TALLINN UNIVERSITY OF TECHNOLOGY
School of Information Technologies

Vladislav Konstantinov 193598IADB

GAMIFICATION OF HAND THERAPY USING LEAP MOTION

Bachelor's Thesis

Supervisor: Yevhen Bondarenko
Early Stage Researcher

Tallinn 2024

TALLINNA TEHNIKAÜLIKOOL
Infotehnoloogia teaduskond

Vladislav Konstantinov 193598IADB

KÄTETERAAPIA MÄNGULISUS LEAP MOTIONI ABIL

Bakalaureusetöö

Juhendaja: Yevhen Bondarenko
Early Stage Researcher

Tallinn 2024

Author's Declaration of Originality

I hereby certify that I am the sole author of this thesis. All the used materials, references to the literature and the work of others have been referred to. This thesis has not been presented for examination anywhere else.

Author: Vladislav Konstantinov

03.01.2024

Abstract

This thesis explores the integration of gamification into hand therapy using Leap Motion technology, aiming to enhance rehabilitation processes for patients with upper limb dysfunctions. It delves into the development and implementation of a comprehensive program that employs innovative motion-tracking technologies to offer a personalized, motivating approach to recovery. The primary objective is to create a unique therapeutic tool that merges scientifically proven rehabilitation methods with elements of game design, thereby making therapy more engaging and effective for patients. Additionally, the thesis addresses the development of adaptive tools for exercise customization, enabling both medical professionals and patients to adjust exercises for optimal complexity and effectiveness. The incorporation of a real-time data collection and analysis system further allows for the assessment and adjustment of the therapy program. Overall, the research aims to expand the capabilities of rehabilitation techniques through gamification, ultimately improving patient engagement, motivation, and faster, more effective recovery of upper limb functions.

The thesis is written in English and is 64 pages long, including 7 chapters, 26 figures and 1 tables.

Annotatsioon

Käteteraapia mängulisus Leap Motioni abil

See töö uurib mängulisuse integreerimist käeteraapiasse, kasutades Leap Motion tehnoloogiat, eesmärgiga parandada ülajäsemete düsfunktsioonidega patsientide rehabilitatsiooniprotsesse. Töö süveneb tervikliku programmi väljatöötamisse ja rakendamisse, mis kasutab uuenduslikke liikumise jälgimise tehnoloogiaid, pakkudes isikupärastatud ja motiveerivat lähenemist taastumisele. Peamine eesmärk on luua ainulaadne terapeutiline vahend, mis ühendab teaduslikult tõestatud rehabilitatsioonimeetodid mängudisaini elementidega, muutes seeläbi teraapia patsientide jaoks kaasahaaravamaks ja tõhusamaks. Lisaks käsitleb töö kohandatavate harjutusvahendite arendamist, võimaldades nii meditsiinitöötajatel kui ka patsientidel harjutusi optimaalse keerukuse ja tõhususe saavutamiseks kohandada. Reaalajas andmete kogumise ja analüüsisüsteemi integreerimine võimaldab veelgi hinnata ja kohandada teraapiaprogrammi. Üldiselt on uuringu eesmärk laiendada rehabilitatsioonitehnikate võimalusi mängulisuse kaudu, parandades lõppkokkuvõttes patsientide kaasatust, motivatsiooni ja kiirendama ülajäsemete funktsioonide tõhusamat taastumist.

Lõputöö on kirjutatud Inglise keeles ning sisaldab teksti 64 leheküljel, 7 peatükki, 26 joonist, 1 tabelit.

List of Abbreviations and Terms

C#	C Sharp, programming language
VR	Virtual Reality
DALY	Disability-Adjusted Life Year
Flexion	A movement that decreases the angle between two body parts. In wrist movements, it refers to bending the wrist so that the palm moves towards the forearm. In the context of finger movements, flexion is the bending of the fingers towards the palm, bringing the bones of the fingers closer to the palm
Extension	The opposite of flexion, where the angle between two body parts increases. For wrist movements, extension involves moving the back of the hand towards the forearm, increasing the angle between the hand and forearm. In finger movements, extension refers to the straightening of the fingers away from the palm, increasing the angle between the bones of the fingers and the palm and spreading the fingers apart
Abduction	A movement that takes a body part away from the midline of the body. In wrist movements, it refers to moving the hand away from the body's central axis, laterally
Adduction	The opposite of abduction, this movement brings a body part closer to the body's midline. In the context of wrist movements, it involves moving the hand towards the body's central axis
Pronation	A rotational movement where the hand and upper arm are turned inwards. For the wrist, it refers to rotating the forearm so the palm faces downwards
Supination	This movement is the opposite of pronation. It involves the rotation of the hand and upper arm outward, resulting in the palm facing upwards

Table of Contents

List of Figures

List of Tables

1. Introduction

In 2019, there were 12.2 million new stroke cases worldwide and 101 million existing cases. The total number of disability-adjusted life years (DALYs) due to stroke was 143 million, and the mortality was 6.55 million. Stroke remained the second most frequent cause of death (accounting for 11.6% of total deaths) and the third most common cause of death and disability combined (5.7% of total DALYs) in 2019. From 1990 to 2019, the number of stroke cases increased by 70%, stroke deaths by 43%, and DALYs by 32%[1].

In Europe in 2017, there were 1.12 million new stroke cases, 9.53 million stroke survivors, 0.46 million deaths, and 7.06 million disability-adjusted life years (DALYs) lost due to stroke. By 2047, the number of stroke cases is expected to increase by 40,000 (3%) and the number of existing cases by 2.58 million (27%)[2].

Stroke is the leading cause of acquired permanent disability worldwide. The majority of patients are left with impairments that affect their functional independence and quality of life [3].

Upper limb motor impairment, such as muscle weakness, loss of dexterous movement, and reduced sensation is a common manifestation after stroke, compromising independence in fundamental daily activities involving the ability to reach, grasp, and manipulate objects[4]. Upper limb dysfunction has a great negative impact on the quality of people's daily lives. Complicating or making impossible many of the essential everyday tasks[5]

It is very important to help people with upper limb dysfunction to become more independent in their daily life through rehabilitation. Occupational therapy is one of the most essential ways of curing the post insult rehabilitation[6]. A long intensive therapy process may be exhausting. Therapy gamification better motivates and engages patients in the process of the therapy, because with the usage of the gamified exercises patients tend to put more time into therapy exercises and exercise more carefully, therefore the positive effect of therapy significantly increases[7].

Furthermore digital solutions for occupational therapy can be more economically efficient because they enable real-time monitoring and feedback, which can reduce the frequency of direct therapist interventions and thereby lower costs[8][9].

Hand therapy digital solutions also may resolve some adjacent problems related to the therapy as cost, scheduling and attendance of the therapy classes. These are often reasons for therapy sessions missing or discontinuing hand therapy amongst patients[10][11]. Furthermore, home-based upper limb therapy may be more efficient than conventional methods of upper limb therapy[12].

The aim of this study is to develop and implement a comprehensive program for gamified therapy of the upper limbs, employing the innovative Leap Motion motion tracking technology[13]. The program is designed to improve the rehabilitation process in patients with upper limb dysfunction, providing a personalized and motivating approach to recovery. The primary objective is to create a unique therapeutic tool that combines scientifically based rehabilitation methods with elements of game design, making therapy more engaging and effective for patients.

The research also focuses on developing adaptive customization tools, allowing both medical professionals and patients themselves to adjust exercises for optimal levels of difficulty and effectiveness. The inclusion of a system for collecting and analyzing statistical data on exercise performance will enable real-time assessment and adjustment of the therapy program.

2. Background

In this section, the growing field of gamification in the context of upper limb rehabilitation is explored. Based on a range of academic studies, it is outlined how gamification can transform traditional approaches to rehabilitation, offering unique advantages in enhancing patient motivation and engagement.

The main goals and scope of the work are defined, clarifying how the project fits into existing research and what new opportunities it opens. Subsequently, a review of existing solutions in the field of hand rehabilitation is undertaken, assessing their applicability and effectiveness in solving current problems.

An important part of the analysis is the study of the hand rehabilitation situation in the local hospital, which allows for a deeper understanding of the unique challenges and needs of patients, as well as finding ways to meet them.

2.1 Problem Research

The gamification of occupational therapy for upper limbs represents a significant research problem, focusing on the impact of gamification on motivation, interest, and therapy outcomes in patients with disabilities or injuries. Occupational therapy assists people in developing or recovering skills necessary for daily tasks[14]. Gamification can be integrated into occupational therapy with the goal of stimulating motivation, satisfaction, and patient engagement[15].

There are also studies showing that therapy using digital solutions such as virtual reality for gamifying the process can demonstrate significantly higher results compared to traditional approaches. This is because gaming activities and their ability to adjust difficulty levels to the patient's needs lead to greater motivation among patients compared to traditional therapy[16].

Furthermore, research indicates that design patterns in gamified therapy games for hands, such as varying levels of difficulty, the ability to earn victory points or bonuses, and post-game feedback, make rehabilitation games more motivating and engaging for players. This increases the duration of gaming sessions and consequently the execution of exercises and the amount of effort exerted by the patient, which positively impacts patient progress

during rehabilitation[17].

2.2 Goal and Scope of the Work:

The goals of this work are as follows:

1. Development of software solutions for individualized gamified therapy of the upper limbs. Figure 1
2. Creation of a toolkit for adjusting exercise parameters, usable by both medical professionals and patients themselves, ensuring an optimal level of difficulty before starting the therapeutic session.
3. Development of a data collection system that records exercise performance during therapy, for subsequent analysis and evaluation of treatment effectiveness.
4. Implementation of gaming mechanics to stimulate patient interest and participation, which helps maintain regularity and quality of therapeutic tasks.
5. Adaptation and application of the exercise methodology proposed by Tallinn East Central Hospital[18], to ensure the scientific validity and clinical value of the therapeutic programs.
6. Investigation of potential barriers and challenges associated with the integration of gamification into occupational therapy, such as cost, accessibility, and applicability of technologies.

