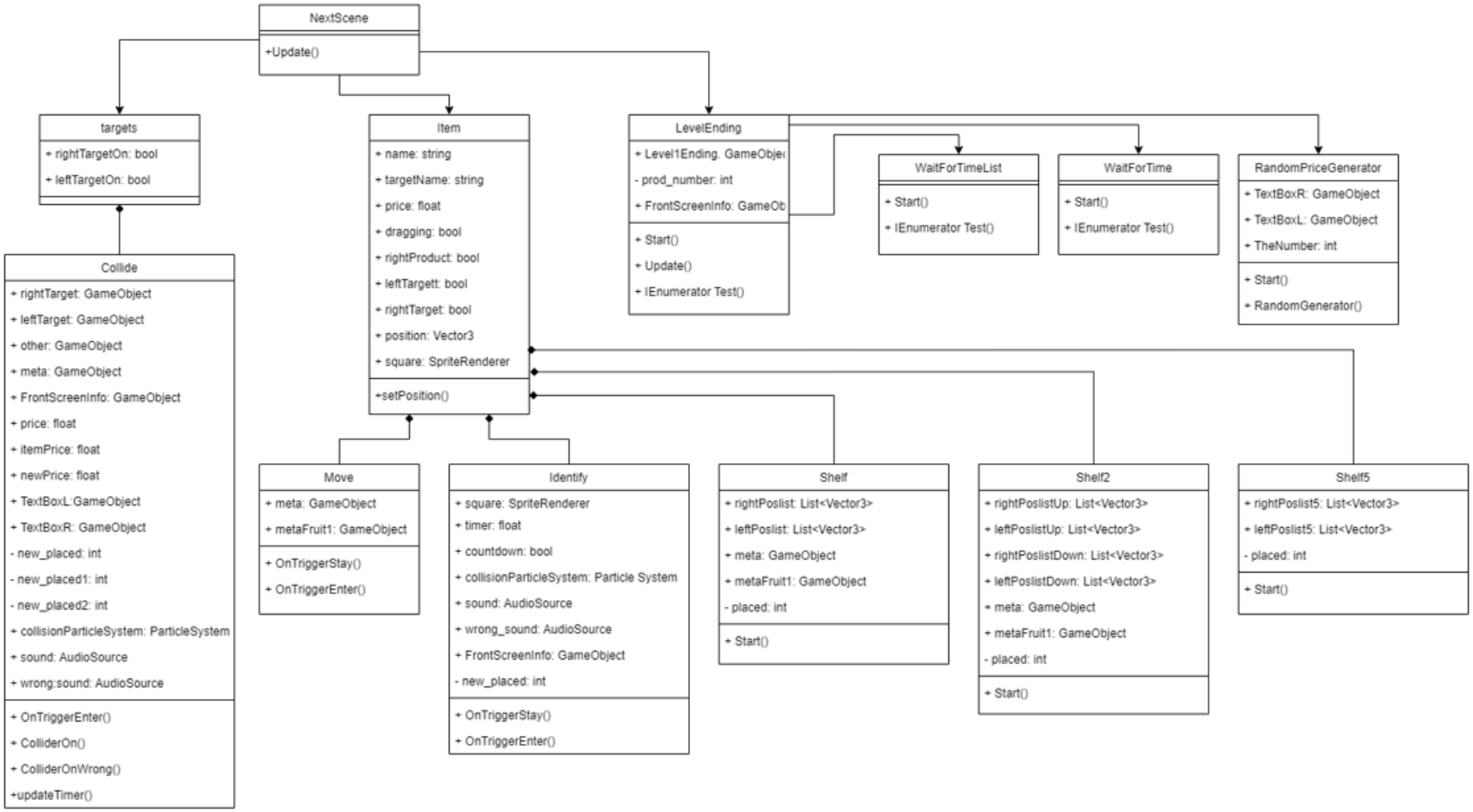
Appendix M. Customized Questionnaire results



Appendix N. Gantt Diagram



Appendix O. Diagram Class Updated



Appendix P. Flow Diagram