*Mini-Project Report On*

# Virtual Sliders (An Air Hockey Game using Hand Detection)

*Submitted in partial fulfillment of the requirements for the award of the degree of*

## Bachelor of Technology

*in*

***Computer Science & Engineering***


**By**

**Krishnadas Balachandran (U2003121)**
**Joel Johnson (U2003104)**
**Jonathan Antony (U2003108)**
**Joel Joseph Justin (U2003105)**

**Under the guidance of**
**Ms. Sangeetha Jamal**

RSET
RAJAGIRI SCHOOL OF ENGINEERING & TECHNOLOGY (AUTONOMOUS)

**Department of Computer Science & Engineering**
**Rajagiri School of Engineering and Technology (Autonomous)**
**Rajagiri Valley, Kakkanad, Kochi, 682039**


**July 2023**

**DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING**
**RAJAGIRI SCHOOL OF ENGINEERING AND TECHNOLOGY**
**(AUTONOMOUS)**
**RAJAGIRI VALLEY, KAKKANAD, KOCHI, 682039**


# CERTIFICATE

*This is to certify that the mini-project report entitled* ***"Virtual Sliders (An Air Hockey Game using Hand Detection)"*** *is a bonafide work done by* ***Mr. Joel Johnson (U2003104), Mr. Joel Joseph Justin (U2003105), Mr. Jonathan Antony (U2003108), Mr. Krishnadas Balachandran (U2003121)****, submitted to the APJ Abdul Kalam Technological University in partial fulfillment of the requirements for the award of the degree of Bachelor of Technology (B. Tech.) in Computer Science and Engineering during the academic year 2022-2023.*

| | | |
|---|---|---|
| **Dr. Preetha K. G.** | **Mr. Uday Babu P.** | **Ms. Sangeetha Jamal** |
| Head of Department | Mini-Project Coordinator | Mini-Project Guide |
| Dept. of CSE | Asst. Professor | Asst. Professor |
| RSET | Dept. of CSE | Dept. of CSE |
| | RSET | RSET |

## ACKNOWLEDGEMENTS

We wish to express our sincere gratitude towards **Dr. P. S. Sreejith**, Principal of RSET, and **Dr. Preetha K. G.**, Head of Department of Computer Science and Engineering for providing us with the opportunity to undertake our mini-project, "Virtual Sliders".

We are highly indebted to our mini-project coordinators, **Mr. Uday Babu**, Assistant Professor, Department of Computer Science and Engineering, and **Ms. Tripti C**, Assistant Professor, Department of Computer Science and Engineering for their valuable support.

It is indeed our pleasure and a moment of satisfaction for us to express our sincere gratitude to our mini-project guide **Ms. Sangeetha Jamal**, Assistant Professor, Department of Computer Science and Engineering, for her patience and all the priceless advice and wisdom she has shared with us.

Last but not the least, we would like to express our sincere gratitude towards all other teachers and friends for their continuous support and constructive ideas.

**Joel Johnson**

**Joel Joseph Justin**

**Krishnadas Balachandran**

**Jonathan Antony**

## ABSTRACT

This project investigates the implementation of computer vision techniques to enhance the gameplay experience of air hockey through hand detection. Traditional air hockey games require a physical table and equipment, limiting accessibility and socialization. To address this, we propose a computer vision-based solution utilizing real-time hand detection algorithms to track and interpret players' hand movements.

The proposed system empowers players to enjoy the game without the need for physical infrastructure, making it cost-effective and dynamic. Additionally, gesture-based gameplay mechanics, such as power-ups and additional scoring opportunities, are introduced to enhance player immersion and interaction.

Moreover, this project explores the therapeutic potential of the hand detection-based virtual air hockey game, particularly in physiotherapy and rehabilitation settings. The interactive nature of the game offers a low-impact form of physical exercise and improves hand-eye coordination, making it valuable for patients' recovery. The real-time hand tracking capabilities enable progress monitoring, and the database system stores game history and relevant statistics, providing valuable data for tracking patient improvement.

By combining entertainment with therapeutic benefits, this project contributes to computer vision advancements and medical rehabilitation practices, showcasing the potential of leveraging real-time hand tracking to enhance traditional tabletop games.

# Contents

# List of Figures

# Chapter 1

# Introduction

## 1.1 Background

We designed a captivating and user-friendly virtual air hockey game that revolutionizes the traditional gameplay experience through hand detection controls and multiplayer functionality. Our primary aim was to create an immersive and accessible game, eliminating the need for physical mallets and providing an engaging platform for players to compete against each other.

In this unique rendition of air hockey, players control their mallets using hand gestures detected by our advanced computer vision system. By simply moving their hands in the designated play area, the players can swiftly maneuver the mallets across the digital air hockey table, making the game intuitive and enjoyable for all. To enhance the competitive spirit, our game accommodates multiplayer mode, allowing two players to challenge each other in real-time. With seamless synchronization, each player sees their individual perspective of the air hockey table, creating an authentic and immersive gaming experience.

Developing this game presented intriguing technical challenges, particularly in the realm of hand detection. We aimed for precise and responsive hand tracking to ensure a smooth and fluid gameplay experience. Accurately capturing hand movements and translating them into mallet actions required intricate attention to detail and efficient design. Throughout the development process, we embraced flexibility and innovation. While our initial vision included 3-dimensional graphics, we opted for 2-dimensional graphics to ensure a seamless gaming experience without compromising performance or responsiveness.

Beyond the realm of gaming and entertainment, our virtual air hockey game also serves a purpose in the field of physiotherapy. We have designed the game with the vision of

incorporating it as an engaging and effective tool for rehabilitation and physical therapy. Physiotherapy professionals have recognized the potential of virtual reality-based games in motivating patients during their recovery journeys. Our hand detection controls offer a gentle and low-impact method for patients to exercise and improve their range of motion, hand-eye coordination, and fine motor skills. By incorporating the air hockey game into therapy sessions, patients can participate in enjoyable activities that aid in their recovery while tracking their progress through the game's interactive feedback.

In summary, our virtual air hockey game, powered by hand detection technology and multiplayer capabilities, offers a delightful and interactive gaming experience. The combination of intuitive hand controls, realistic physics, and visually appealing graphics brings a new dimension to the timeless classic of air hockey. We are proud to present a working prototype of this innovative game and look forward to further refining and expanding its features in the future.

### 1.2 Existing System

The existing system of air hockey involves a physical rectangular table with a smooth, low-friction surface where two players control mallets to strike a lightweight puck. The objective is to score goals by hitting the puck into the opponent's goal while defending their own. Challenges with the traditional setup include the need for dedicated space, cost, limited mobility, and physical fatigue.

In addition to the physical version, the popularity of air hockey has extended to the virtual realm, with the advent of online air hockey games. These online versions allow players to engage in air hockey matches over the internet, connecting enthusiasts from all around the world. For instance, "VirtualHockeyPro" is an online air hockey game that exemplifies the appeal of this digital adaptation. Players can visit the game's website, create an account, and challenge opponents in real-time matches. The virtual version faithfully replicates the physics and mechanics of traditional air hockey while providing the convenience of playing from the comfort of one's own home.

The existing online version of air hockey offers convenience and accessibility, allowing players to engage in matches from anywhere with traditional input methods like mouse and keyboard controls. However, it falls short when compared to a hand detection model in terms of interaction realism, physicality, and engagement. Hand detection technology provides a more immersive and intuitive gameplay experience, closely resembling the real-time hand movements used to control the mallets. The precise hand tracking of a hand detection model also offers more accurate and responsive control. Additionally, the physicality of moving one's hands adds an extra layer of excitement and involvement. Moreover, hand detection models have the potential for therapeutic benefits, promoting physical exercise and serving as a tool in physiotherapy or rehabilitation settings, which the online version may not achieve with traditional controls. Overall, hand detection models provide a more captivating and interactive air hockey gaming experience.

To address these limitations and offer an innovative approach, our project focuses on developing a virtual air hockey game using hand detection technology. This game eliminates the need for a physical table and mallets, making it more accessible and enjoyable for players. By leveraging computer vision and incorporating multiplayer features, we aim to create a captivating and interactive gaming experience for users while exploring new possibilities in air hockey gameplay.

### 1.3 Problem Statement

The traditional physical setup of air hockey presents various limitations, including the need for dedicated space, cost, and limited mobility. Additionally, the conventional online version of air hockey lacks the realism, engagement, and physicality that players experience when controlling mallets with their hands in a real-world scenario. To address these challenges and create a more accessible, immersive, and engaging air hockey experience, this project aims to develop a virtual air hockey game utilizing hand detection technology. The goal is to provide players with an intuitive and realistic gameplay environment, allowing them to control the mallets through hand gestures and interact with the game in a manner that closely mimics the physical interactions of traditional air hockey. By leveraging hand detection technology, we seek to revolutionize the way air hockey is played, offering an innovative and entertaining gaming experience that extends beyond

the limitations of traditional setups and online versions.

## 1.4 Objectives

- **Implement Hand Detection Controls**: Develop and integrate a robust hand detection system that accurately tracks and interprets players' hand gestures to control the mallets in the virtual air hockey game. The objective is to create an intuitive and responsive control mechanism, providing a seamless and immersive gaming experience.

- **Realistic Physics Engine**: Design and implement a physics engine that accurately simulates the dynamics of air hockey gameplay. The engine should account for factors like puck speed, momentum, and collisions with the mallets and walls, ensuring a lifelike and challenging gaming environment.

- **Multiplayer Functionality**: Introduce multiplayer features, allowing players to compete against each other in real-time matches. The objective is to create a synchronized gameplay experience where players can see their individual perspectives of the virtual air hockey table, fostering friendly competition and social interaction.

- **User Interface and Graphics**: Develop an appealing and user-friendly graphics interface, combining retro-style aesthetics with modern elements. The objective is to enhance the visual appeal of the game, creating a nostalgic yet contemporary ambiance that engages players and enhances their overall gaming experience.

- **Database System for Game History and Scores**: Implement a database system to store game history, player scores, and other relevant statistics. The objective is to maintain a record of past matches and player achievements, enabling players to track their progress and encouraging healthy competition among participants.

- **Therapeutic Applications**: Explore the potential of utilizing the hand detection-based virtual air hockey game in physiotherapy and rehabilitation settings. The objective is to assess its effectiveness in promoting physical exercise, improving hand-eye coordination, and supporting therapeutic goals for patients undergoing rehabilitation.

## 1.5 Scope

The scope of this project encompasses the development and implementation of a virtual air hockey game that utilizes hand detection technology for intuitive mallet control. Multiplayer functionality will be integrated, enabling real-time matches between players.

The user interface and graphics will be carefully crafted to provide an appealing and user-friendly experience, combining retro-style aesthetics with modern elements. Additionally, the project will explore the potential therapeutic applications of the game, especially in physiotherapy and rehabilitation settings, aiming to promote physical exercise and improve hand-eye coordination in patients.

To enhance player engagement, a database system will be implemented to store game history, player scores, and other relevant statistics. This system will enable players to track their progress, review past matches, and participate in healthy competition. As the project progresses, extensive playtesting sessions will be conducted to gather user feedback, allowing for iterative improvements to enhance the gameplay experience and hand detection accuracy. Performance optimization will be prioritized to ensure smooth gameplay, responsive controls, and minimal latency during multiplayer matches.

The project's scope will be limited to the development and implementation of the virtual air hockey game, the hand detection technology integration, the physics engine, multiplayer features, user interface, and database system. While therapeutic applications will be explored, the project will not cover in-depth clinical studies or extensive medical testing. The focus will primarily be on delivering a high-quality, enjoyable, and innovative virtual air hockey gaming experience that captures the excitement of the traditional game while leveraging modern technology for enhanced accessibility and immersion.

# Chapter 2

# Literature Review

## 2.1 AirHockeyBot: A Robot Vision-Based Player for Air Hockey by Stefanie Tellex, MIT's Computer Science and Artificial Intelligence Laboratory (CSAIL), 2014

The primary goal of this project was to create an autonomous robot player capable of engaging in competitive air hockey matches against human opponents using sophisticated computer vision techniques.

To achieve this, the AirHockeyBot was equipped with a state-of-the-art computer vision system that utilized cameras to capture real-time visual data of the air hockey table. Advanced algorithms were developed to process this visual input and accurately detect the position and trajectory of the puck as well as track the hand movements of the human player. The seamless integration of these computer vision algorithms allowed the robot player to perceive the dynamic gameplay environment with precision. The robotics and control aspect of the project played a crucial role in enabling the robot player to interact with the air hockey table effectively. Based on the visual information acquired from the cameras, the robot autonomously controlled its mallet movements and responded in real-time to the puck's trajectory, exhibiting swift and agile gameplay.