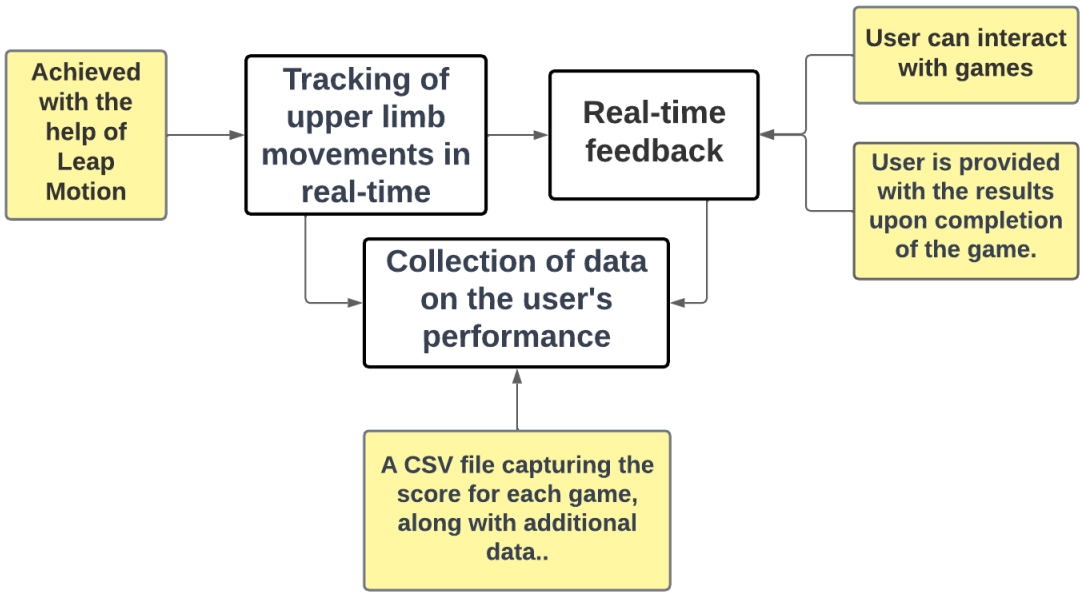

Figure 1. App Scheme

The goal of this work encompasses a comprehensive approach to the development and

testing of a software product, as well as assessing its applicability in a clinical setting to enhance the effectiveness of the rehabilitation process for patients suffering from upper-limb dysfunction.

Overall, conducting research on the topic of gamification in occupational therapy for upper limbs could significantly broaden the understanding of professionals about the possibilities of using gamification in rehabilitation processes and contribute to the development of new, more effective, and motivating treatment and rehabilitation methods.

2.3 Existing solutions

In the field of gamification of upper limb therapy, there are already some solutions that provide patients with an interactive and motivating rehabilitation experience. Below are examples of such solutions:

Neofect Smart Glove: The Neofect Smart Glove is a robotic glove used in conjunction with a mobile application for gamified rehabilitation following a stroke or injury. The glove tracks hand and finger movements and provides real-time feedback through games and exercises. [19]

Jintronix Rehabilitation System: Jintronix is a comprehensive rehabilitation software that utilizes Microsoft Kinect technology. It offers gamified exercises for upper and lower limbs, as well as for balance and coordination. [20]

Tyromotion AMADEO: AMADEO is a robotic system for hand rehabilitation, based on the principle of gamified learning. It offers a variety of games and exercises for developing hand and finger functions. [21]

MindMaze MindMotion GO: MindMotion GO is a portable platform for upper limb rehabilitation based on virtual reality. It offers a variety of games and scenarios for motor rehabilitation and cognitive stimulation. [22]

Existing solutions in the field of gamification for upper limb therapy have demonstrated significant potential in improving patient outcomes and providing a motivating rehabilitation experience. However, the development of new games and approaches can help expand opportunities for different patient groups and meet individual needs. Additionally, most current gamified therapy solutions for the hands require the patient to wear special gloves, hold a joystick, or have attachments on the hand during therapy. Despite the effectiveness of this approach, it necessitates the patient wearing or holding an additional device. This

can be inconvenient and off-putting for the patient and may limit the freedom of limb movement, not fit properly, and so forth. In settings where the tool is used for a large number of patients, such as hospitals, this solution may not be hygienic.

Additionally, due to the large volume of specialized sensors, such technologies are quite expensive and highly specialized. This makes them inaccessible to many patients and hospitals. These shortcomings can be addressed by using solutions that do not require the patient to wear or hold any additional devices on their hand. For example, a device like Leap Motion subsection 3.3.2, which allows for the remote reading of a patient's hand movements, can be a viable alternative. It is also relatively inexpensive and multi-functional, making it more widely usable.

2.4 Issue of Hand Therapy in Local Hospital

East Tallinn Central Hospital is in need of a digital gamified solution for occupational therapy of the upper limbs.

Gamification of hand therapy can help achieve the following advantages for East Tallinn Central Hospital:

1. Improved Patient Motivation: Gamification makes therapy more interesting and engaging for patients, which in turn can increase their motivation to perform exercises and improve adherence to medical recommendations.
2. Increased Therapy Effectiveness: Gamified therapy programs can encourage patients to frequently and regularly perform exercises, which may lead to better outcomes and faster recovery.
3. Reduced Staff Workload: Gamification can ease the workload of doctors and therapists, as patients engaged with gaming elements may perform exercises more independently, reducing the time spent with healthcare personnel.
4. Individualized Approach: Gamified programs can be easily tailored for each patient, taking into account their individual characteristics, preferences, and progress, which can enhance treatment outcomes.

East Tallinn Central Hospital has provided a list of upper limb rehabilitation exercises that could benefit from gamification. These exercises, developed according to the hospital's methodology and actively used in their practice, served as a basis for determining which games would be developed as part of this work.

3. Implementation

In this section, the focus is on the practical aspects of the gamified hand therapy project. It includes the selection and design of exercises for gamification, development of engaging game scenarios, and a detailed look at the technological tools used in the project. The section also covers the optimization of the user experience, the structural design of the application, and a exploration of the development process for each therapeutic game.

3.1 Exercises

When selecting exercises for gamification, it is important to consider several key aspects:

1. The exercise should be well-tracked using Leap Motion:
   Leap Motion allows for the tracking of hand and finger positions, but the technology is not yet perfect, and overly complex or rapid changes in hand and finger positions might not always be perfectly captured. Therefore, one of the main criteria for selecting exercises is the ability of Leap Motion to track them accurately, maximizing the effectiveness of the exercises and enhancing the user experience.
2. Exercises should cover various muscle groups and types of upper limb movements:
   Proper functioning of the upper limbs requires coordinated movement of different parts of the arm, finger movements, fist clenching, rotational, horizontal, and vertical wrist movements, etc. The chosen exercises should encompass the widest possible range of upper limb movements for a diverse and effective therapy.
3. Exercises most popular among physiotherapists:
   The selection of exercises is based on the recommendations and needs of the physiotherapists at East Tallinn Central Hospital. This ensures that the exercises meet clinical requirements and are effective in the rehabilitation process.
4. Exercises can be combined:
   A good practice is the ability to combine movements from different exercises to create individual and more complex routines. This allows the rehabilitation program to be tailored to the specific needs of each patient, improving outcomes and making the process more interesting and motivating. Additionally, the gameplay of games created based on a combination of exercises becomes deeper and more engaging.

Based on the listed criteria, a comprehensive set of exercises has been compiled that provides a holistic approach to the rehabilitation of the upper limbs.

1. Abduction (Figure 2) and Adduction (Figure 3) of the Wrist: These exercises involve slowly moving the hand away from and towards the body, helping to strengthen the muscles responsible for side-to-side wrist movements. Leap Motion effectively tracks such movements, as they involve clear and distinct changes in position.

Figure 2. Wirst abduction

Figure 3. Wirst adduction

2. Pronation (Figure 4) and Supination (Figure 5) of the Wrist: Pronation (turning the palm downward) and supination (turning the palm upward) of the wrist are important for daily hand functions. Exercises that use these movements are also well-suited for the Leap Motion system, as they involve relatively simple and controlled rotational movements.

Figure 4. Wirst pronation

Figure 5. Wirst supination

3. Flexion (Figure 7) and Extension (Figure 6) of the Wrist: Wrist flexion involves moving the back of the hand towards the forearm, while extension involves moving the palm away from the forearm. These movements are key for many daily tasks such as writing, holding objects, and lifting things. Exercises for wrist flexion and extension are also ideally suited for motion tracking systems like Leap Motion, as

they require clearly defined and easily measurable movements in the plane of the wrist.

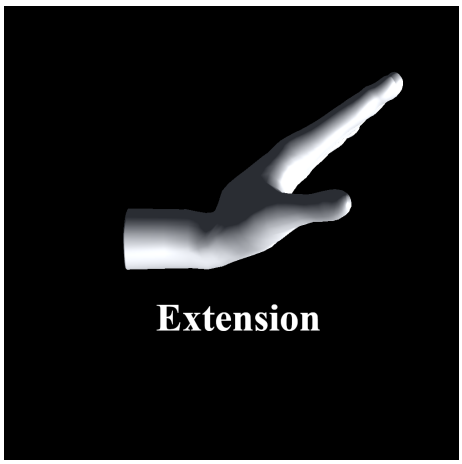

Figure 6. Wirst extension

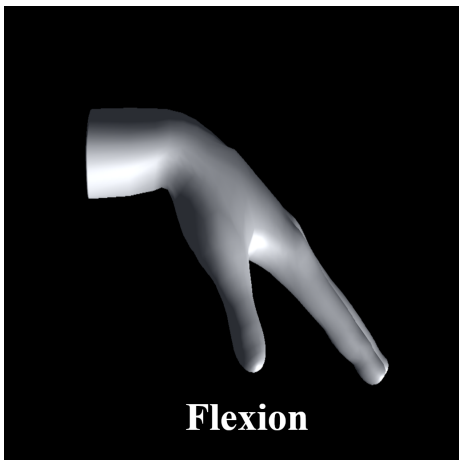

Figure 7. Wirst flexion

4. Finger Flexion (Figure 9) and Extension (Figure 8): Finger flexibility exercises are particularly important as they improve dexterity and contribute to the rapid recovery of basic hand functions. Leap Motion can track these actions, although it requires the user to perform movements slowly and clearly for better recognition.

Figure 8. Fingers Extension

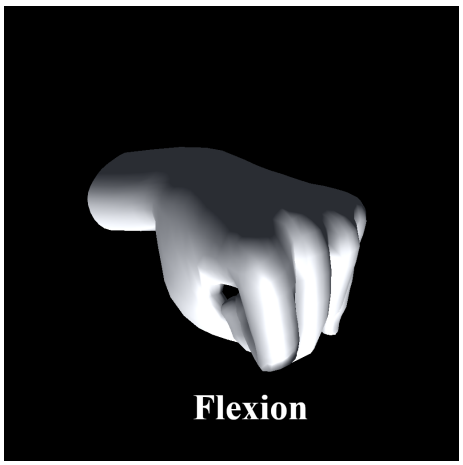

Figure 9. Fingers flexion

3.2 Preparation of the game structure and scenarios

Before starting the development of this program, game scenarios must be written. The main goal of each game is to develop movements of the wrist, forearm, and fingers in different planes. The first game is focused on developing the abduction and adduction movements of the wrist. The second game is centered on pronation and supination. The third game combines flexion, extension, and abduction, adduction, pronation, and supination of the wrist. The fourth game focuses on the flexion, extension, abduction, and adduction of the fingers.

3.3 Equipment used

This section examines the crucial equipment utilized in the study. The emphasis is on three key elements: Game Engine Software, Leap Motion, and the UltraLeap Plugin for Unity. The functionalities and importance of these technologies in the context of the work will be briefly outlined.