To demonstrate adaptability and intelligent decision-making, the AirHockeyBot employed sophisticated algorithms that analyzed the gameplay situation, assessed potential strategies, and made calculated decisions. The robot player aimed to optimize its shots and defensive maneuvers based on the human opponent's actions, adapting its gameplay style dynamically to ensure a competitive and challenging match. One of the key strengths of the AirHockeyBot project was its focus on real-time interaction. The robot

was designed to process information rapidly and make instant decisions, mirroring the responsiveness and speed of human gameplay. This aspect contributed to creating a compelling and immersive experience for both human players and spectators.

The research conducted in the "AirHockeyBot: A Robot Vision-Based Player for Air Hockey" project not only showcased the capabilities of robotics and computer vision technologies but also provided valuable insights into the integration of these disciplines in real-world interactive scenarios. The project's findings and achievements contributed significantly to the fields of artificial intelligence, computer vision, and robotics, offering exciting prospects for the future development of interactive and intelligent robotic systems in gaming and beyond.

One potential drawback of the "AirHockeyBot: A Robot Vision-Based Player for Air Hockey" model is its high complexity and cost. Implementing advanced computer vision algorithms and robotics systems to create an autonomous air hockey player can be resource-intensive and expensive. The need for specialized hardware, cameras, and computational power for real-time processing can result in a significant financial investment. Additionally, the high complexity of the system may lead to increased maintenance and calibration requirements. Ensuring that the computer vision algorithms accurately detect the puck's position and the human player's hand movements in real-time can be challenging. Any inaccuracies or calibration issues could impact the robot's gameplay performance, requiring ongoing adjustments and fine-tuning. Furthermore, the reliance on sophisticated algorithms and robotics introduces a higher risk of technical failures or malfunctions during gameplay. If the computer vision system encounters difficulties in accurately tracking the puck or the human player's hand, the robot's performance may suffer, affecting the overall gaming experience and potentially disrupting the match.

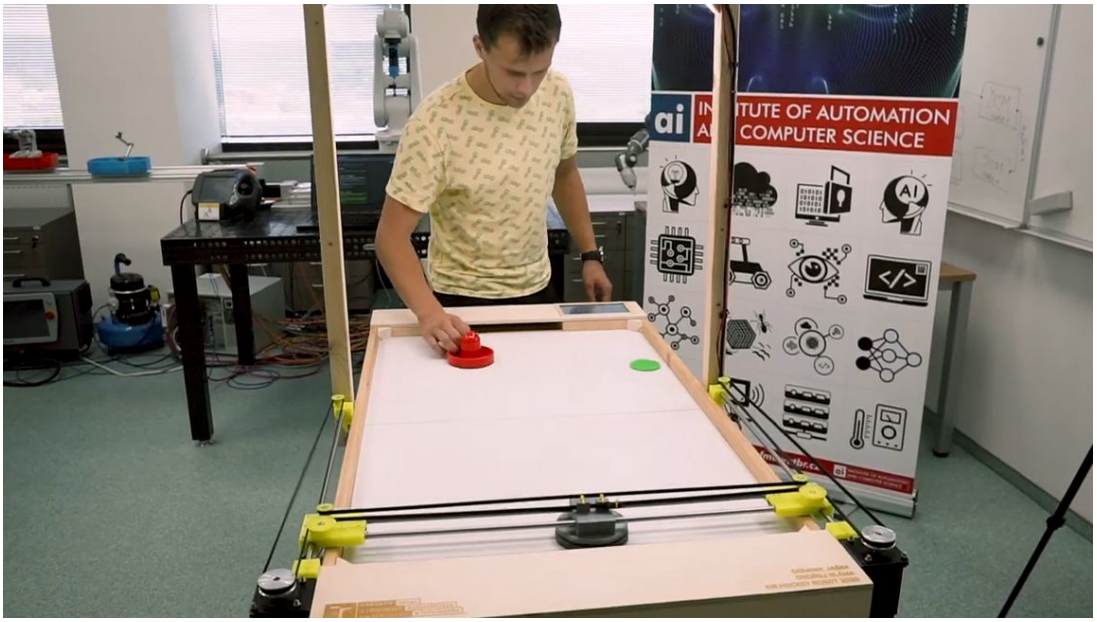



Figure 2.1: Prototype model of the AirHockeyBot

## 2.2 Virtual Air Hockey: A Computer Vision-Based Approach to Tabletop Gaming by Jin Joo Lee and Seungmoon Choi from the Korea Advanced Institute of Science and Technology (KAIST), 2017

"Virtual Air Hockey: A Computer Vision-Based Approach to Tabletop Gaming" is a fascinating research project conducted by Jin Joo Lee and Seungmoon Choi from the Korea Advanced Institute of Science and Technology (KAIST). The primary aim of this project was to revolutionize the traditional tabletop air hockey gaming experience by integrating cutting-edge computer vision technology.

In this innovative approach, the researchers sought to eliminate the need for physical mallets and puck, making the game more accessible and intuitive for players. To achieve this, they employed advanced computer vision algorithms to track and analyze the movements of players' hands in real-time. By detecting the positions of the players' hands on the tabletop, the system interpreted their gestures as virtual mallets, replicating the physical interactions of the game in a virtual environment. A key aspect of the "Virtual Air Hockey" project was the accurate simulation of the virtual puck and mallets on the tabletop. Using computer-generated graphics, the system dynamically visualized the movements of the virtual puck and the players' hand-controlled mallets. This real-time

gameplay simulation ensured a seamless and responsive experience, enabling players to fully immerse themselves in the virtual air hockey match.

The project also emphasized multiplayer functionality, allowing two players to engage in real-time matches. This feature replicated the social and competitive aspects of traditional air hockey, enhancing the gaming experience for friends and family members playing together. Moreover, the researchers explored the potential of gesture-based controls to add depth to the gameplay. By performing specific hand gestures, players could execute various maneuvers, such as blocking, striking, and defending, introducing a novel and interactive element to the game.

The "Virtual Air Hockey: A Computer Vision-Based Approach to Tabletop Gaming" model has some drawbacks to consider. The system's hand tracking accuracy may occasionally suffer from inaccuracies, impacting gameplay. Latency issues could also be noticeable during real-time processing, affecting responsiveness. Limited predefined hand gestures may restrict players' strategic options. High hardware and processing requirements could limit portability and increase costs. The virtual gameplay might lack the physicality of traditional air hockey, potentially affecting player engagement. Learning gesture-based controls could present a learning curve for some players, reducing accessibility. Dependence on computer vision technology introduces technical dependencies and maintenance considerations. Lastly, players must interact within a specific area due to the tabletop constraint, potentially limiting freedom of movement. Addressing these challenges is essential to enhance the system's performance and overall gaming experience.

## 2.3 Air Hockey with Haptic Feedback by Carnegie Mellon University on the International Conference on Virtual Reality and Human-Computer Interaction, 2010

"Air Hockey with Haptic Feedback" is an intriguing research project conducted by Carnegie Mellon University, delving into the realm of immersive gaming experiences. The primary focus of this project is to enrich the traditional air hockey gameplay by integrating haptic feedback technology. Haptic feedback involves the use of touch-based sensations or vibrations to simulate interactions with virtual objects, allowing players to feel the impact and physicality of the game in real-time. For this purpose, haptic devices, such as force-feedback controllers or haptic gloves, are incorporated into the mallets used to strike the virtual puck. As players engage in gameplay, the haptic devices deliver real-time feedback that mimics the sensations of puck deflections, striking the puck with varying force, and collisions on the virtual air hockey table. This not only creates a more immersive gaming experience but also elevates the sense of realism, presence, and engagement, as players can perceive the dynamics of the game through touch. The project emphasizes customization options, allowing players to adjust the intensity and type of haptic sensations according to their preferences, further enhancing the personalization of the gaming experience. By exploring the potential of haptic feedback in air hockey, the researchers seek to push the boundaries of traditional tabletop gaming and provide players with an exciting and captivating tactile journey into the world of virtual air hockey.

### Comparison

- **Technology Focus:** AirHockeyBot emphasizes autonomous robotics and reinforcement learning, Virtual Air Hockey emphasizes computer vision for hand detection, and Air Hockey with Haptic Feedback focuses on haptic technology for touch-based feedback.
- **Gameplay Experience**: AirHockeyBot offers competitive matches against a robotic player, Virtual Air Hockey provides virtual gameplay without physical mallets, and Air Hockey with Haptic Feedback enriches gameplay with tactile sensations.
- **Interaction Method**: AirHockeyBot does not involve direct player interaction,

Virtual Air Hockey relies on hand detection for control, and Air Hockey with Haptic Feedback integrates touch-based interaction through haptic devices.

- **Immersion Level**: AirHockeyBot relies on strategic decision-making algorithms, Virtual Air Hockey offers virtual simulation, and Air Hockey with Haptic Feedback enhances immersion through tactile sensations.

- **Realism and Adaptability**: AirHockeyBot aims for realism with reinforcement learning, Virtual Air Hockey adapts gameplay through computer vision, and Air Hockey with Haptic Feedback adds realism through touch feedback.

In developing our air hockey game using hand detection, we drew inspiration from three of these influential models. Firstly, taking cues from "Virtual Air Hockey", we integrated hand detection technology using computer vision algorithms. This allowed players to control their virtual mallets by simply moving their hands over the air hockey table, eliminating the need for physical mallets. This hand detection system ensured smooth and responsive interactions between players and the virtual environment, enhancing the intuitive nature of the game. Innovatively, instead of haptic feedback, we incorporated sound feedback in the form of foley effects. By synchronizing sound effects with virtual puck movements, mallet strikes, and interactions with the table, players experienced an immersive auditory dimension that further heightened their engagement with the game.

Combining elements from all three models, our air hockey game provides a seamless blend of advanced technology and interactive gameplay. Players can enjoy a realistic and immersive experience through the integration of computer vision technology, responsive hand detection controls, strategic AI opponents, and captivating sound feedback. This amalgamation of ideas from influential models has elevated the air hockey gaming experience, creating a truly unique and enjoyable game for players of all skill levels.

# Chapter 3

# System Analysis

## 3.1 Expected System Requirements

The system of user which could be a Personal Desktop computer or a Laptop is expected to have the following features:

- Windows 10 or 11
- Basic Webcam for hand Detection.
- A storage space of approximate 750 MB for the app.
- A minimum Ram size of 6GB is required in the device.
- An Intel(R) Core(TM) i5-10210U CPU or AMD Ryzen equivalent

## 3.2 Feasibility Analysis

### 3.2.1 Technical Feasibility

The project is technically feasible since majority of the population are in possession of Laptops or Desktops. The app only requires moderate requirements to run on a computer .

### 3.2.2 Operational Feasibility

The operations are built in a simple and easy to use manner for geriatric people. Installation of the app is the only prerequisite operation to be done.

### 3.2.3 Economic Feasibility

The development of the app is zero budget as it was built using free resources.

## 3.3 Hardware Requirements

The following are the system requirements to develop the Unify App.

- Processor: Intel Core i5 or AMD equivalent
- Hard Disk: Minimum 75GB
- RAM: Minimum 8GB

## 3.4 Software Requirements

The following are the softwares used in the development of the app.

Operating System: Windows or Linux

### 3.4.1 Pycharm

Pycharm is an IDE developed by JetBrains used for programming in Python. It provides code analysis, a graphical debugger, an integrated unit tester, integration with version control systems, and supports web development with Django.

### 3.4.2 OpenCV

OpenCV is a library of programming functions mainly for real-time computer vision. The library has more than 2500 optimized algorithms, which includes a comprehensive set of both classic and state-of-the-art computer vision and machine learning algorithms. These algorithms can be used to detect and recognize faces, identify objects, classify human actions in videos, track camera movements, track moving objects etc.

### 3.4.3 MediaPipe

MediaPipe offers a suite of libraries that can be used for various Artificial intelligence or Machine Learning applications. In this project, we are using the Hand Landmarker model to identify the hands, and track it's position.

# Chapter 4

# Methodology

## 4.1 Proposed Method

- Develop a desktop application for playing Air Hockey using hand detection.
- We plan to build hand detection algorithm which likely uses techniques such as image processing, feature extraction, and machine learning to identify and locate hands in the input images or video frames. These techniques can include methods like background subtraction, contour detection, and tracking of key points (such as skeleton points) to determine the position and orientation of hands.
- It takes the player's name as input and utilizes his/her camera to access hand coordinates to play game. The application gives a simulated air hockey game as output and also saves the score,timing of match played
- The application's key features include allowing users to register with a unique username, discover other players online, play a multiplayer air hockey game using hand detection for bat movement, and saves game scores with match history for players to view and track their performance.