3.3.1 Game engine Software

In the domain of game development, gaming engines stand as the cornerstone software platforms, providing developers with the tools and capabilities necessary to bring their creative visions to life. These engines, like Unity [23] and Unreal Engine [24], are integral in the development of a wide range of interactive applications, including video games, simulations, and gamified solutions for various sectors such as education, healthcare, and entertainment. Their versatility and advanced features make them ideal for integrating technologies like Leap Motion, especially in projects that require sophisticated hand and finger tracking capabilities, as seen in gamified therapeutic applications.

Integration of Leap Motion with Gaming Engines: Unity and Unreal Engine

Leap Motion technology, noted for its precision in tracking hand and finger movements, is compatible with two of the leading gaming engines in the industry: Unity and Unreal Engine. Both engines offer a comprehensive suite of tools and functionalities that are crucial for developing interactive and immersive applications. Notably, both Unity and Unreal Engine are accessible for non-commercial projects through free versions, making them attractive options for academic and research-focused projects.

Decisive factors for Choosing Unity

While both Unity and Unreal Engine are equipped to support Leap Motion and offer free access for non-commercial uses, the selection of Unity as the development environment for this thesis was driven by specific considerations:

1. Author's Proficiency in Unity and C#: The author's extensive experience with Unity and the C# programming language is the primary factor influencing the choice of Unity. With over three years of hands-on experience in Unity and C# development, the author is adept in navigating the intricacies of the Unity environment and utilizing the full potential of C# scripting. This expertise ensures an efficient development

process, allowing the author to concentrate on the innovative aspects of the project without the need for extensive acclimatization to a new development environment.

2. Unity's User-Friendly Interface and Flexibility: Unity is known for its user-friendly interface and adaptable development environment, which are particularly advantageous for projects that involve complex technologies like Leap Motion. The engine's straightforward workflow, combined with comprehensive documentation, streamlines the implementation of complex functionalities and aids in effectively addressing any development challenges.
3. Scripting Efficiency of C#: The use of C# in Unity is another significant advantage. As a powerful and versatile programming language, C# offers robust features and a strong typing system, essential for crafting sophisticated gamified applications. The author's fluency in C# enables the creation of efficient, maintainable, and scalable code, which is vital for the success of the project.

3.3.2 Leap Motion

The Leap Motion Controller[13] is a small USB device developed by Leap Motion, a technology company specializing in motion tracking and gesture control. This device is designed to track hand movements and finger positions with high accuracy, allowing users to interact with their computers, virtual reality (VR) headsets, and other devices using natural hand gestures. The Leap Motion Controller uses infrared cameras and advanced software algorithms to detect and track the user's hand and finger movements in real-time. The device creates a 3D interaction volume around itself, providing precise tracking of hand and finger positions within this space.

3.3.3 Ultra leap plugin for Unity

The Ultraleap plugin version 6.5.0 for Unity[25] is a SDK that enables developers to integrate Ultraleap's hand tracking and haptic feedback technology into their Unity projects. The Ultraleap plugin for Unity provides an intuitive and easy-to-use interface, making it possible for developers to implement advanced hand tracking and haptic interactions in their projects. It allows users to interact with virtual objects using natural hand movements and gestures, enhancing immersion and user experience in virtual reality (VR), augmented reality (AR), and mixed reality (MR) applications. By combining hand tracking and haptic feedback, the Ultraleap plugin for Unity enables developers to create more natural and immersive user experiences, pushing the boundaries of interactive content in gaming, simulation, education, and more.

3.4 User experience optimization

Leap Motion calibration

To ensure effective and productive therapy, it is crucial to apply an individualized approach to each patient. It should be taken into account that different patients may have significantly varying characteristics, such as the degree of limb mobility, permissible range of motion, speed of exercise execution, response to certain stimuli, rehabilitation needs, and many other aspects.

In this regard, when developing rehabilitation games, special attention should be paid to creating a flexible system of settings and calibration, which would be able to consider the capabilities and needs of the widest range of patients. Such a system should provide adaptability both in terms of the game settings themselves and in relation to working with the Leap Motion device and the application as a whole.

The Leap Motion calibration process involves setting the permissible amplitude of motion for the patient's upper limbs in various planes, which, in turn, allows the games to be adequately adapted to the individual characteristics of each patient.

Let's consider an example of such adaptation. Suppose in one of the games, interaction with the virtual environment is carried out by clenching the hand into a fist. However, some patients may have difficulty fully clenching their hand. In this case, during the calibration stage, the system should determine the degree of clenching available to the patient and make corresponding changes to the game settings to ensure comfortable and effective interaction with the game content. Thus, customization of settings and calibration are key components of successful rehabilitation therapy using Leap Motion and specialized games.

3.5 Application structure

The application has the following structure:

1. Launch: Upon launching the application, the user is taken to the main menu.
2. Main Menu: This contains access to general settings, four games, and an option to exit the application. The main menu serves as the central navigation point, allowing the user to select the game of their interest. The menu consists of several blocks, each responsible for selecting a game; each block displays an exercise and an image from the game. When a block is selected, the *MenuManager.cs* script loads the

corresponding game scene.

3. General Settings: A section where the user can adjust general application settings. The *GeneralSettings.cs* script controls key features such as volume adjustment, providing a user-friendly interface for customizing basic application settings.
4. Games: Inside each game, there is a settings menu where game parameters can be adapted to meet the individual needs of the user. After selecting settings, the user enters the action scene where the main gameplay activity takes place. After completing the game session, the results are displayed on the score panel.
5. Exit: An option to terminate the application.

The completed user flow can be seen on the diagram presented in (Figure 10).

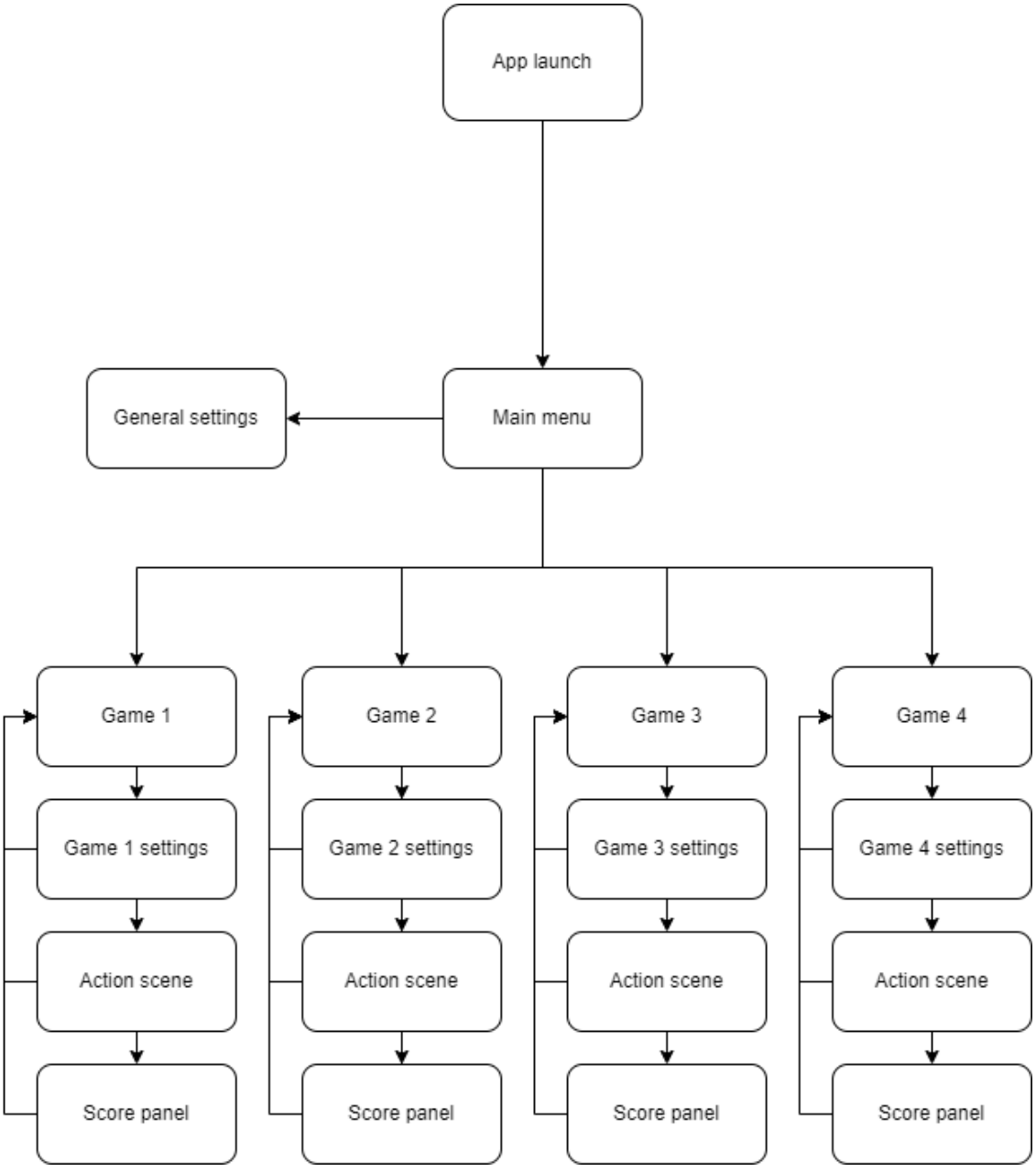

Figure 10. App diagram

3.6 Game implementations

In this section, the focus is on the practical aspects of converting the theoretical framework into functional gamified exercises for hand therapy. This involves designing interactive games tailored to specific rehabilitation needs, integrating the previously discussed technologies such as Unity, Leap Motion, and the UltraLeap Plugin.

3.6.1 Game 1: Ship

Exercises

The exercise is designed to restore and improve the function of the upper limbs in patients with upper limb dysfunction by developing the strength, flexibility, and coordination of wrist muscles during the process of controlling a ship in the game.

The patient sits on a chair in front of the game screen, with a straight back, arms hanging down along the body. Raises the right (or left) arm, bending it at the elbow joint, so that the forearm is parallel to the floor, and the palm is facing downwards. Begins to turn the wrist to the right (Adduction) to steer the ship to the right and to the left (Abduction) to steer the ship to the left. The patient must dodge obstacles and collect bonus items to earn points. After completing the game, returns the hand to the starting position and repeats the exercise with the other hand if necessary.

Gameplay

The aim of the game is to provide effective rehabilitation for the patient's upper limbs, focusing on improving the mobility and coordination of the wrist. The player controls a ship, dodging obstacles and collecting bonus items to score points.

The player steers the ship by performing wrist abduction and adduction movements. The angle of the wrist's rotation directly affects the course of the ship, allowing the player to dodge obstacles and collect bonuses.

Throughout the game, the player encounters obstacles in the form of debris that must be avoided. There are also various bonus items along the way, collecting which will allow the player to score additional points.

After completing the game, the player will see the number of points scored during the

game.

Implementation Wrist Movement Tracking Using Leap Motion, the game accurately tracks wrist abduction and adduction movements. These movements are translated into game commands, which control the ship's movement to the left and right, respectively.

Conversion of Movements into Controls When the player performs a wrist movement, Leap Motion recognizes the angle of the wrist's rotation and converts it into movement commands for the ship. A script analyzes data from Leap Motion, converts it into a vector of course change, and dynamically adapts the position of the ship in the game, allowing the player to avoid obstacles and collect bonuses.

Ship Control Implementation:

1. Capturing the Player's Hand The method begins by acquiring the current frame from Leap Motion *currentFrame*. If the frame contains hand data, the first detected hand *_currentHand* is selected.
2. Determining the Wrist Rotation Angle:
   For the initial setup, the angle between the palm position and the wrist, saved as *_previousWristRotationAngle*, is used. In subsequent iterations, the current wrist rotation angle *currentWristRotationAngle* is updated, also calculated based on the angle between the palm position and the wrist.
3. Calculating the Rotation Difference: The *rotationDifference*, which is the difference between the current and previous wrist rotation angles, is computed. Adjustments are made to account for the limits of rotation (for example, if the angle passes through 0 degrees).
4. Updating the Ship's Position:
   Based on the obtained rotation difference *rotationDifference*, the ship is moved horizontally *newPos.x*. The ship's movement *speed* is multiplied by *rotationDifference* to determine the extent of the position change (Figure 11).
5. Applying the New Position:
   The updated position *newPos* is applied to the ship object, resulting in its movement within the game space.

Additionally, the game includes the *ShipGameManager.cs* class for managing the main game process. *ObstaclesSpawner.cs* is used for creating obstacle objects, and *ObstacleBehaviour.cs* manages the logic of obstacles.

Game Dynamics and Feedback

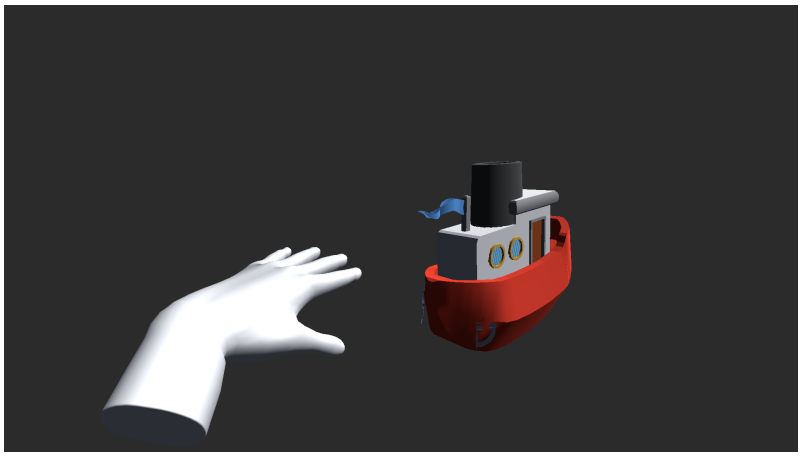

Figure 11. Ship controll

Obstacles and bonus items are distributed throughout the game space in such a way as to encourage the player to frequently use wrist Abduction and Adduction movements, thereby providing a therapeutic effect. The game offers immediate feedback to the player by visualizing successful manoeuvres and accumulated points.

Personalization

The game accommodates the possibility of adaptation to the individual characteristics and needs of each patient. The level of difficulty can be modified by adjusting the following parameters:

1. Adjusting the Speed of the Game The speed of the game can be adjusted within a range from 1 to 10 arbitrary units, allowing for the individualization of game difficulty for each patient. A lower speed provides more time for decision-making and reaction to emerging obstacles. The highest speed, conversely, imposes higher demands on reaction speed. The standard set speed is 5 arbitrary units.
2. Adapting the Duration of the Gaming Session The duration of the gaming session can be adapted from 30 seconds to 5 minutes, accommodating the individual characteristics and needs of patients at different stages of rehabilitation. Patients experiencing difficulties with arm movement and physical fatigue due to limb issues can opt for shorter sessions to avoid overexertion. However, longer sessions offer the opportunity for gradual increase in physical activity and stimulation of the recovery process.
3. Adjusting the Arm Tilt Angle The ability to change the arm tilt angle offers broad prospects for adapting the game to patients with varying degrees of joint mobility. Reducing the range of arm movement makes the game accessible even for those with limited mobility, thereby enhancing each movement. At the same time, the option to

increase the range can encourage patients to perform movements in their full extent.

3.6.2 Game 2: Plane

Exercises

The game exercises is aimed at developing forearm muscles, as well as improving wrist flexibility and coordination. During the exercise, the player must slowly and smoothly rotate the forearm, changing the hand position from facing downwards to facing upwards, with the default hand position being perpendicular to the surface.

The patient sits on a chair in front of the game screen, back straight, arms hanging along the body. Raises the right (or left) hand, bending it at the elbow joint so that the forearm is parallel to the floor, and the palm is perpendicular to the surface. Begins to rotate the forearm clockwise (Supination) to steer the plane to the right and counterclockwise (Pronation) to steer the plane to the left. The patient must avoid obstacles and collect bonus items to earn points. After completing the game, the patient returns the hand to the initial position and repeats the exercise with the other hand if necessary.

Gameplay

The player needs to control the plane, dodge obstacles, and collect bonus items to score points. The player controls the plane by rotating the palm. The angle of the palm rotation directly affects the change in the movement vector of the plane, allowing the player to dodge obstacles and collect bonuses. The rotation of the hand simulates the movement of the plane's wings, with the palm's rotation tilting the plane in the corresponding direction, causing it to change its course. Throughout the game, the player encounters various obstacles that must be avoided. There are also various bonus items appearing on the player's path, collecting which allows earning additional points.

The number of points earned depends on the successful collection of bonuses and evasion of obstacles.

Implementation

Defining the control mechanics: The primary input method is Leap Motion technology, which accurately recognizes the player's hand movements. The *PlaneController.cs* script implements the control of the plane, analyzing the player's palm rotational movement around the z-axis.

The main game logic, including the scoring system, generation of obstacles and bonuses on the playing field, and implementation of the plane's movement, was developed on the Unity platform using the C# programming language.

Plane Control Implementation

1. Capturing the Player's Hand The method begins by acquiring the current frame from Leap Motion *currentFrame*. If the frame contains hand data, the first detected hand *_currentHand* is selected.
2. Determining the Palm Normal: We obtain the normal vector to the palm. If the hand is flat, this vector will point downward. *Vector3 normal = hand.PalmNormal;*
3. Calculating the Rotation Angle: The palm normal is projected onto a plane perpendicular to the vertical
   Vector3 projectedNormal = Vector3.ProjectOnPlane(normal, Vector3.up);
   The rotation angle *yaw* of the palm is calculated relative to the forward-facing vector
   yaw = -Vector3.SignedAngle(Vector3.forward, projectedNormal, Vector3.up)
   We calculate the plane's rotation angle considering speed and time
   *var rotateAngleY = (new Vector3(0, yaw, 0) * Time.deltaTime * rotateSpeed).y*
4. Updating the Plane's Position: The obtained movement value is translated into a vector, which is then applied to the tilt of the plane in the game, ensuring intuitive and smooth control (Figure 12).
   RotatePlane(new Vector3(0, rotateAngleY, 0))

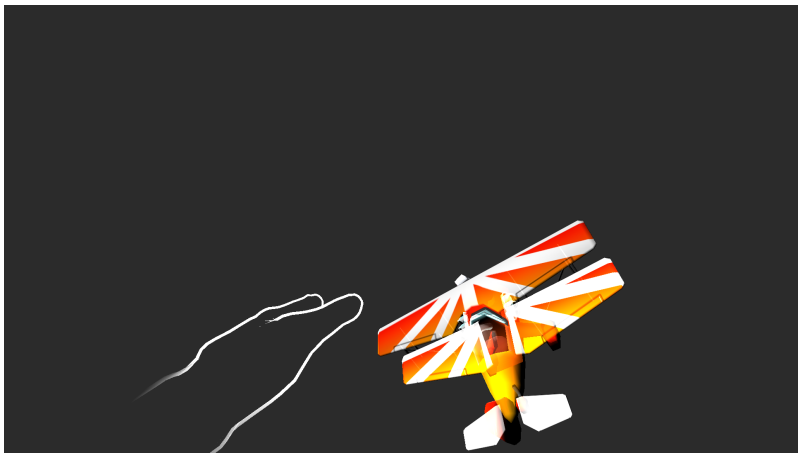

Figure 12. Plane controll

Also, the game includes the *HandPlaneGameManager.cs* class for managing the main game process, *ObstaclesSpawner.cs* for creating obstacle objects, and *ObstacleBehaviour.cs* for managing the logic of obstacles.

Game Dynamics and Feedback

Obstacles and bonus items are distributed throughout the game space in such a way as to encourage the player to frequently use Supination and Pronation wrist movements, thereby providing a therapeutic effect. The game provides immediate feedback to the player by visualizing successful maneuvers and the points scored.

Personalization

Personalization

The game accommodates the possibility of adaptation to the individual characteristics and needs of each patient. The level of difficulty can be modified by adjusting the following parameters:

1. Adjusting the Speed of the Game The speed of the game can be adjusted within a range from 1 to 10 arbitrary units, allowing for the individualization of game difficulty for each patient. A lower speed provides more time for decision-making and reaction to emerging obstacles. The highest speed, conversely, imposes higher demands on reaction speed. The standard set speed is 5 arbitrary units.
2. Adapting the Duration of the Gaming Session The duration of the gaming session can be adapted from 30 seconds to 5 minutes, accommodating the individual characteristics and needs of patients at different stages of rehabilitation. Patients experiencing difficulties with arm movement and physical fatigue due to limb issues can opt for shorter sessions to avoid overexertion. However, longer sessions offer the opportunity for gradual increase in physical activity and stimulation of the recovery process.
3. Adjusting the Arm Tilt Angle The ability to change the arm tilt angle offers broad prospects for adapting the game to patients with varying degrees of joint mobility. Reducing the range of arm movement makes the game accessible even for those with limited mobility, thereby enhancing each movement. At the same time, the option to increase the range can encourage patients to perform movements in their full extent.

3.6.3 Game 3: Wrist sniper

Exercises In the first exercise, the patient must rotate the wrist in various planes, for example, from top to bottom, diagonally, or from left to right and vice versa. During the exercise, it's important to make smooth and slow movements, avoiding jerks and tension. This will effectively warm up and train different groups of hand and wrist muscles.

In the second exercise used in this game, the patient needs to clench their hand into a fist,

slowly and smoothly bending all fingers simultaneously.