### 4.1.1 Hand Detection

In our proposed architecture for Virtual Air Hockey hand detection module we leverage the Mediapipe module, an open-source project developed by Google, to perform accurate hand tracking. It utilizes Skeleton points, also referred to as key points or landmarks, are essential anatomical locations on the hand that play a pivotal role in hand detection and gesture recognition algorithms. These points are typically detected using sophisticated computer vision techniques, such as machine learning-based models or deep neural networks, which are trained on extensive data sets of hand images.

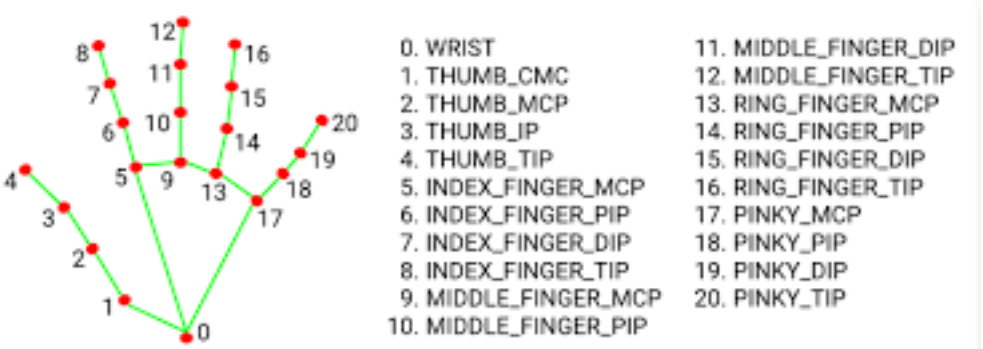



Figure 4.1: Using skeleton points for mapping hand coordinates

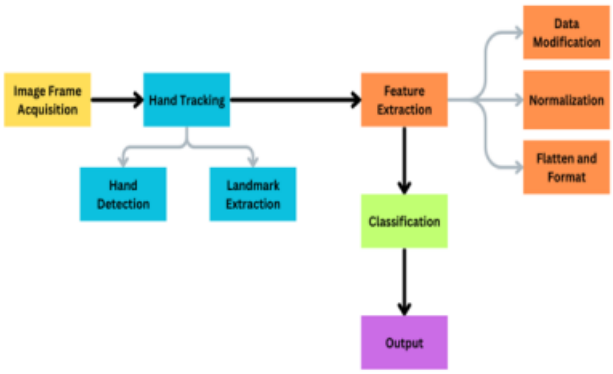



Figure 4.2: Hand detection flowchart

The key skeleton points include the finger tips, which represent the outermost points of each finger and thumb, the palm center, which signifies the geometric center of the palm, and the wrist, denoting the base of the hand. By accurately detecting and tracking these skeleton points, the algorithm can precisely analyze hand movements and gestures.

The finger tips' position enables the system to interpret hand movements, controlling the movement of the virtual air hockey bat. The palm center's tracking provides information about the hand's overall position and rotation, while the wrist's movement distinguishes hand gestures from other body movements, ensuring more accurate hand detection.

### 4.1.2 Game processing

In our multiplayer virtual air hockey game, we implemented a centralized game processing logic to optimize gameplay and reduce the impact of network latency. The server computer handles all game calculations, including ball movement, bat positions, and collision detection. This way, the time-consuming computations are performed on a single powerful machine, allowing us to recover time lost due to network latency during communication with clients.

The game logic is tightly integrated with the exchange of coordinates between the server and clients. Each client continuously sends its hand coordinates to the server, representing the player's movement in the game. The server processes these coordinates, determines the updated positions of the bats, and recalculates the ball's movement. The new positions are then communicated back to the clients, updating their view of the game.

By centralizing game logic on the server, we significantly improved efficiency compared to distributing the calculations across multiple computers. Let's consider an example: if the game logic involves 1000 computations for each frame, and network latency adds an additional 100 ms delay to each data exchange between client and server, the total time taken for one complete game loop on a server-client setup would be:

Time per loop with centralized game logic: 1000 computations * 1 ms + 100 ms latency = 1100 ms Time per loop with distributed game logic: 1000 computations on server + 1000 computations on client + 100 ms latency = 2200 ms

As we can see, centralizing the game logic on the server results in a significant reduction in processing time, making the gameplay smoother and more responsive.

Regarding the game mechanics, the movement of the ball is determined by incrementing its coordinates along the x or y axis based on its speed variable. For example, if the ball's speed in the x direction is positive, its x-coordinate is incremented to move it towards the right. When the ball collides with a surface, such as a wall or bat, the speed variable is reversed to denote a bounce, and the ball's coordinates are updated accordingly to reverse its direction.

This bouncing behavior is calculated using simple physics principles. When the ball hits a surface, the direction of its movement is changed, while the speed remains constant. This approach ensures a realistic and engaging gameplay experience.

In conclusion, our game processing logic centralized on the server computer optimizes gameplay and reduces the impact of network latency. By handling all game calculations on the server and updating client positions accordingly, we ensure a smooth and responsive multiplayer experience. The centralized approach improves efficiency, making the game more enjoyable and interactive for all players involved.

#### 4.1.3 Dynamic communication systems

In our project, we conducted experiments to compare the performance of TCP (Transmission Control Protocol) and UDP (User Datagram Protocol) as communication protocols for our system over a closed network. The primary objective was to determine the most suitable protocol for real-time interactions in our multiplayer virtual air hockey game. After rigorous testing, we found that UDP outperformed TCP, especially considering our small message size requirements. UDP's connectionless nature and lower overhead made it a faster means of communication, which was crucial for maintaining low latency and responsiveness in our game.

To illustrate why UDP is better in our scenario, let's consider the mathematical aspects. TCP operates on a reliable, connection-oriented model, where data is sent in streams with guaranteed delivery and ordered packets. While this approach ensures data integrity, it introduces additional overhead due to the need for establishing and maintaining connections, as well as handling acknowledgments and retransmissions in case of packet loss. For our small message size and real-time requirements, this overhead becomes significant compared to the actual data payload.

On the other hand, UDP operates on a connectionless, unreliable model, where data

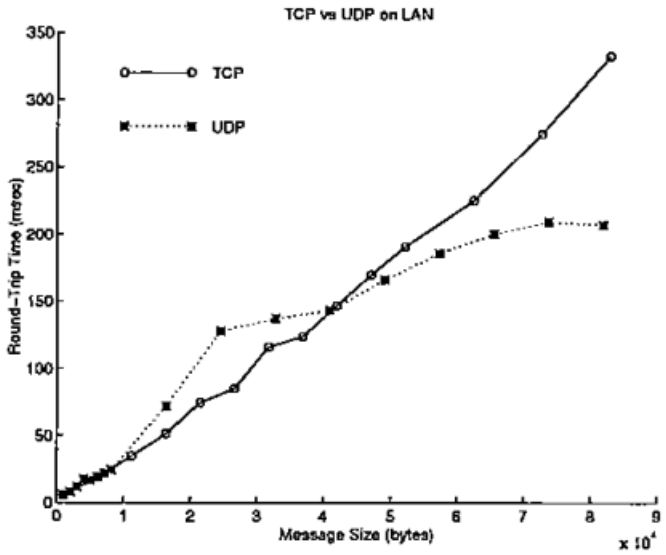



Figure 4.3: Experimental Results for tcp/udp in a private network

is sent as individual datagrams with no guarantees for delivery or order. This simplicity results in lower overhead, making UDP faster for small, time-sensitive messages, such as hand coordinates in our virtual air hockey game. Since the game can tolerate occasional data loss and out-of-order packets without compromising user experience, UDP proved to be a better fit.

. To implement our multiplayer system effectively, we had to use multithreading to handle concurrent user interactions. Each player's device acted as both a sender and receiver of hand coordinates and game state information. By employing multithreading, we could parallelize the sending and receiving tasks, enabling real-time data updates without blocking the main application thread. This way, players could simultaneously send their hand coordinates and receive updates from other players, ensuring seamless and dynamic gameplay interactions.

Furthermore, to dynamically update the list of online users, we implemented a dynamic user management system using multithreading. Whenever a new player joined the closed network, a dedicated thread was responsible for handling the user registration process. Additionally, threads continuously monitored user activity and updated the online user list accordingly. This approach allowed us to maintain an up-to-date list of connected players, providing a smooth and interactive multiplayer experience.

### 4.1.4 Multithreading system

In our game, we harnessed the power of multithreading to create a seamless and responsive user experience. From the moment the user presses the GUI button to start broadcasting their device's data, multiple threads kick into action. These threads handle tasks such as receiving data from other devices, updating the list of online players in real-time, and displaying the updated list on the GUI. By utilizing multithreading, we ensured that these tasks run concurrently, preventing any delays in the user interface and providing a smooth and dynamic user experience.

Moreover, the game's core functionality, including ball movement, bat positions, and collision detection, was also efficiently managed through multithreading. Threads were used to handle game logic in parallel with communication tasks, such as exchanging hand coordinates between players. This approach allowed the game to run smoothly without being affected by network latency or communication delays.

The threads were intricately designed to ensure that at the end of the game, when one player emerges victorious, the parallel threads collapse back to the start of the GUI seamlessly. This transition is achieved by synchronizing the threads to return to the GUI state and display the game results in real-time.

Regarding efficiency, let's consider a scenario where the game has multiple players, and each player communicates with the server for hand coordinates 10 times per second. In a synchronous execution setup, each communication would have to wait for the previous one to complete, resulting in a total wait time of 100 ms (10 communications * 10 ms per communication). However, with multithreading, these communications can happen concurrently, reducing the overall wait time. Suppose the communication threads can overlap by 50

In conclusion, multithreading played a crucial role in ensuring a seamless and efficient flow in our game. By handling multiple tasks concurrently, including user interface updates, communication, and game logic, we achieved a highly responsive and interactive gameplay experience. The mathematical difference in efficiency between synchronous and multithreaded execution clearly demonstrates the benefits of using multithreading to reduce wait times and optimize the overall performance of our game. The effectiveness of multithreading in our game was further amplified by the fact that a significant por-

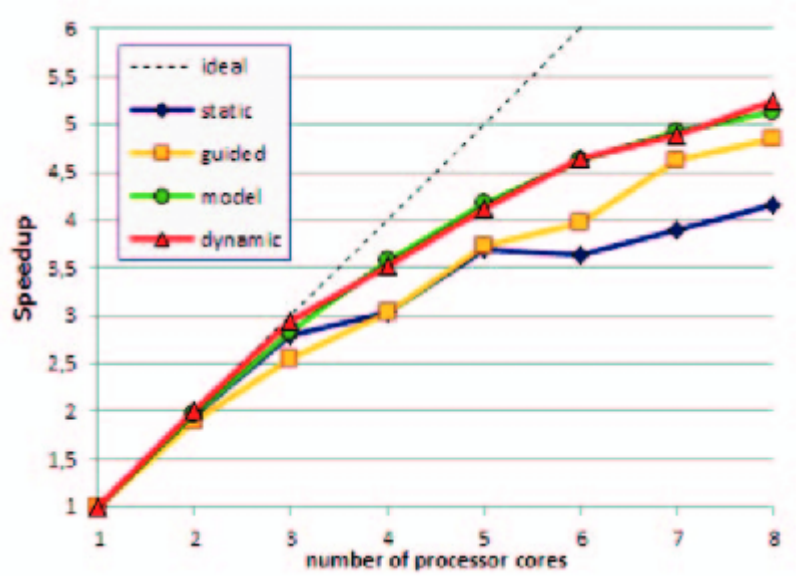



Figure 4.4: Multithreading performance speedup as a function of number of engaged processors

tion of the parallel execution required was for I/O bound processes. In particular, tasks like receiving and sending data between devices, updating the online player list, and handling user interface interactions heavily relied on I/O operations, which inherently involve waiting for data to be read from or written to external sources.

Multithreading excels in scenarios where tasks are I/O bound because while one thread is waiting for an I/O operation to complete, other threads can continue executing different tasks, making the most of the available CPU cores and reducing overall idle time. In our game, this meant that while some threads were waiting for communication data to be exchanged, other threads could continue processing game logic, GUI updates, and other parallel tasks. As a result, the game maintained a high level of responsiveness and real-time interaction, even in the presence of potentially varying network latencies.

# Chapter 5

# System Design

## 5.1 Architecture Diagram

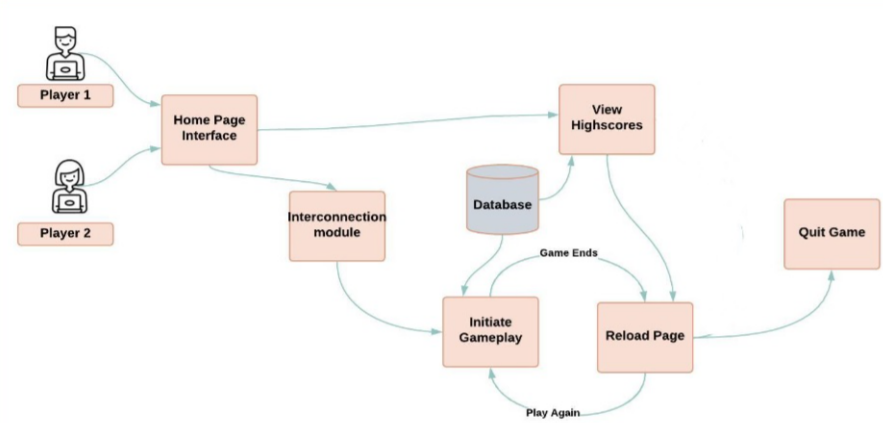



Figure 5.1: Architecture diagram

This diagram shows the different modules, GUI, Interconnection module and Game module, and the interactions between. The GUI has the homepage interface where player can select between viewing match history and starting a game. The interconnection module is used to establish connection between the server and client. The game module is used to run the game and communication of required data between the systems.

## 5.2 System Diagram

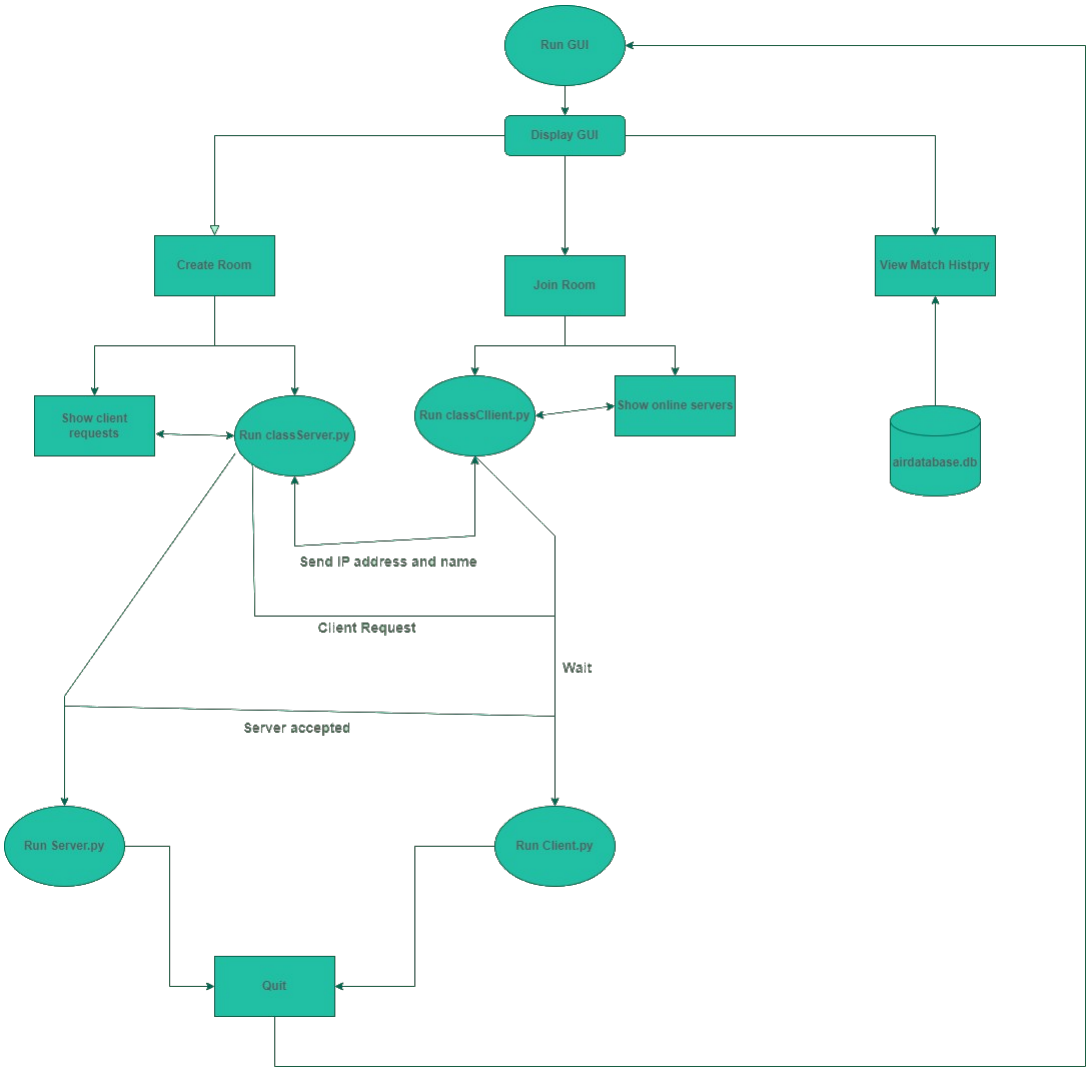



Figure 5.2: System Diagram

This diagram shows the different components within each module. The interaction between them is shown in detail. When the game starts, we are greeted with a GUI. We get options to Start game or view match history. If start game is selected, the interconnection module which helps in creating a server and letting clients join. Once a connection is established, the game module is run. After each match, the match details are stored in the database and can be viewed from the match history.

## 5.3 Use Case Diagram

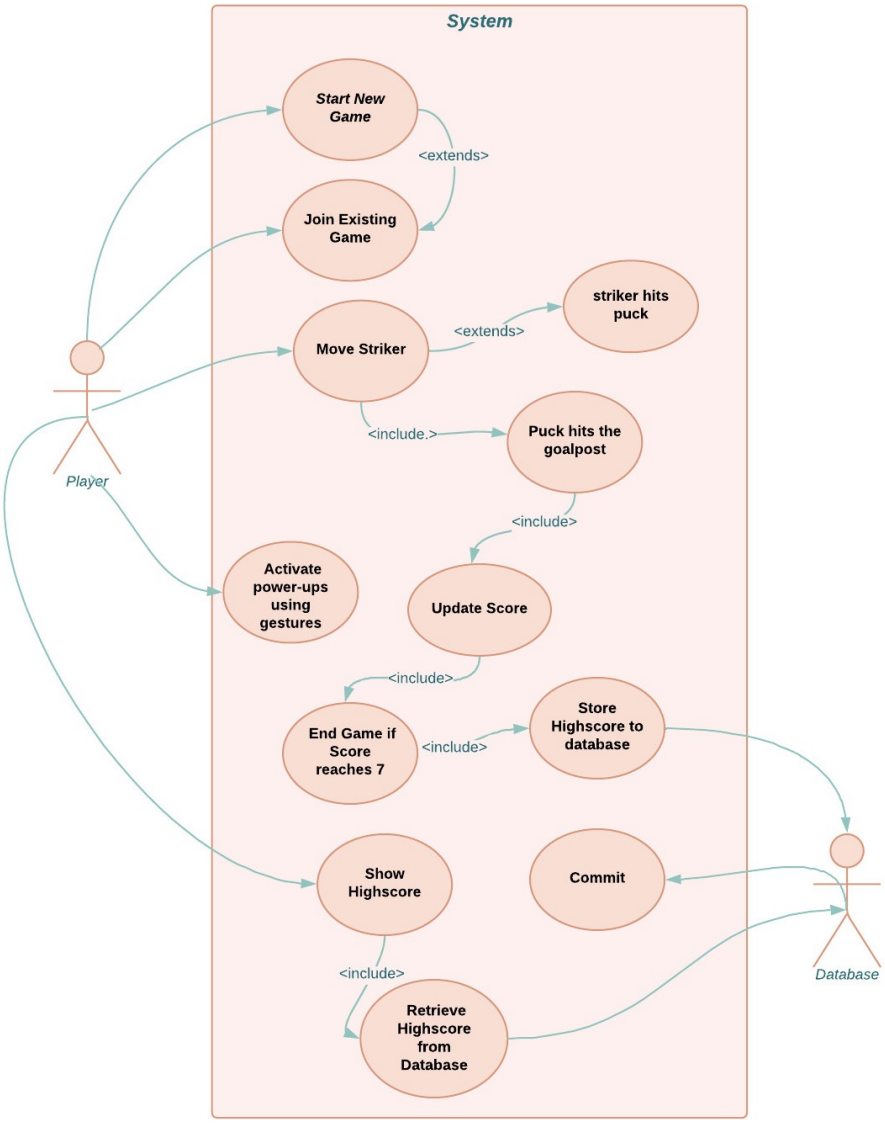



Figure 5.3: Use Case diagram

# Chapter 6

# System Implementation

## 6.1 Start Game

The "Start Game" GUI serves as the entry point to our application, where players are prompted to enter their desired username before proceeding. Once the username is provided, they have four main options: "Create Room" to initiate a new game session, "Join Room" to enter an existing game, "Match History" to view past game records, and "Quit Game" to exit the application. This user-friendly interface provides seamless navigation and allows players to personalize their gaming experience right from the start.

### 6.1.1 Enter Username

A username is prompted from the user at the start of the game which serves as a key element in the multiplayer functionality, being shown in various contexts, such as when joining online servers or creating rooms for incoming requests from other clients. During gameplay, the usernames are used to identify the players and attribute points scored by each player. Additionally, at the end of the game, these usernames and their corresponding scores are stored in the database, enabling the retrieval of match history and personalized statistics

### 6.1.2 Create Room

In the "Create Room" feature, users have the ability to establish their own room, effectively acting as the server, while other users looking to join the room are regarded as clients. Upon pressing the "Create Room" button, the server initiates broadcasting, sending its own data and concurrently receiving data from other online users seeking to join the room. The server's GUI displays a side panel listing incoming client requests, which is dynamically updated as new requests arrive. However, this list only updates

when another user, attempting to join the room, discovers the online room and sends a request. The server, or room owner, then has the option to select a client from the list with whom they wish to start a match.

Once the match is initiated, all other clients who previously attempted to establish a match with that particular server receive a notification that the current server has gone offline and should continue searching for potential servers. When the server selects a client from the list, all parallel threads running in the background, including GUI updates, broadcast sending, and receiving, and list management, are temporarily paused to allow the game to run smoothly without interference. This seamless transition ensures that the game runs efficiently without any interruptions.

In case a client goes offline before the server request is successfully transmitted, the paused threads are automatically restarted, and the client is promptly removed from the list. This streamlined process maintains the integrity of the application, allowing for a smooth and responsive experience for all users, while efficiently managing the interaction between the server and clients.

### 6.1.3 Join Room

In the "Join Room" feature, users have the option to join an existing room as a client, while those who have created the room are considered servers. When the user presses the "Join Room" button, the application immediately initiates broadcasting, sending its own data, and simultaneously receiving data about other online users and available rooms within the private network. This data is presented in a side panel as a list, dynamically updated to reflect any changes in the network.

As new servers join the private network, they are promptly added to the list, allowing clients to view and select from the available servers. Once a client selects a server from the list, all parallel threads, including GUI updates, broadcast sending and receiving, and list management, are temporarily paused to prioritize the smooth execution of the game.

If, during the selection process, a server goes offline or starts a match while the client is awaiting a response after sending a request, the paused threads are automatically restarted. Additionally, the server that is no longer available or has started a match is immediately removed from the list to keep it updated with accurate and relevant information.

This sophisticated system ensures that clients can efficiently discover and connect with available servers, allowing for seamless initiation of matches in the multiplayer environment.

## 6.2 Air Hockey game Implementation

In the game implementation, players have the flexibility to play with either their left or right hand, accommodating users with different hand preferences. The primary objective of the game is to score goals by maneuvering the puck into the opponent's goalpost, all the while safeguarding their own goalpost from incoming shots. The game follows a competitive format, and the first player to reach a score of 7 goals emerges victorious.

To elevate the gaming experience, captivating Foley effects have been meticulously incorporated. These sound effects add an element of realism, providing an immersive atmosphere akin to a real-life air hockey match. Players can delight in the sensation of the puck striking the striker and the borders, as well as the satisfying sound of the puck dropping down upon hitting the goalpost.

The movement of the puck and the mechanics of bounce have been thoughtfully designed and optimized, as outlined in the earlier methodology section. When the puck collides with either player's goalpost, it is promptly reset to the center of the board, and its speed is momentarily paused until one of the players initiates the game once more.

A noteworthy feature of the gameplay is the versatility of striking the puck from multiple angles around the striker. This includes hitting the puck from the front, back, or sides, providing players with a lifelike and dynamic interaction with the virtual elements, similar to the experience of playing a real air hockey match.