Gameplay

The player will control an aim and pop flying balloons on the screen by clenching their hand into a fist. The player controls the aim by rotating the wrist in different directions. The direction of wrist rotation directly affects the change in the aim's movement direction, allowing the player to aim at flying balloons. Wrist rotations control the movement of the aim; the aim moves in the direction the player's wrist is turned. Throughout the game, the player aims the aim at balloons and pops them by clenching their hand into a fist. Various bonus items will also appear on the screen, hitting which will give the player additional points and other bonuses.

The number of points earned depends on the number of balloons popped by the player. Points serve as an indicator of the patient's progress and also encourage further rehabilitation.

Implementation

The primary input method is Leap Motion technology, which allows for high-precision recognition of the player's hand movements. The *AimController.cs* script implements the control of the aim, analyzing the rotational movement of the player's palm around the z-axis and along the x and y-axis.

The main game logic, including the scoring system, generation of balloons on the playing field, and implementation of aim movement, was developed on the Unity platform using the C# programming language.

Aim Control Implementation:

1. Capturing the Player's Hand:
   The method begins by acquiring the current frame from Leap Motion *frame = leapServiceProvider.CurrentFrame;*. If the frame contains hand data, the first detected hand *_currentHand* is selected.
2. Determining the Position and Orientation of the Wrist: The wrist position *wristPosition = _currentHand.WristPosition* is used as the starting point for motion tracking.
3. Calculating the Orientation of the Palm and Wrist:
   Based on the detected hand, the wrist rotation is calculated. The palm normal

Vector3 palmNormal = hand.PalmNormal and palm direction *Vector3 palmDirection = hand.Direction* are determined. Then, the wrist rotation *Quaternion wristRotation = Quaternion.LookRotation(palmDirection, -palmNormal)* is calculated.

4. Determining the Direction of aim Movement:
   Based on the rotations of the wrist and forearm, movement directions are calculated *Vector3 wristForward = wristRotation * Vector3.forward* and *Vector3 upperArmForward = upperArmRotation * Vector3.forward*. These vectors are used to determine the overall direction of aim movement.
5. Applying the Calculated Direction to Control the aim:
   The calculated movement direction *movementDirection = wristForward + upperArmForward* is normalized and used to determine the new position of the aim. The aim moves in the direction corresponding to the player's wrist movement *transform.position += movementDirection * Time.deltaTime * speed* (Figure 13).
6. Interacting with Game Objects:
   When the aim is aimed at balloons and the hand is clenched into a fist, the balloons pop, and the player is awarded points.
   if (hand.GrabStrength + playerGrabStrenghtDelta > 0.95f) Destroy(collider.gameObject); WristSniperGameManager.AddPoint.Invoke();

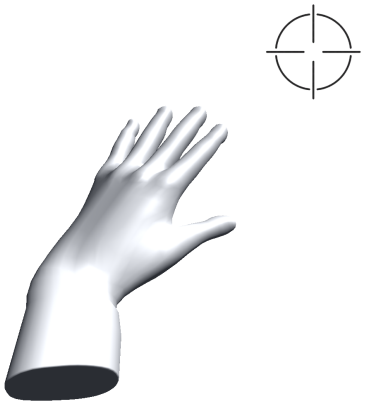

Figure 13. Aim controll

Also, the game includes the classes *WristSniperGameManager.cs* for managing the main process of the game, *BalloonsSpawner.cs* for creating balloons, and *BalloonBehaviour.cs* for managing the logic of balloons.

Game Dynamics and Feedback

Targets are distributed throughout the game space in such a way as to encourage the player to frequently use precise and controlled hand and finger movements, thereby providing a therapeutic effect. The game offers immediate feedback to the player by visualizing

successful shots and accumulating points. Thus, the gameplay involves controlling the aim using natural wrist movements, ensuring intuitive and smooth control during the execution of the exercise.

Personalization

The game accommodates the possibility of adaptation to the individual characteristics and needs of each patient. The level of difficulty can be modified by adjusting the following parameters:

1. Adjusting the Speed of the Game The speed of the game can be adjusted within a range from 1 to 10 arbitrary units, allowing for the individualization of game difficulty for each patient. A lower speed provides more time for decision-making and reaction to emerging obstacles. The highest speed, conversely, imposes higher demands on reaction speed. The standard set speed is 5 arbitrary units.
2. Adapting the Duration of the Gaming Session The duration of the gaming session can be adapted from 30 seconds to 5 minutes, accommodating the individual characteristics and needs of patients at different stages of rehabilitation. Patients experiencing difficulties with arm movement and physical fatigue due to limb issues can opt for shorter sessions to avoid overexertion. However, longer sessions offer the opportunity for gradual increase in physical activity and stimulation of the recovery process.
3. Adjusting the Arm Tilt Angle The ability to change the arm tilt angle offers broad prospects for adapting the game to patients with varying degrees of joint mobility. Reducing the range of arm movement makes the game accessible even for those with limited mobility, thereby enhancing each movement. At the same time, the option to increase the range can encourage patients to perform movements in their full extent.

3.6.4 Game 4: Piano

Exercises Sequential bending of fingers downwards from a palm-down hand position. During the exercise, the patient sits at a table with the forearm comfortably positioned on the surface, and the palm turned downwards. The patient bends each finger in turn, starting from the thumb and ending with the little finger, holding the bend for 1-2 seconds before returning the finger to its original position.

Gameplay

In this game, five strips of different colors are displayed on the screen, each corresponding

to a specific finger on the player's hand. During the game, a portion of each strip lights up and moves toward the player.

The goal of the game is for the player to lower the corresponding finger at the right moment and hold it on the lit strip as it passes by. The player needs to carefully time the movement of their fingers and maintain the correct position to successfully interact with the moving illuminated section.

Implementation

In the game, five colored strips on the screen correspond to the five fingers of the player's hand. When a part of a strip lights up and moves towards the player, they must lower the corresponding finger at the right moment to "capture" the illuminated section. Using the *DetectTargetFingerBend* method, the game determines whether the player has correctly performed the action and provides visual feedback. The effectiveness of the game largely depends on the accuracy and timing of the player's response, which helps develop coordination and motor skills of the fingers.

1. Determining Player Finger Activity:
   The method begins by acquiring the current frame of data from Leap Motion *frame = leapController.Frame()*. If the frame does not contain hand data (less than one hand), the method stops executing.
2. Selecting the Target Finger:
   A finger is selected that will be tracked in the current frame for the current strip *Finger targetFinger = hand.Fingers[targetFingerIndex]*.
3. Determining the Finger's Position Relative to the Palm:
   It checks whether the target finger is lowered below a certain threshold relative to the palm *targetFinger.TipPosition.y < hand.PalmPosition.y - loweredThreshold* (Figure 14).
4. Responding to Finger Position Change:
   If the finger is lowered and was not previously lowered *!fingerLowered*, the flag *fingerLowered = true* is set, and the color of the control element (image) changes to target color, signaling successful task completion .
   If the finger is raised, the *fingerLowered* flag is reset and the control element's color returns to its original.

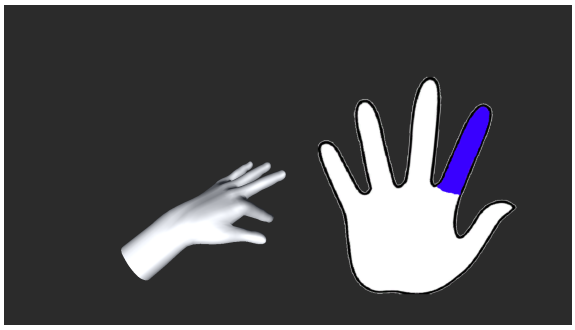

Figure 14. Extended Index Finger: Motion Tracking Example

Additionally, the game includes the classes *PianoGameManager.cs* for managing the main game process and *LaneBehaviour.cs* for managing the behavior of the lines.

Game Dynamics and Feedback

Game dynamic arranged to encourage the player to use diverse finger movements and coordination. This setup is designed to provide therapeutic benefits by improving finger dexterity and hand coordination. The game offers immediate feedback by visualizing correct and incorrect key presses.

Personalization

The game provides the possibility of adaptation to the individual characteristics and needs of each patient. The level of difficulty can be changed by adjusting the following parameters:

1. Adjusting the Speed of the Game The game speed can be adjusted within a range from 1 to 10 arbitrary units, allowing for the individualization of the game's difficulty for each patient. At lower speeds, the participant has more time to make decisions and react to emerging obstacles. The highest speed, correspondingly, creates higher demands on reaction speed. The standard set speed is 5 arbitrary units.
2. Adapting the Duration of the Gaming Session The duration of the gaming session can be adapted from 30 seconds to 5 minutes, accommodating the individual characteristics and needs of patients at different stages of rehabilitation. Patients experiencing difficulties with arm movement and physical fatigue due to limb problems can opt for shorter sessions to avoid overexertion. However, longer sessions offer the opportunity for a gradual increase in physical activity and stimulation of the recovery process.
3. Adjusting the Finger Bending Angle The ability to change the finger bending angle offers broad prospects for adapting the game to patients with varying degrees of joint

mobility. Reducing the range of finger movement makes the game accessible even for those with limited mobility, thereby enhancing each movement. At the same time, the option to increase the range can encourage patients to perform movements to their full extent.

3.6.5 Data Collection

To enable medical staff to track the results of patient exercises during the use of this application, it is necessary to implement data collection and storage.

For this task, there is an object on the main scene named *DataCollectionManager* with an attached script *DataCollectionManager.cs*. This script implements the singleton pattern, and its instance is available for recording user results after the game is completed. This class writes data to a CSV file, where it is available for further analysis by the patient and the treating physician.

Approximate Data Table Structure:

Table 1. Game Session Data

Session Start: 2023-11-14 15:30					
Game Name	Duration (min)	Speed (level)	Hand (Right/Left)	Limb Movement Amplitude (degrees)	Success Rate (%)
Game 1	0:30	3	Right	45	90
Game 2	2:00	7	Left	60	75
Game 3	1:30	5	Right	90	80
Game 4	4:30	10	Left	30	85
Session End: 2023-11-14 16:30					

4. Games verification

This section undertakes the verification of the functionality of the four developed games: Ship, Plane, Wrist Sniper, and Piano. It includes gameplay screenshots and detailed descriptions of the implemented functionalities for each game.

4.1 Game 1: Ship

The game "Ship" is designed to aid in effective hand therapy, focusing on enhancing wrist mobility and coordination. The gameplay involves controlling a ship to dodge obstacles and collect bonus items, thereby earning points. This control is achieved through wrist abduction and adduction movements, where the wrist's rotation angle directly influences the ship's course. This mechanism allows for navigating through obstacles and gathering bonuses (Figure 15).