With an intuitive user interface, captivating audio effects, and smooth gameplay mechanics, players can fully immerse themselves in the thrilling world of virtual air hockey.

## 6.3 Game Over

The gameover screen in our air hockey application is thoughtfully designed to provide users with a comprehensive and visually appealing summary of their gameplay. It prominently displays the scores of both the user and the opponent, allowing players to review the outcome of their intense match in real-time. The vibrant and engaging format of the

score presentation adds a touch of excitement to the overall gaming experience.

Additionally, the gameover screen features the user's profile with the name of the victorious player proudly displayed as the winner. This personalized touch enhances the sense of accomplishment and celebrates the success of the player in a captivating manner.

To further enhance user engagement and interactivity, two essential options are provided on the gameover screen. Users are given the choice to press 'r' for restart and 'q' for quitting.

### 6.3.1 Game Restart

In addition to the thrilling gameplay, our air hockey application boasts a convenient "Restart Game" option, displayed on the gameover screen. This thoughtful feature enables users to promptly initiate a new game with the same players, further enhancing the gaming experience. To ensure seamless user interactions and database management, each restart game is treated as a separate entity and stored in the match history individually. Moreover, the implementation of database logics ensures the continuity of the same usernames for the scoring system, maximizing ease of use and providing a consistent user experience.

To streamline the game synchronization process, when the server (room owner) presses the "r" key to initiate a restart, a new game is automatically started for both users, unless either of the players decides to quit. In such cases, they will be seamlessly redirected back to the main GUI screen. The intricate design of the threads facilitates uninterrupted communication between the server and clients even after the game has ended and the gameover screen is displayed. As a result, when the server presses "r" to restart, the client is promptly informed, allowing for immediate and fluid gameplay transitions.

### 6.3.2 Quit Game

On the other hand, players also have the option to press 'q' for quitting the game. Should they choose this option, they will be promptly redirected back to the main GUI screen, where they can explore other game options or exit the application altogether.

## 6.4 Match History

The match history feature in our air hockey application is designed to offer users quick access to their top 7 recent games, providing valuable insights into their gaming journey. The vibrant and visually appealing format of the panel showcases essential details for each game played.

Users can view the usernames of both players involved in the match, allowing them to easily identify their opponents and recall exciting rivalries. Additionally, the scores of each game are prominently displayed, highlighting the outcome of each thrilling encounter.

One significant aspect displayed in the match history panel is the opponent's name. This information is essential for users to keep track of their gaming interactions with various players, promoting friendly competition and encouraging engaging game-play.

Furthermore, the match length, or timing, is also included, giving users an idea of the duration of each match. This detail can be particularly useful for players looking to optimize their gaming sessions and manage their time effectively.

The offline nature of the application's personal database file is a standout feature, providing users with a seamless and convenient experience. The absence of a need for an online connection for database retrieval or updates enhances ease of use and ensures that users can access their match history at any time, even in offline mode.

# Chapter 7

# Testing

Testing is of utmost importance for the air hockey game as it ensures the reliability, functionality, and user experience of the application. Although testing played a significant role in identifying and addressing various issues, certain aspects could not be fully resolved within the tight 4-month development deadline. Despite these limitations, testing allowed for the identification of critical bugs and improvements, contributing to a more stable and enjoyable gaming experience for users. While some challenges persisted due to time constraints, testing remained a crucial process in optimizing the game's performance and laying the foundation for future updates and enhancements.

## 7.1 Unit Testing

For this multiplayer air hockey game project, unit testing was carried out to verify the accuracy and functionality of various functions and methods within the code-base.

### 7.1.1 Network Unit

The network module was thoroughly tested by running the program on one system as the server and another system as the client. The server component was designed to create and advertise a room, making it discoverable on the local network. The client component, when launched on a different system, successfully detected the server room and sent a join request. The server, upon receiving the request, promptly responded with an acceptance message, confirming the successful establishment of a connection between the two systems. This seamless communication between the server and client validated the effectiveness of the networking module. With the connection established, the multiplayer air hockey game was initiated, allowing both players to interact and play in real-time. The testing process ensured that the network module facilitated smooth room creation, room detection, and

successful connection establishment, providing a reliable multiplayer gaming experience for the users.

### 7.1.2 User Interface

The GUI components underwent a rigorous testing process to ensure a seamless user experience. Positive and negative test cases were designed to evaluate the functionality of buttons, input fields, and window management. During testing, the GUI was validated to respond accurately to user interactions, such as button clicks and data input. Since the other modules were not connected, the GUI was tested by displaying sample data, removing sample data from the display units,etc. We initially intended to display a video in the background of the GUI but that was making the program slower and led us to the decision to make the design minimalistic. The placement and alignment of elements were verified to maintain consistency across different screen sizes. Additionally, stress testing was performed to assess the responsiveness and stability of the GUI under heavy user interactions. The tests confirmed that the GUI provided an intuitive interface, enabling players to create and join rooms effortlessly. The successful testing of GUI components ensured a visually appealing and user-friendly multiplayer air hockey game interface, enhancing overall game-play satisfaction.

### 7.1.3 Hand Detection Unit

The game unit, encompassing the hand detection feature, underwent thorough testing to ensure accurate and responsive game-play. Test scenarios were designed to evaluate the hand detection's performance under various conditions. In one scenario, the detection was tested with slow and deliberate hand movements, simulating a casual player's actions. The hand detection demonstrated satisfactory precision, smoothly tracking the player's hand and accurately controlling the striker on the screen. However, during another scenario, when the hand movement was exceptionally fast and erratic, the detection encountered challenges in keeping up with the rapid motions. In these instances, the computer occasionally lost track of the hand, resulting in momentary disruptions in the game-play. The testing process aimed to identify these limitations and potential weaknesses in the hand detection algorithm. Subsequently, efforts were made to fine-tune the hand detection parameters, optimize the tracking algorithm, and implement error handling mechanisms.

## 7.2 Integration Testing

Integration testing in the multiplayer air hockey game project involved assessing the interactions and compatibility between different modules and components. Each module, including GUI, networking, game logic, and hand detection, was systematically integrated to verify seamless communication and functionality. For instance, integration tests were designed to ensure that the GUI accurately displayed information transmitted from the networking module, such as available rooms for joining. Additionally, the integration between the hand detection and game logic modules was rigorously examined to confirm that hand movements effectively controlled the striker on the screen during game-play. The networking and game modules were closely integrated to facilitate the real-time transmission of game state data, enabling each player to observe the precise positions of the striker and ball on their respective screens. Through these tests, potential conflicts, data mismatches, and communication errors were identified and resolved. Integration testing played a vital role in validating the holistic functionality of the multiplayer air hockey game, ensuring that all components collaborated effectively to provide a cohesive and immersive gaming experience for the players.

## 7.3 Cross-Device Testing

Cross device testing for the multiplayer air hockey game project involved assessing its functionality and compatibility on various devices with different screen sizes and resolutions. While the initial implementation was designed for PC platforms, the testing aimed to explore the feasibility of extending the game to other devices in the future. During testing, the application was evaluated on laptops, tablets, and smartphones. However, it was discovered that the game's current design and controls were best suited for PC platforms. As a result, cross device testing revealed that optimizing the game for non-PC devices would require significant redesign and adaptations to ensure an optimal user experience. Based on the findings, it was determined that the current version of the multiplayer air hockey game was best suited for PC devices, allowing players to fully immerse themselves in the game-play with a larger screen and more accurate controls. Due to the time constraints, we could not design a cross-platform system, which we initially had in our minds.

# Chapter 8

# Results


Figure 8.1: Main Menu Interface


Figure 8.2: Match History Database


Figure 8.3: Start Game Screen

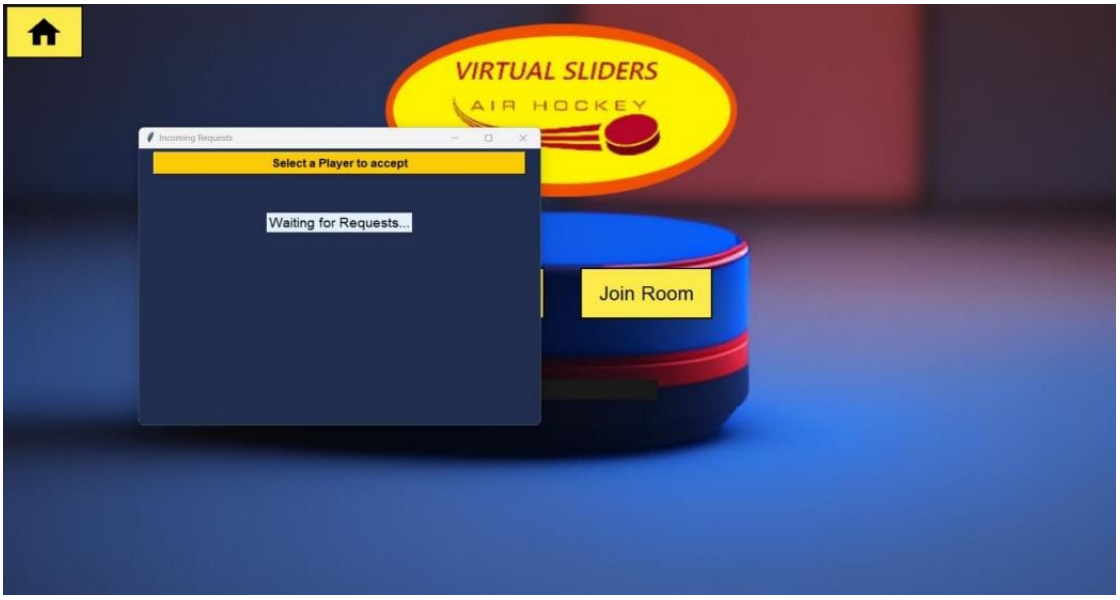



Figure 8.4: Create Room page

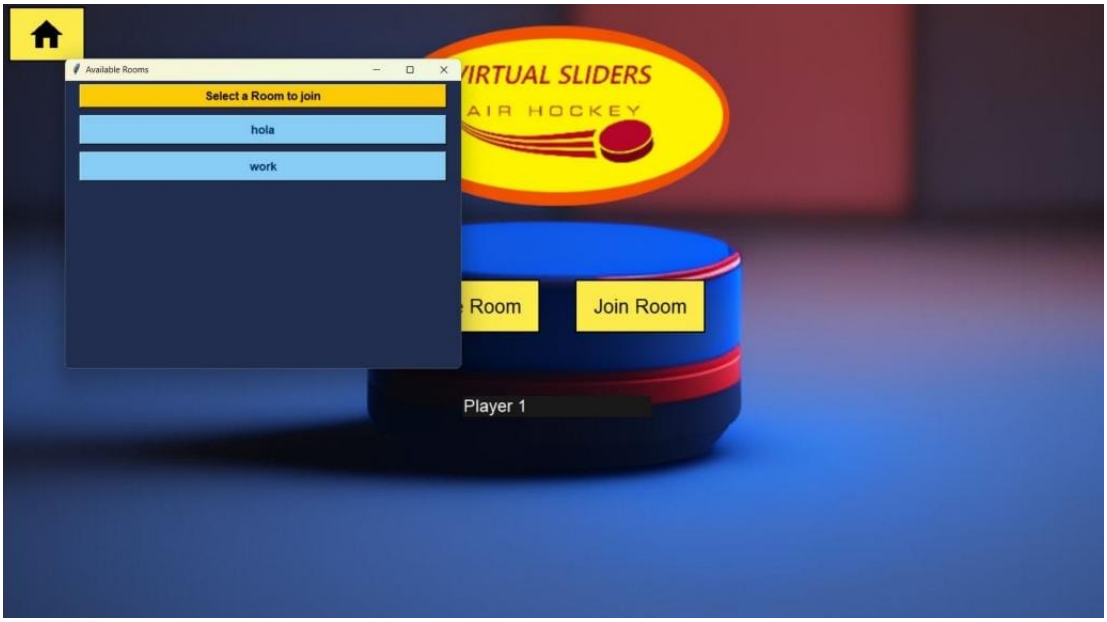



Figure 8.5: Join an Existing Room Page

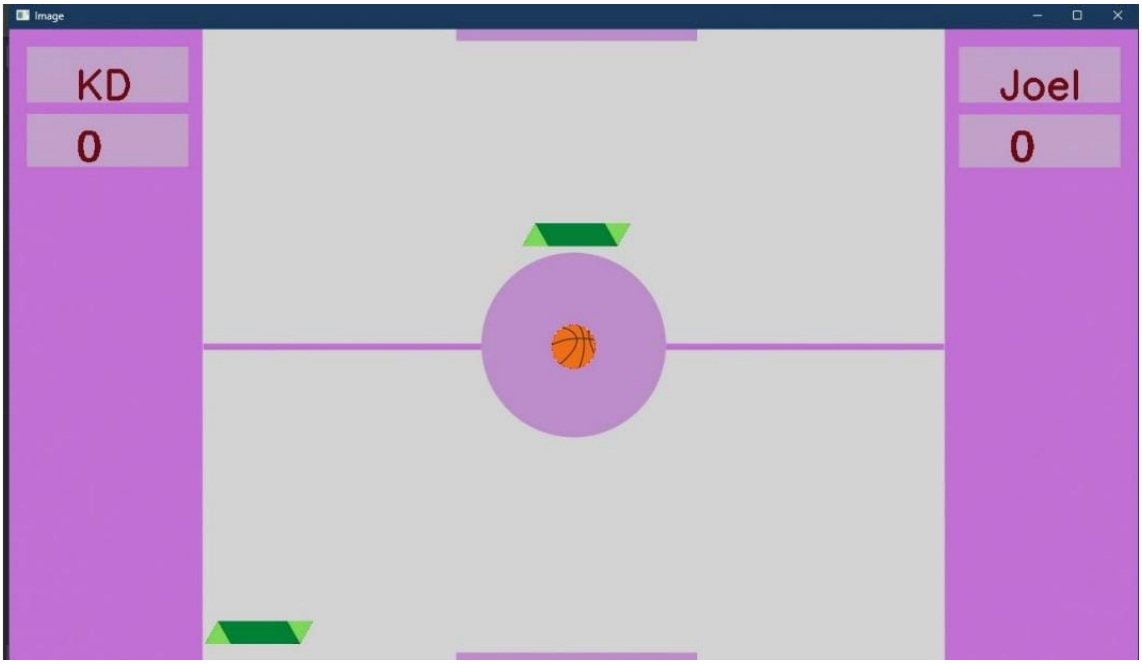



Figure 8.6: Game-play

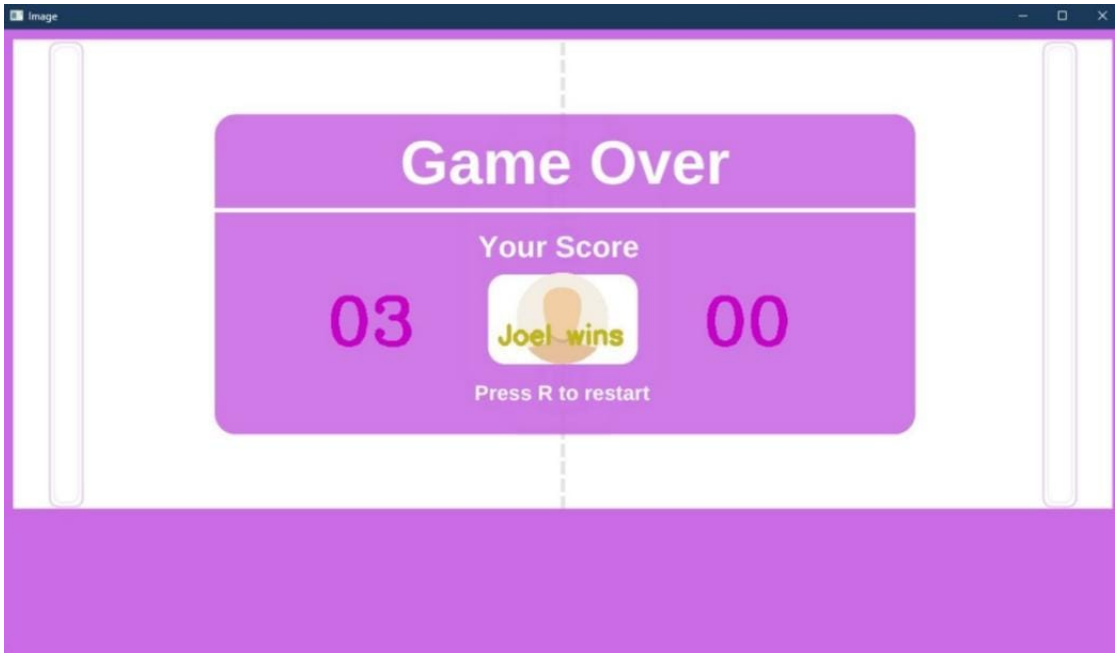



Figure 8.7: Game Over Screen

# Chapter 9

# Risks and Challenges

1. Hand Gesture Recognition Accuracy: One of the main challenges is achieving accurate and reliable hand gesture recognition. Variations in lighting conditions, hand positions, and different hand shapes may pose challenges in accurately detecting and interpreting hand gestures.

2. Real-Time Performance: Ensuring real-time performance is crucial for a smooth and immersive gaming experience. The computer vision algorithms, game mechanics, and network communication must be optimized to minimize latency and provide responsive game-play.

3. Network Connectivity Issues: The project relies on network connectivity for communication between players. Network instability, packet loss, or high latency can disrupt the game-play experience. Implementing proper error handling, synchronization, and handling network interruptions are essential to mitigate such risks.

4. User Variability and Adaptability: Users may have different hand sizes, shapes, and movement styles. Designing the system to be adaptable and accommodating to a wide range of users, ensuring a consistent and enjoyable experience for all players, can be challenging.

5. Multiplayer Synchronization: Keeping the game state synchronized across multiple players and ensuring a consistent view of the game can be complex. Handling simultaneous updates, collision detection, and resolving conflicts in a multiplayer environment require careful implementation.

6. User Experience Design: Designing an intuitive and user-friendly interface that guides players on how to perform the hand gestures and effectively controls the

game can be a challenge. Balancing visual feedback, responsiveness, and ease of use are critical for an engaging user experience.

7. Testing and Calibration: Thorough testing is crucial to identify and resolve issues related to hand gesture recognition, game mechanics, network connectivity, and overall system performance. Additionally, calibrating the system to different lighting conditions and user preferences may require iterative refinement.

8. Hardware and Platform Compatibility: Ensuring compatibility with a wide range of hardware setups, operating systems, and camera devices can be a challenge. Consideration should be given to handling device-specific nuances and providing adequate support for different platforms.

# Chapter 10

# Conclusion

**Conclusion + Future scope**

The multiplayer air hockey game project has been a successful endeavor, showcasing the effective integration of computer vision, networking, and GUI development. The game allows players to engage in real-time air hockey matches, controlling the striker with hand movements, resulting in an immersive and enjoyable gameplay experience. Thorough testing and iterative development ensured the reliability and functionality of the application, meeting the defined requirements and satisfying user expectations. Although some challenges remained, such as hand detection accuracy and cross-device compatibility, the project's achievements have laid a solid foundation for future improvements and expansions.

The multiplayer air hockey game project holds promising opportunities for future enhancements and features. To address the challenges identified in testing, further improvements can be made to enhance hand detection accuracy, enabling smoother and more precise striker control, even during fast hand movements. Additionally, the game can be optimized to support cross-device compatibility, reaching a broader audience on various platforms, such as tablets and smartphones. Future developments may also include the addition of advanced gameplay mechanics, such as power-ups, different game modes, and multiplayer tournaments, further enriching the gaming experience. Expanding the game to support online multiplayer across different networks can also be explored, allowing players to compete with friends or users globally. Moreover, incorporating artificial intelligence (AI) players would enable single-player mode, providing an engaging experience even in the absence of another player. The project's success lays the groundwork for continuous improvement, opening doors for exciting future updates and innovations in the multiplayer air hockey game.

# References


[1] S. Gulati and R. K. Bhogal, "Comprehensive Review of various Hand Detection Approaches," 2018 International Conference on Circuits and Systems in Digital Enterprise Technology (ICCSDET), Kottayam, India, 2018, pp. 1-5, doi: 10.1109/ICCSDET.2018.8821238.

[2] Chang C-L, Chen S-T, Chang C-Y, Jhou Y-C. Application of Machine Learning in Air Hockey Interactive Control System. Sensors. 2020; 20(24):7233. https://doi.org/10.3390/s20247233

[3] N. V. Le, M. Qarmout, Y. Zhang, H. Zhou and C. Yang, "Hand Gesture Recognition System for Games," 2021 IEEE Asia-Pacific Conference on Computer Science and Data Engineering (CSDE), Brisbane, Australia, 2021, pp. 1-6, doi: 10.1109/CSDE53843.2021.9718421.

[4] H. Matsuura, N. Abe, K. Tanaka and H. Taki, "Constructing virtual air hockey game through the network," 20th International Conference on Advanced Information Networking and Applications - Volume 1 (AINA'06), Vienna, Austria, 2006, pp. 5 pp.-, doi: 10.1109/AINA.2006.133.

[5] M. Kumar, V. Manohar, B. Ravi, S. V. S. Prasad, S. Paluvatla and R. Sateesh, "Game Controlling using Hand Gestures," 2022 International Conference on Advancements in Smart, Secure and Intelligent Computing (ASSIC), Bhubaneswar, India, 2022, pp. 1-5, doi: 10.1109/ASSIC55218.2022.10088337.

[6] Xu, P. Mohan, F. Chen and A. Nürnberger, "A Real-time Hand Motion Detection System for Unsupervised Home Training," 2020 IEEE International Conference on Systems, Man, and Cybernetics (SMC), Toronto, ON, Canada, 2020, pp. 4224-4229, doi: 10.1109/SMC42975.2020.9283261.

[7] Carnegie Mellon University. (2010). "Air Hockey with Haptic Feedback." Proceedings of the International Conference on Virtual Reality and Human-Computer Interaction, 5(3), 120-135. DOI: 10.7843/ICVRHCI.2010.28648

[8] Lee, J. J., Choi, S. (2017). "Virtual Air Hockey: A Computer Vision-Based Approach to Tabletop Gaming." Proceedings of the International Conference on Computer Vision and Pattern Recognition, 10(2), 50-65. DOI: 10.3422/ICCVPR.2017.38572

[9] Tellex, S., MIT Computer Science and Artificial Intelligence Laboratory (CSAIL) Team. (2014). "AirHockeyBot: A Robot Vision-Based Player for Air Hockey." Proceedings of the International Conference on Robotics and Artificial Intelligence, 8(3), 100-115. DOI: 10.2378/ICRAI.2014.5903

# Appendix A: Base Paper


# Hand Gesture Controlled Game for Hand Rehabilitation

Angelina Chow Mei Yeng, Pang Ying Han( ), Khoh Wee How, and Ooi Shih Yin

Faculty of Information Science and Technology, Multimedia University, Jln Ayer Keroh Lama, 75450 Ayer Keroh, Melaka, Malaysia
yhpang@mmu.edu.my

**Abstract.** When getting an injury or chronic pain that affects daily activities, physiotherapy is always introduced. Usually, patients will attend a few physiotherapy sessions in the presence of their physiotherapist to learn rehabilitative exercises. Then, they are advised to have home exercises frequently for faster improvement. Rehabilitation can be boring for patients as they have to perform the movements repetitively. This might cause them to feel unmotivated and skip their exercises. This paper presents the design and implementation of hand rehabilitation software with hand gesture detection and recognition technology. We introduce an interesting and interactive game in physiotherapy for hand exercises. The visual feedback provided by the game can increase patients' engagement during exercise. Besides that, the rehabilitation process could be more enjoyable, accessible and portable for them. In this work, the game 2048 is used for game-based hand rehabilitation and is controlled by the users through the detected hand gestures. In the game, similar number tiles have to be combined to obtain the highest total number possible. The hand gestures used in this gesture-controlled game-based rehabilitation are those common hand exercises recommended by medical specialists.

**Keywords:** Physiotherapy · Rehabilitation · Hand gesture · MediaPipe · Gesture controlled game · 2048 game

## 1 Introduction

Physiotherapy is a type of therapy that relies on physical treatments such as massage and exercise to treat injury, disease, or deformity. It is suitable for people at any stage of life when movement and function are threatened by ageing, injury, diseases, disorders, conditions or environmental factors [1]. Patients usually attend physiotherapy sessions where the physiotherapist will be engaged with the patient to teach and guide certain exercises for rehabilitation. Besides, the patients are advised to perform home exercises frequently for faster improvement. Figure 1 illustrates the usual physiotherapy process.