Figure 15. Game "Ship"

Leap Motion technology is utilized to accurately track wrist movements of abduction and adduction. These movements are translated into game commands that control the ship's left and right movements. When a movement is made, Leap Motion detects the wrist's rotation angle and converts it into the ship's movement commands. A script processes the Leap Motion data, converting it into a vector for changing course and dynamically adjusting the ship's position in the game (Figure 16). This enables avoiding obstacles and collecting bonuses.

The game dynamics and feedback are structured to encourage frequent use of wrist abduction and adduction movements, thus providing therapeutic benefits. Immediate feedback is given to the player through visualizations of successful maneuvers and accumulated points.

Figure 16. Game "Ship"

Moreover, the game is designed to be adaptable to the unique characteristics and needs of each patient. The difficulty level can be customized by adjusting various parameters, such as the ship's speed, game length and player wrist movement amplitude (Figure 17).

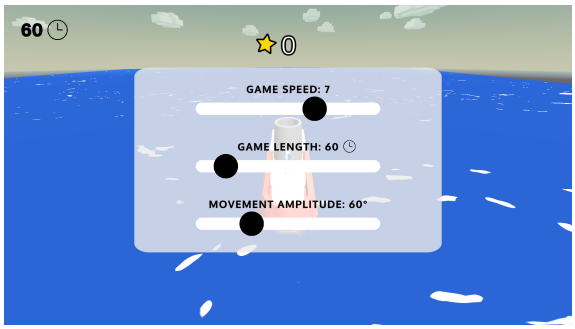

Figure 17. Game "Ship" settings window

4.2 Game 2: Plane

The game "Plane" is a game specifically developed to facilitate hand therapy, focusing on exercises that involve wrist pronation and supination. The gameplay requires players to control an airplane, with the airplane's movement directly linked to the player's wrist actions. Pronation (rotating the forearm counterclockwise) causes the airplane to veer left, while supination (rotating the forearm clockwise) makes it turn right (Figure 18). This design allows players to navigate through in-game obstacles and collect bonus items, thus earning points and demonstrating their proficiency in both the exercise and the game.

Figure 18. Game "Plane"

The technology used for tracking these specific wrist movements and translating them into game actions would be crucial in this setup. The accurate detection and interpretation of pronation and supination movements ensure smooth and intuitive control of the airplane in the game environment (Figure 19).

Figure 19. Game "Plane"

The game's dynamics and feedback system are designed to promote the regular practice of these wrist movements, contributing to therapeutic outcomes. Players receive immediate feedback on their performance through visual cues that indicate successful maneuvers and the points they've accumulated. This not only motivates continued engagement but also provides a clear measure of progress in both the therapeutic exercise and the game.

Additionally, the game offers adaptability to meet the unique needs and abilities of each player. Parameters such as the airplane's speed, game length and player wrist movement amplitude (Figure 20). This flexibility ensures that the game can be tailored to various stages of hand therapy, accommodating a wide range of patient abilities and rehabilitation goals.

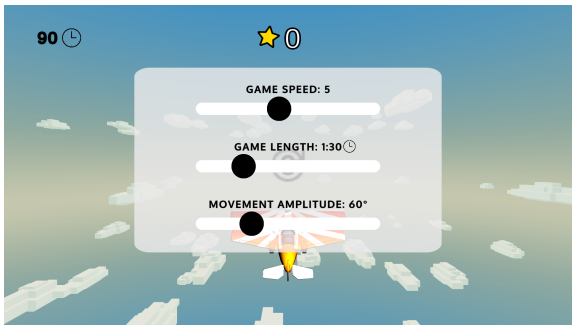

Figure 20. Game "Plane" settings window

4.3 Game 3: Wrist sniper

The game "Wrist Sniper" is specifically developed to aid in hand therapy, focusing on exercises involving various wrist movements like pronation, supination, adduction, and abduction. The gameplay requires players to control a crosshair, using wrist movements to aim at and pop balloons appearing on the screen. The direction of the wrist's rotation directly influences the crosshair's movement, allowing the player to target the balloons accurately (Figure 21). Additionally, popping these balloons is achieved by squeezing the hand into a fist, representing flexion and extension of the fingers, thereby enabling the player to earn points.

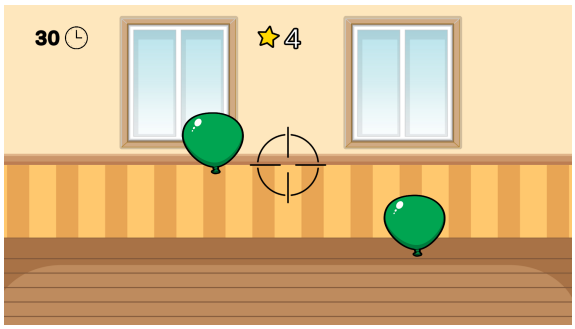

Figure 21. Game "Wrist sniper"

Leap Motion technology plays a critical role in the game, providing high-precision recognition of the player's hand movements. The game's script, AimController.cs, manages the control of the crosshair by analyzing the rotational movement of the player's palm. This analysis is done around the z-axis and along the x and y axes (Figure 22). The game's primary logic, including the scoring system, the generation of balloons on the playfield, and the implementation of the aim's movement.

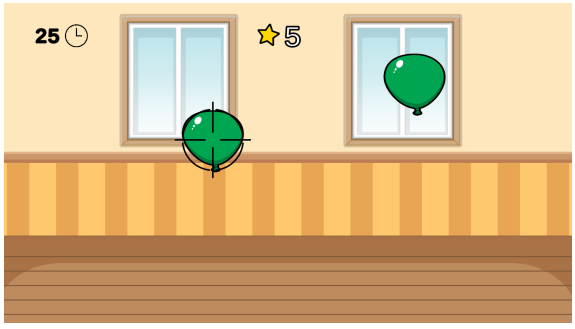

Figure 22. Game "Wrist sniper"

The game's design ensures that players are continuously engaged by the need to adjust their wrist position and control the force used in finger flexion actions. This setup not only makes the game challenging and entertaining but also provides a clear measure of progress

in both the therapeutic exercise and the game through a scoring system. Additionally, the game's adaptability allows it to be customized to different skill levels or rehabilitation needs, accommodating a wide range of patient abilities and rehabilitation goals (Figure 23).

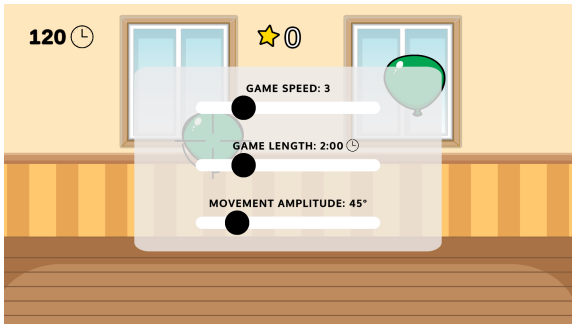

Figure 23. Game "Wrist sniper" settings window

4.4 Game 4: Piano

The game "Piano," specifically developed for hand therapy, emphasizes exercises that involve the flexion and extension of the fingers. During gameplay, players engage in a sequence of finger-bending actions, initiated from a palm-down hand position. This activity involves bending each finger in turn, starting with the thumb and ending with the little finger, and holding the bend for 1-2 seconds before returning the finger to its initial position (Figure 24). Players sit at a table, ensuring their forearm is comfortably rested with their palm facing downward.

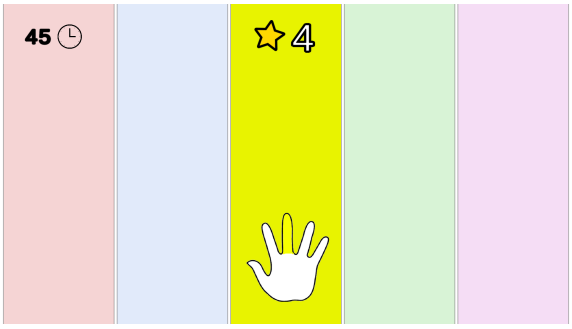

Figure 24. Game "Piano"

The game leverages a visual interface displaying five colored strips, each corresponding to one of the player's fingers. As the game progresses, parts of these strips light up (Figure 25). The objective is to lower the finger matching the lit-up strip as it approaches, requiring precise timing and finger positioning. This interaction is pivotal for successfully completing the game's challenges.

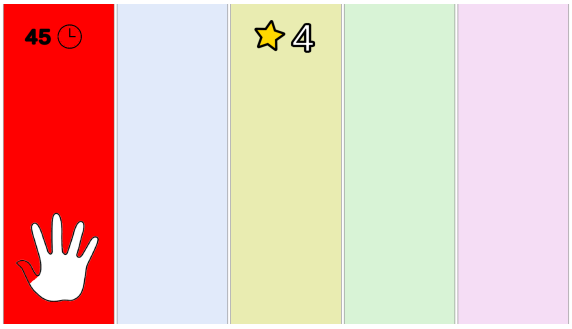

Figure 25. Game "Piano"

The game employs a specific method, DetectTargetFingerBend, to assess the player's accuracy in bending the correct finger at the right moment. The game, designed with a focus on the player's reaction timing and accuracy, contributes significantly to the enhancement of finger coordination and motor skills. The implementation of the game's mechanics, including the finger detection and visual feedback.

The design of "Piano" ensures that players are continuously engaged in therapeutic exercises for their fingers. It combines the challenge and entertainment of a game with the benefits of physical rehabilitation. This approach not only makes therapy sessions enjoyable but also provides measurable progress in both the game and the therapeutic exercises. Furthermore, the game's adaptability allows customization to various skill levels and rehabilitation needs, accommodating a broad spectrum of patient abilities and therapy objectives (Figure 26).

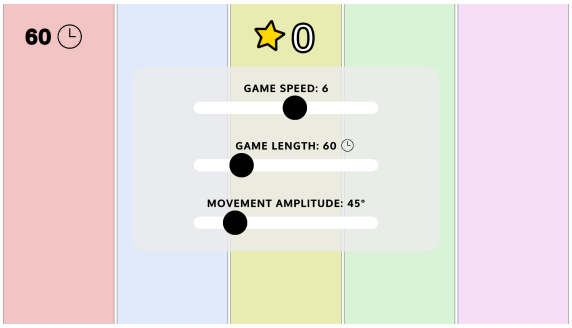

Figure 26. Game "Piano" settings window

5. Economic overview

To conduct a financial assessment of this project, we need to identify the resources required for its development.

Hardware used for the project (provided by the university):

- Leap Motion (200$)

Software used for the project:

- Unity game engine (free personal license)
- JetBrains Rider (free education license)
- GIMP (free open source software)
- Ultraleap Plugin for Unity (free open source software)

Workforce

- Developer hours spent – 156 hours

In summary, the economic assessment of this gamified hand therapy project underscores its cost-efficiency and development effectiveness. Leveraging free or open-source software and university-provided hardware, the project demonstrates a cost-effective approach to developing therapeutic games. The efficient use of technology and developer hours has not only kept initial costs low but also paved the way for long-term sustainability and scalability. Compared to traditional therapy methods, this digital approach offers a potentially more economical alternative, reducing the need for extensive therapist supervision while providing customizable and replicable therapeutic content. The project's economic feasibility, coupled with its innovative approach, positions it favorably for future funding opportunities and broader implementation in the field of rehabilitative therapy.

6. Future Developments and Enhancements

Despite the substantial amount of work already undertaken in the project on gamification of hand therapy using Leap Motion technology, there remain numerous opportunities for further improvement and expansion of this rehabilitation approach. A number of key developments and enhancements are proposed, aimed at deepening the effectiveness of the therapy and expanding its functionality, which could potentially improve patient engagement and outcomes.

Proposed Enhancements

1. Enhanced Motion Tracking: Implementing more sophisticated tracking algorithms to improve the accuracy and sensitivity of motion detection. This enhancement will allow for more detailed feedback for patients, facilitating more precise exercise execution and improved motor skills. It will also enable the expansion of the exercise list with more complex hand and finger movements.
2. Personalization and Individualization: Developing a more flexible personalization system will allow therapists to more effectively tailor exercises and gaming scenarios to the individual needs of each patient. This may include customization of difficulty levels, personal goals, and an adaptive gameplay process that evolves in response to the patient's progress.
3. User Interface and Experience Improvement: To enhance the overall user experience, the application could offer more intuitive navigation, a user-friendly interface, and engaging game designs. This will make the application more accessible to a wider range of patients, including those with limited technological literacy.
4. Integration with Virtual Reality (VR): Combining Leap Motion technology with VR can create a more immersive rehabilitation environment, potentially leading to increased patient motivation, adherence to recommendations, and improved therapeutic outcomes.
5. Tele-rehabilitation Capabilities: Developing tele-rehabilitation features will allow for remote monitoring and guidance by therapists. This approach can provide continuous support and assistance, especially for patients who are unable to regularly visit medical facilities.
6. Expansion of the Exercise Library: Expanding the range of exercises and games in the application will cover a broader spectrum of hand and wrist function impairments, improving the therapy's versatility and applicability at different stages of

rehabilitation.

7. Collaboration with Medical Professionals: Ongoing collaboration with medical professionals, including occupational therapists and physiotherapists, will ensure the clinical relevance and effectiveness of the application. Their input will assist in developing new exercises, gaming scenarios, and assessment tools.

Conclusion

The future developments proposed in the Leap Motion-based hand therapy gamification project represent a significant advancement in enhancing the effectiveness of rehabilitation methods.

7. Summary

In this thesis, the integration of gamification elements into hand therapy was explored, specifically for patients with upper limb dysfunctions, with an emphasis on the development of games to gamify therapy.

The initial plan was to develop a therapeutic tool that merges traditional rehabilitation methods with the engaging elements of game design, aiming to make the sessions more interactive and motivating for patients.

Throughout the research, a series of game-based exercises intended for hand therapy were developed. These games employed Leap Motion technology for accurate tracking of patients' hand movements during the exercises, integrating control into the gaming process.

The produced project can be evaluated further with detailed users studies to assess practical effectiveness of gamified approach. As the base data collecting logic already present in the application, it can be used to collect data from practical exercises. This functionality can enable continuous evaluation and adjustment of therapeutic programs, considering the unique needs of each patient.

Directions for future development and enhancement were identified, including improving game designs, expanding the range of exercises, enhancing systems for evaluating exercise success, dynamically adjusting the difficulty level for individual patients, and investigating additional technologies to improve the therapeutic experience.

To sum up, the conducted work allowed to achieve all the goals set in the beginning of this thesis work. The produced application can pave the way to more immersive and engaging therapeutic routines.

The program code for this project is available here: [26]. This repository hosts the source code and associated materials pertinent to the thesis, focusing on the gamification of hand therapy using Leap Motion technology.

References

[1] Dick F Swaab, Samantha E C Wolff, and Ai-Min Bao. “Sexual differentiation of the human hypothalamus: Relationship to gender identity and sexual orientation”. In: *The Human Hypothalamus - Neuroendocrine Disorders*. Handbook of clinical neurology. Elsevier, 2021, pp. 427–443.

[2] Richard T Scott 3rd et al. “Mitochondrial DNA content is not predictive of reproductive competence in euploid blastocysts”. en. In: *Reprod. Biomed. Online* 41.2 (Aug. 2020), pp. 183–190.

[3] Christian Grefkes and Gereon R Fink. “Recovery from stroke: current concepts and future perspectives”. en. In: *Neurol. Res. Pract.* 2.1 (June 2020), p. 17.

[4] Lewis Ingram et al. “Quantifying upper limb motor impairment in chronic stroke: a physiological profiling approach”. In: *Journal of Applied Physiology* 131.3 (Sept. 1, 2021), pp. 949–965. DOI: 10.1152/japplphysiol.00078.2021.

[5] Catherine Lang et al. “Assessment of upper extremity impairment, function, and activity after stroke: foundations for clinical decision making”. In: *Journal of Hand Therapy* 26.2 (Apr. 2013), pp. 104–115. DOI: 10.1016/j.jht.2012.06.005.

[6] Seedahmed Mahmoud et al. “Occupational Therapy Assessment for Upper Limb Rehabilitation: A Multisensor-Based Approach”. In: *Frontiers in Digital Health* 3 (Dec. 17, 2021). DOI: 10.3389/fdgth.2021.784120.

[7] Keith Lohse et al. “Video Games and Rehabilitation”. In: *Journal of Neurologic Physical Therapy* 37.4 (Dec. 2013), pp. 166–175. DOI: 10.1097/npt.0000000000000017.

[8] Jun Wei Then et al. “Gamification in rehabilitation of metacarpal fracture using cost-effective end-user device: A randomized controlled trial”. In: *Journal of Hand Therapy* 33.2 (Apr. 2020), pp. 235–242. ISSN: 0894-1130. DOI: 10.1016/j.jht.2020.03.029. URL: http://dx.doi.org/10.1016/j.jht.2020.03.029.

[9] Fábio Marcon Alfieri et al. “Gamification in Musculoskeletal Rehabilitation”. In: *Current Reviews in Musculoskeletal Medicine* 15.6 (Oct. 2022), pp. 629–636. ISSN: 1935-9748. DOI: 10.1007/s12178-022-09797-w. URL: http://dx.doi.org/10.1007/s12178-022-09797-w.

[10] Nrupen Bhavsar et al. “Prevalence and predictors of no-shows to physical therapy for musculoskeletal conditions”. In: *PLOS ONE* 16.5 (May 28, 2021), e0251336. DOI: 10.1371/journal.pone.0251336.

[11] Marcel Dijkers and Jeanne Zanca. “Factors Complicating Treatment Sessions in Spinal Cord Injury Rehabilitation: Nature, Frequency, and Consequences”. In: *Archives of Physical Medicine and Rehabilitation* 94.4 (Apr. 2013), S115–S124. DOI: 10.1016/j.apmr.2012.11.047.

[12] Sharon Toh, Pei Chia, and Kenneth Fong. “Effectiveness of home-based upper limb rehabilitation in stroke survivors: A systematic review and meta-analysis”. In: *Frontiers in Neurology* 13 (Sept. 9, 2022). DOI: 10.3389/fneur.2022.964196.

[13] *Leap Motion*. URL: https://www.ultraleap.com/.

[14] Margo Sheerin et al. “Effectiveness of occupational therapy interventions on function and occupational performance among adults with conditions of the hand, wrist, and forearm: A systematic review and meta-analysis”. In: *Australian Occupational Therapy Journal* (Oct. 4, 2023). DOI: 10.1111/1440-1630.12905.

[15] Aleksi Penttilä. *Increasing User Motivation of Neurological Occupational Therapy in Virtual Reality Using Gamification*. 2023.

[16] Mónica Da Silva Cameirão et al. “Virtual reality based rehabilitation speeds up functional recovery of the upper extremities after stroke: A randomized controlled pilot study in the acute phase of stroke using the Rehabilitation Gaming System”. In: *Restorative Neurology and Neuroscience* 29.5 (2011), pp. 287–298. DOI: 10.3233/rnn-2011-0599.

[17] Nauman Shah, Angelo Basteris, and Farshid Amirabdollahian. “Design Parameters in Multimodal Games for Rehabilitation”. In: *Games for Health Journal* 3.1 (Feb. 2014), pp. 13–20. DOI: 10.1089/g4h.2013.0044.

[18] *Tallinn East Central Hospital*. URL: https://www.itk.ee/en.

[19] URL: https://www.neofect.com/us/smart-glove.

[20] URL: https://www.jintronix.com/.

[21] URL: https://www.tyromotion.com/en/products/amadeo.

[22] URL: https://www.mindmaze.com/healthcare/products/mindmotion-go/.

[23] *Unity*. URL: https://unity.com/.

[24] *Unreal Engine*. URL: https://www.unrealengine.com/.

[25] *ULTRALEAP PLUGIN FOR UNITY*. URL: https://developer.leapmotion.com/unity.

[26] *Gamification-of-Hand-Therapy-using-Leap-Motion*. URL: https://github.com/Vlagod/Gamification-of-Hand-Therapy-using-Leap-Motion.

Appendix 1 – Non-Exclusive License for Reproduction and Publication of a Graduation Thesis[1]

I Vladislav Konstantinov

1. Grant Tallinn University of Technology free licence (non-exclusive licence) for my thesis "Gamification of Hand Therapy using Leap Motion", supervised by Yevhen Bondarenko
   1.1. to be reproduced for the purposes of preservation and electronic publication of the graduation thesis, incl. to be entered in the digital collection of the library of Tallinn University of Technology until expiry of the term of copyright;
   1.2. to be published via the web of Tallinn University of Technology, incl. to be entered in the digital collection of the library of Tallinn University of Technology until expiry of the term of copyright.
2. I am aware that the author also retains the rights specified in clause 1 of the non-exclusive licence.
3. I confirm that granting the non-exclusive licence does not infringe other persons' intellectual property rights, the rights arising from the Personal Data Protection Act or rights arising from other legislation.

03.01.2024

[1]The non-exclusive licence is not valid during the validity of access restriction indicated in the student's application for restriction on access to the graduation thesis that has been signed by the school's dean, except in case of the university's right to reproduce the thesis for preservation purposes only. If a graduation thesis is based on the joint creative activity of two or more persons and the co-author(s) has/have not granted, by the set deadline, the student defending his/her graduation thesis consent to reproduce and publish the graduation thesis in compliance with clauses 1.1 and 1.2 of the non-exclusive licence, the non-exclusive license shall not be valid for the period.