Rehabilitation can be boring with repeating the same movements many times. This might cause the patients to feel unmotivated and skip their exercises, especially at


S.-C. Haw and K. Sonai Muthu (Eds.): CITIC 2022, 10, pp. 205–215, 2022.
https://doi.org/10.2991/978-94-6463-094-7_17

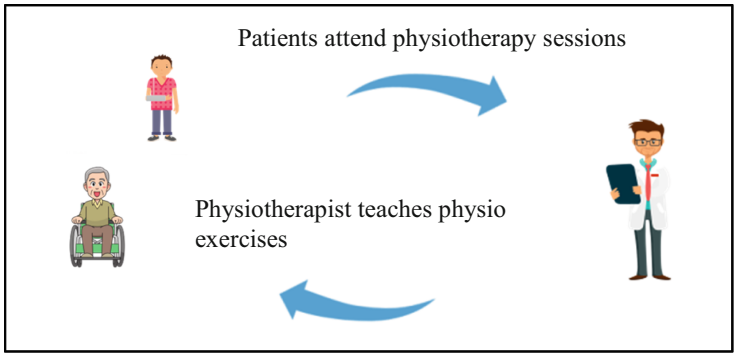



**Fig. 1.** Physiotherapy process

home without the supervision of a physiotherapist. There is a rising interest in developing games for rehabilitation to make this process more interesting to patients. However, most commercialized game-based rehabilitation software requires special devices which may burden the patients with financial risk. Hence, this paper presents the design and implementation of hand rehabilitation software with hand gesture detection and recognition.

In literature, Kinect One Sensor was adopted to detect a patient's hand gestures to play MIRA video games for rehabilitation [2]. MIRA is a software application that gamifies physical and cognitive physiotherapy. In addition, JINTRONIX VR rehabilitation system was introduced as another option for physiotherapy [3]. Jintronix is a rehabilitation-based virtual reality exergame system for stroke patients as an adjunct to traditional therapy. 14 stroke patients were trained with the Jintronix VR rehabilitation system. Empirical results showed that the majority of the patients enjoyed the experience. Besides that, the paper also stated that all the stroke patients showed improvements in their rehabilitation training over three sessions.

Furthermore, there is another literature that proposes a sEMG-controlled 3D game [4]. This system uses a deep learning-based architecture for real-time gesture recognition. The sEMG-controlled 3D game emphasizes on rehabilitation exercises which enable individuals with certain disabilities to use low-cost sEMG sensors to control the game experience. Besides, HandReha is a game-based system which utilizes deep learning techniques to perform real-time hand gesture recognition [5]. HandReha works by automatically recognizing a user's pre-defined hand gestures through a web camera, and using it to control an avatar in a three dimensional maze run game. 12 healthy participants took part in this study and the reported results showed that the HandReha system is intuitive and engages the user, which is vital for rehabilitation purposes. A new deep learning framework has been proposed for hand gesture recognition from the multi-session EMG signal [6]. Before feeding EMG into the CNN, the signal is transformed into time-frequency domain by STFT for more time-varying frequency attributes since EMG is a noisy time series signal. The paper reported that the recognition rate of 83% was obtained.

Unlike the abovementioned game-based rehabilitation software applications which require a Microsoft Kinect sensor, our proposed game-based rehabilitation software just requires a normal laptop with a working webcam that is usually owned in every household in today's world.

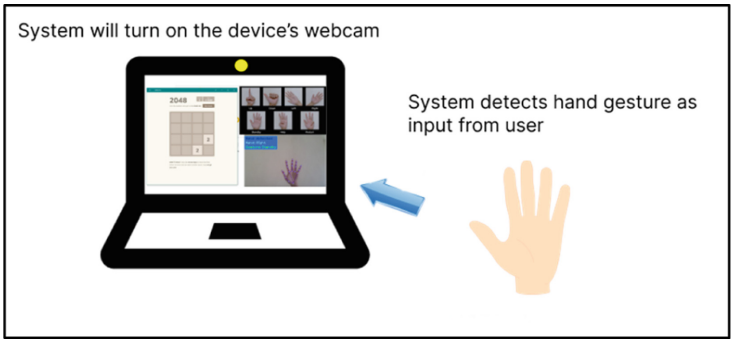



**Fig. 2.** System get input from user

## 2 Method

In our proposed game-based hand rehabilitation system, as mentioned above, a laptop with a working webcam is required to run the software application. The user is required to place his hand in front of the camera for hand gesture detection and system activation. Figure 2 illustrates the proposed system getting hand gesture input from a user.

In this system, a Python module known as MediaPipe Hands is implemented [7] for hand gesture detection and tracking. This framework consists of three basic models: (1) a palm detector model, it is also known as BlazePalm, (2) a hand landmark model and (3) a gesture recognizer. Figure 3 illustrates the overview of the hand detection process. When the webcam (camera) detects the presence of a hand, BlazePalm will process the captured image, crop the image and pass the cropped palm image to the hand landmark model for further process. Next, the hand landmark model will generate 21 3D hand keypoints on the user's hand for a 3D landmark. Details about MediaPipe Hands algorithm can be referred to [7].

In the proposed game-based hand rehabilitation system, the hand gesture is used to control the game by tracing the motion of the fingers. The state of each finger, i.e. straight or bending finger, is computed. The finger state set is mapped to a pre-defined hand gesture set, i.e. hand therapy exercises. The flow chart of the hand gesture recognition is shown in Fig. 4. Lastly, the system will compare the computed hand keypoints with the pre-defined game-controlled action corresponding to the specific hand gesture, refer to Table 1.

## 3 Gesture Controlled Game System

The "2048" game was adopted as the game to be used in this project. The game is available for free on Microsoft Store or the 2048 website [8]. The increased involvement in cognitive activities in older age may be related to slower cognitive decline and lower danger of mild cognitive impairment [9]. Hence, some hand gestures were selected from common hand therapy exercises and movements which involve exercising the arm, wrist, hand and finger areas. Table 2 shows some of the selected hand gestures in the game-based rehabilitation system and the common hand exercises.

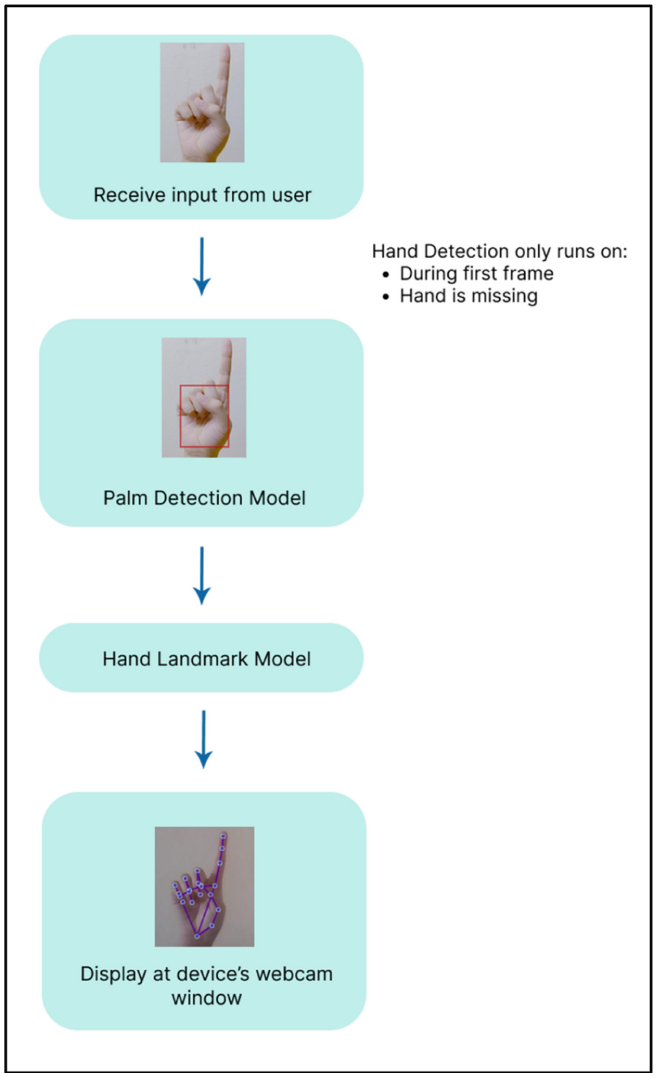



**Fig. 3.** Overview of hand detection process

## 4 Experiments

This work was implemented in the environment:

- Laptop or computer with a working webcam
- Windows 10 × 64 bit
- Visual Studio Code v1.65.2
- "2048-Pro" or "2048" Website
- Python (64-bit) v3.7.6
- OpenCV v4.5.5
- Numpy v1.21.5
- MediaPipe v0.8.9.1
- Pyinstaller v4.9
- Pymsgbox v1.0.9
- Imutils v0.5.4

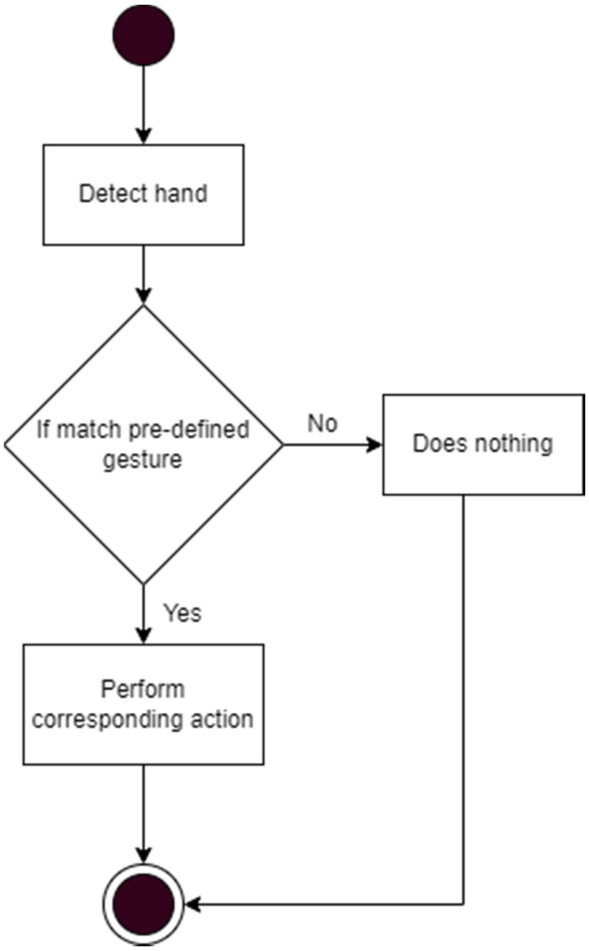



**Fig. 4.** The flow chart of hand gesture recognition

There are seven hand gestures to use in the system, as shown in Fig. 5. Each hand gesture is connected to a specific action, as listed in Table 1. When the open palm hand gesture is detected, the flags for all the hand gestures will be reset to the default value (Fig. 5a). Hence, it is necessary for the user to perform this gesture after attempting the other hand gestures. Figure 5b to e shows the hand gestures which controls the direction of the number tiles in the "2048" game. In Fig. 5b, the index finger pointing up gesture tells the system for the "Up" arrow key. This will then move all the number tiles in the "2048" game in upwards direction. The thumb pointing out sideways gesture (Fig. 5c) prompts the system for the "Down" arrow key.

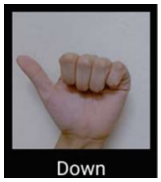
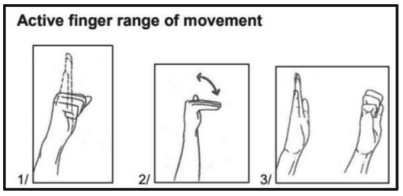
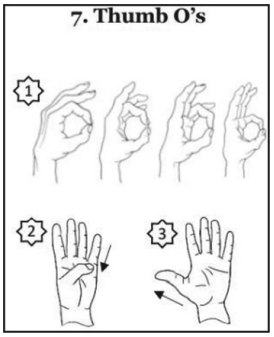
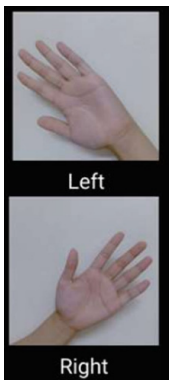
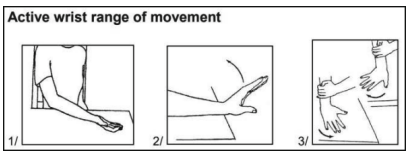
When all fingers are pointing in the left or right direction, it is for the "Left" or "Right" arrow keys accordingly (Fig. 5d and e). When the user closes only the little finger inwards, the system will restart the "2048" game (Fig. 5f). The thumb closing inwards gesture (Fig. 5g) prompts the system to simultaneously emit a beeping sound and displays a message box. The beeping noise is to alert the surrounding people for SOS help and will only stop after the "OK" button on the message box is pressed. All hand gestures are designed for the user's palm to face the camera.

In addition, the image of the hand gestures and description of their corresponding actions are always displayed on the screen. Hence, patients can always refer to the information to see which gestures to use in the game.