Appendix 2 — App Source Code

Listing 1. Ship Controller

```
using System;
using System.Collections;
using System.Collections.Generic;
using Leap;
using Leap.Unity;
using UnityEngine;
public class ShipController : MonoBehaviour
{
    public LeapServiceProvider leapServiceProvider;
    public float speed = 1f;

    private Hand _currentHand;
    private Arm _currentArm;
    [SerializeField] private float _previousWristRotationAngle;
    [SerializeField] private float currentWristRotationAngle;
    [SerializeField] private int _rotationCounter;
    private float _rotationThreshold = 20;
    [SerializeField] private float _cumulativeRotation;


    [SerializeField] private float rotationDifference;

    void Start()
    {
        _rotationCounter = 0;
        _cumulativeRotation = 0f;
    }

    void Update()
    {
        if(ShipGameManager.Instance.isPaused) return;
        Frame currentFrame = leapServiceProvider.CurrentFrame;
        if (currentFrame != null && currentFrame.Hands.Count >
            0)
        {
            _currentHand = currentFrame.Hands[0];
```

```
        _currentArm = _currentHand.Arm;

        if (_previousWristRotationAngle == 0)
        {
            _previousWristRotationAngle =
            Vector2.Angle(_currentHand.PalmPosition,
                _currentArm.WristPosition);
        }
        else
        {
            currentWristRotationAngle =
            Vector2.Angle(_currentHand.PalmPosition,
                _currentArm.WristPosition);
            if (currentWristRotationAngle < 60 &&
                _previousWristRotationAngle > 300)
            {
                rotationDifference =
                360 - _previousWristRotationAngle +
                    currentWristRotationAngle;
            }else if (currentWristRotationAngle > 300 &&
                _previousWristRotationAngle < 60)
            {
                rotationDifference =
                _previousWristRotationAngle + 360 -
                    currentWristRotationAngle;
            }
            else
            {
                rotationDifference =
                currentWristRotationAngle -
                    _previousWristRotationAngle;
            }

            _previousWristRotationAngle =
                currentWristRotationAngle;
        }

        var newPos = transform.position;
        newPos.x -= 1 * speed * rotationDifference * 1.5f;
        transform.position = newPos;
    }
```

```
}


private void OnTriggerEnter(Collider other)
{
    if (other.CompareTag("obstacle"))
    {
        ShipGameManager.Instance.OnShipCollision();
        other.gameObject.SetActive(false);

    }else if (other.CompareTag("bonus"))
    {
        ShipGameManager.Instance.AddScore();
    }
    Destroy(other.gameObject);
}


void HandRotation()
{
    Frame currentFrame = leapServiceProvider.CurrentFrame;
    if (currentFrame != null && currentFrame.Hands.Count >
       0)
    {
        _currentHand = currentFrame.Hands[0];
        _currentArm = _currentHand.Arm;

        if (_previousWristRotationAngle == 0)
        {
            _previousWristRotationAngle = Vector2.Angle(
               _currentHand.PalmPosition, _currentArm.
               WristPosition);
        }
        else
        {
            currentWristRotationAngle = Vector2.Angle(
               _currentHand.PalmPosition, _currentArm.
               WristPosition);
```

```
                rotationDifference = currentWristRotationAngle -
                    _previousWristRotationAngle;

                if (Mathf.Abs(rotationDifference) <=
                    _rotationThreshold)
                {
                    if (Mathf.Sign(rotationDifference) != Mathf.
                        Sign(_cumulativeRotation))
                    {
                        _cumulativeRotation = 0;
                    }

                    _cumulativeRotation += rotationDifference;

                    if (_cumulativeRotation >= 360f)
                    {
                        _rotationCounter++;
                        _cumulativeRotation %= 360f;
                    }
                }

                _previousWristRotationAngle =
                    currentWristRotationAngle;
            }

        }
    }

    void MoveShip()
    {
        float moveDelta = 1f;
        var newPos = transform.position;
        newPos.x += moveDelta * speed * (_cumulativeRotation /
            360);
        transform.position = newPos;
    }

}
```

Listing 2. Plane Controller

```
using System;
using System.Collections;
using System.Collections.Generic;
using System.Numerics;
using Leap;
using Leap.Unity;
using UnityEngine;
using UnityEngine.Analytics;
using UnityEngine.Events;
using Quaternion = UnityEngine.Quaternion;
using Vector2 = UnityEngine.Vector2;
using Vector3 = UnityEngine.Vector3;

public class PlaneController : MonoBehaviour
{

    public float speed = 9f;
    public static PlaneController Instance;
    private Vector3 palmStartPos;
    private Vector3 startRotation;

    public UnityEvent onPlaneCollision = new UnityEvent();

    private void Awake()
    {
        Instance = this;
    }

    void Start()
    {

        startRotation = transform.rotation.eulerAngles;
    }

    void Update()
    {
        PlaneTilt();
        PlaneMovement();
    }
    public LeapServiceProvider leapServiceProvider;
```

```
void OnTriggerEnter(Collider other) {
    onPlaneCollision.Invoke();
    GetComponent<Animator>().SetTrigger("Collision");
}

public void PlaneTilt()
{
    Frame frame = leapServiceProvider.CurrentFrame;

    foreach (Hand hand in frame.Hands)
    {
        if (hand.IsRight)
        {
            Vector3 normal = hand.PalmNormal;
            if (palmStartPos == Vector3.zero)
            {
                palmStartPos = normal;
            }

            Vector3 projectedNormal = Vector3.ProjectOnPlane
               (normal, Vector3.up);
            float yaw = -Vector3.SignedAngle(Vector3.forward
               , projectedNormal, Vector3.up);
            float rotateSpeed = 1f;


            var rotateAngleY = (new Vector3(0, yaw, 0) *
               Time.deltaTime * rotateSpeed).y;
            var currentLocalEulerY = transform.
               localEulerAngles.y;



            float angleDifference = Mathf.Abs(
               currentLocalEulerY - yaw);

            if (rotateAngleY > 0 && angleDifference < 20)
            {
                Debug.Log("Rotate␣right␣Limit␣Reached");
            }
            else if (rotateAngleY < 0 && angleDifference >
```

```
                280)
            {
                Debug.Log("Rotate left Limit Reached");
            }
            else
            {
                RotatePlane(new Vector3(0, rotateAngleY, 0))
                    ;
            }
        }

    }
}

void RotatePlane(Vector3 rotateAngle)
{
    transform.Rotate(rotateAngle, Space.Self);
}
void PlaneMovement()
{
    if (transform.rotation.eulerAngles.y > startRotation.y)
    {
        if (transform.position.x < -90)
        {
            return;
        }

        var shiftX = transform.position.x - ((transform.
            rotation.eulerAngles.y - startRotation.y) ) *
            Time.deltaTime * speed;
        transform.position = new Vector3(shiftX, transform.
            position.y , transform.position.z);
    }else if(transform.rotation.eulerAngles.y <
        startRotation.y)

    {
        if (transform.position.x > 80)
        {
            return;
        }
        var shiftX = transform.position.x + ((startRotation.
```

```
                y - transform.rotation.eulerAngles.y)) * Time.
                    deltaTime * speed;
                transform.position = new Vector3(shiftX, transform.
                    position.y, transform.position.z);
            }

        }
}
```

Listing 3. Aim Controller

```
using System;
using System.Collections;
using System.Collections.Generic;
using Leap;
using Leap.Unity;
using UnityEngine;
using WristSniper.Scripts;

public class AimController : MonoBehaviour
{

    public LeapServiceProvider leapServiceProvider;
    public float speed = 1f;

    public float rightMovementScale = 1;
    public float leftMovementScale = 1;
    public float upMovementScale = 1;
    public float downMovementScale = 1;
    void Update () {
        Frame frame = leapServiceProvider.CurrentFrame;

        if (frame.Hands.Count > 0) {
            Vector3 wristPosition = frame.Hands[0].WristPosition
                ;


            Hand hand = frame.Hands[0];
```

```
Vector3 palmNormal = hand.PalmNormal;
Vector3 palmDirection = hand.Direction;
Quaternion wristRotation = Quaternion.LookRotation(
    palmDirection, -palmNormal);


Quaternion upperArmRotation = hand.Arm.Basis.
    rotation;


Vector3 wristForward = wristRotation * Vector3.
    forward;
Vector3 upperArmForward = upperArmRotation * Vector3
    .forward;


if (wristForward.x > 0)
{
    wristForward *= rightMovementScale;
}
else
{
    wristForward *= leftMovementScale;
}

if (upperArmForward.y > 0)
{
    upperArmForward *= upMovementScale;
}
else
{
    upperArmForward *= downMovementScale;
}
Vector3 movementDirection = wristForward +
    upperArmForward;
movementDirection.Normalize();
movementDirection.z = 0;

transform.position += movementDirection * Time.
    deltaTime * speed;
```

```
        }
    }

    private void OnTriggerStay2D(Collider2D other)
    {
        Frame frame = leapServiceProvider.CurrentFrame;

        if (frame.Hands.Count > 0)
        {

            Hand hand = frame.Hands[0];
            if (hand.GrabStrength > 0.95f)
            {
                Destroy(other.gameObject);
                WristSniperGameManager.AddPoint.Invoke();

            }
        }
    }


}
```

Listing 4. Piano game Finger bend detection

```
using System.Collections;
using System.Collections.Generic;
using Leap;
using Leap.Unity;
using UnityEngine;
using Image = UnityEngine.UIElements.Image;

public class TapFieldBehaviour : MonoBehaviour
{

    [SerializeField] private BoxCollider2D boxCollider2D;
    [SerializeField] private RectTransform rectTransform;
    [SerializeField] private UnityEngine.UI.Image image;
    private const int SECTION_SIZE = 100;
```

```
private float speed = 25;

private Controller leapController;

private int targetFingerIndex;

public void Init(int size, int fingerIndex)
{
    var sizeDelta = rectTransform.sizeDelta;
    sizeDelta.y *= size;
    rectTransform.sizeDelta = sizeDelta;
    this.targetFingerIndex = fingerIndex;
}

void Start()
{
    leapController = FindObjectOfType<LeapServiceProvider>()
        .GetLeapController();
}


void Update()
{
    Vector2 position = transform.position;
    position.y -= speed * Time.deltaTime;
    transform.position = position;


    DetectTargetFingerBend();
}

private bool fingerLowered = false;
private float loweredThreshold = 0.05f;
void DetectTargetFingerBend()
{
    Frame frame = leapController.Frame();
    if (frame.Hands.Count < 1)
    {
        return;
    }
```

```
        Hand hand = frame.Hands[0];
        Finger indexFinger = hand.Fingers[targetFingerIndex];


        if (indexFinger.TipPosition.y < hand.PalmPosition.y -
            loweredThreshold)
        {
            if (!fingerLowered)
            {
                fingerLowered = true;
                image.color = Color.red;

            }
        }
        else
        {
            fingerLowered = false;
            image.color = Color.white;
        }
    }

}
```