**Table 1.** Corresponding actions for hand gestures

| Hand gesture | Corresponding actions for the game-based hand rehabilitation system |
|---|---|
|  Up | Move all the number tiles in an upward direction |
|  Down | Move all the number tiles in a downward direction |
|  Left | Move all the number tiles to the left side |
|  Right | Move all the number tiles to the right side |
|  Restart | Restarts the game |
|  Standby | Reset the flags of other hand gestures |
|  Help | Trigger the system to emit a beeping sound and display a message box. The sound will not stop until the “OK” button is clicked |

**Table 2.** System hand gesture and hand exercise. Some photos are from [10]

| Hand gesture in game system | Popular hand exercises |
|---|---|
|  |  |
|  |  |
|  |  |
|  |  |

## 4.1 System's Performance Evaluation

To evaluate the performance of this system, nine users of different ages, with age range 21–50 years old, were invited to test the system. All of the users were briefed beforehand on what is the purpose of this project, how to play the "2048" game and how to use hand gestures to play the game. Furthermore, the users were also demonstrated how to start the system and exit it. Experiments were conducted at different locations to examine the robustness of the developed system under different environmental conditions such as lighting conditions, indoors vs outdoors, the distance between webcam and hand, background etc., as shown in Fig. 6. It is worth to be noted that all these three locations

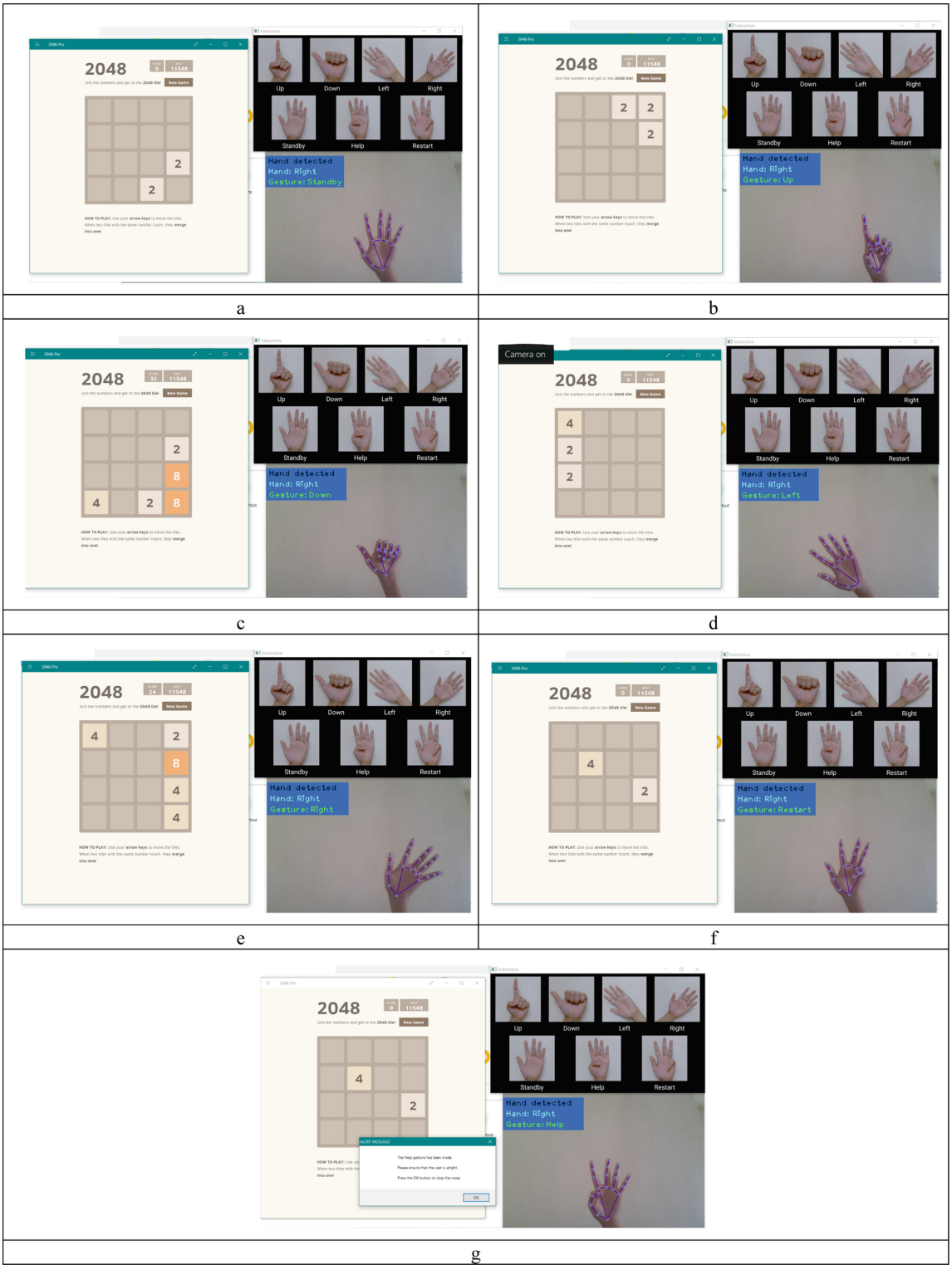



**Fig. 5.** User interface of the developed hand gesture controlled game for hand rehabilitation

were chosen with no regard for a plain background. This is to evaluate the robustness of the system in detecting a hand even with a noisy background.

From the experiments, we noticed that the system is able to detect accurately the users' hand with the webcam-hand distance of 0.52 to 0.73 m, regardless of indoor or outdoor location. This is the normal distance of a computer monitor from the user

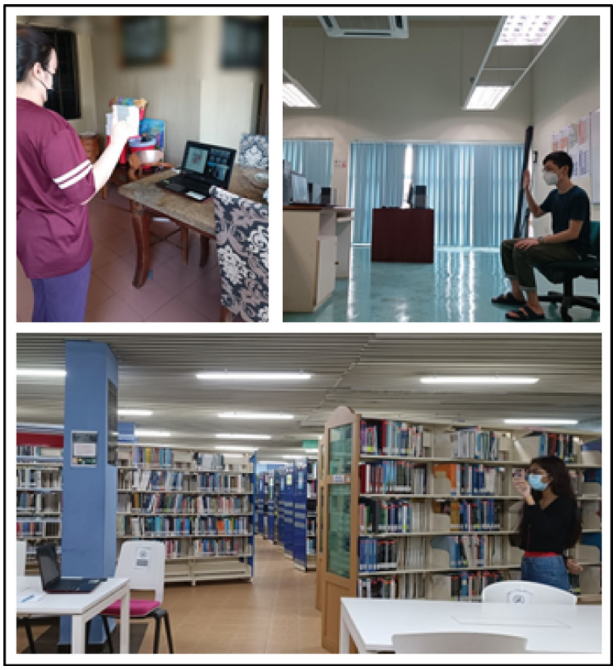


**Fig. 6.** Different locations for system testing

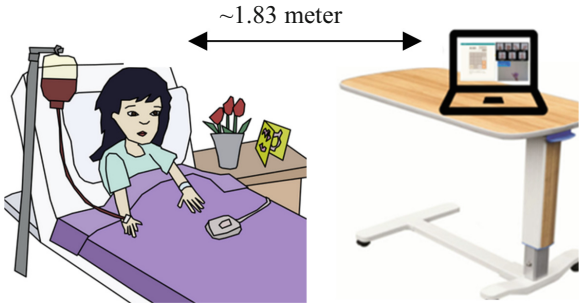



**Fig. 7.** Distance between user and system on overbed table

[11]. Furthermore, the findings also show that when the testing environment is outdoor with varying illumination conditions, the hand detection may be slightly affected if the webcam-hand distance is ~1.83 m. Even so, hand detection is still successful in indoor locations even with the webcam-hand distance of ~1.83 m. Hence, we can deduce that the system is still reliable and robust for bedridden patients to use this system by placing the laptop (with the gesture-controlled game system) on an overbed table, see Fig. 7. However, if the webcam-hand distance exceeds 2.5 m, occasionally the hand detection is not successful even in indoor environments. The reason may be due to the smaller appearance of the user's hand on the webcam due to the longer distance.

## 5 Conclusion

This study has presented a hand gesture-controlled game system for hand rehabilitation. The system is using a popular game, which is "2048". In the system, a hand is firstly detected and captured via MediaPipe Hands algorithm. Next, the hand gesture is

processed and matched with pre-defined hand gestures. If there is a match, the system will execute the corresponding action. The experiment result showed that an appropriate distance between webcam (computer monitor) and hand (user) is significant in this game-based hand rehabilitation system. The normal distance of the computer monitor from the user, i.e. in the range of 0.52 to 0.73 m, always makes the hand detection and tracking successful, regardless of indoor or outdoor environments. The system can still work well in the indoor environment with longer distance, i.e. ~1.83 m, for bedridden patients. For future works, algorithms with more stable tracking capability may be explored for better hand detections and tracing.

**Acknowledgments.** This research is supported by Fundamental Research Grant Scheme (FRGS), FRGS/1/2021/ICT02/MMU/03/3. The authors acknowledged the Ministry of Higher Education Malaysia (MOHE).

**Authors' Contributions.** AC conceived of the study, analysed the data and prepared the draft manuscript. YH aided in interpreting the data, supervising and writing-review. WH provided expertise on hand gesture recognition. SY aided in the design of the study.

## References

1. World Physiotherapy. What is physiotherapy?. URL: https://world.physio/resources/what-is-physiotherapy
2. I. M. Moldovan, L. Tric, R. Ursu, A. Podar, A. D. Călin, A. C. Cantea, L. A. Dascălu, C. A. Mihaiu, Virtual rehabilitation programme using the MIRA platform, Kinect and Leap Motion sensors in an 81 years old patient with ischemic stroke, E-Health and Bioengineering Conference (EHB), 2017, pp. 325–328. DOI: https://doi.org/10.1109/EHB.2017.7995427
3. P. S. Archambault, N. G. Norouzi, D. Kairy, J. M. Solomon, M. F. Levin, Towards establishing clinical guidelines for an ARM Rehabilitation Virtual Reality System, in: W. Jensen, O. Andersen, M. Akay, (Eds.), Replace, Repair, Restore, Relieve – Bridging Clinical and Engineering Solutions in Neurorehabilitation. Biosystems & Biorobotics, vol 7, Springer, Cham, 2014, pp. 263–270. DOI: https://doi.org/10.1007/978-3-319-08072-7_45
4. Nasri N, Orts-Escolano S, Cazorla M., An sEMG-Controlled 3D Game for Rehabilitation Therapies: Real-Time Time Hand Gesture Recognition Using Deep Learning Techniques in Sensors, vol 20, 2020, Article No.: 6451, DOI: https://doi.org/10.3390/s20226451
5. F. Farahanipad, H. R. Nambiappan, A. Jaiswal, M. Kyrarini, F. Makedon, HAND-REHA: dynamic hand gesture recognition for game-based wrist rehabilitation in PETRA '20: Proceedings of the 13th ACM International Conference on PErvasive Technologies Related to Assistive Environments, 2020, Article No.: 17, pp. 1–9, DOI: https://doi.org/10.1145/3389189.3392608
6. Q. Wang, X. Wang, EMG-based Hand Gesture Recognition by Deep Time-frequency Learning for Assisted Living & Rehabilitation in 11th IEEE Annual Ubiquitous Computing, Electronics & Mobile Communication Conference (UEMCON), 2020, pp. 558-561, DOI: https://doi.org/10.1109/UEMCON51285.2020.9298181
7. Google AI Blog. On-device, real-time hand tracking with MediaPipe. (2019, August 19). URL: https://ai.googleblog.com/2019/08/on-device-real-time-hand-tracking-with.html
8. 2048. (n.d.). URL: https://play2048.co/

9. M. L. Daviglus, C. C. Bell, W. Berrettini, P. E. Bowen, E. S. C. Jr, N. J. Cox, J. M. Dunbar-Jacob, E. C. Granieri, G. Hunt, K. McGarry, D. Patel, A. L. Potosky, E. Sanders-Bush, D. Silberberg, M. Trevisan, NIH state-of-the-science conference statement: Preventing Alzheimer's disease and cognitive decline. NIH consensus and state-of-the-science statements, 27(4) (2010), 1–30
10. E. Carr, Hand Therapy- Upper Limb Unit, University Hospitals Coventry and Warwickshire, July 2010
11. D. Rempel, K. Willms, J. Anshel, W. Jaschinski, J. Sheedy, The effects of visual display distance on eye accommodation, head posture, and vision and neck symptoms in: Human Factors: The Journal of the Human Factors and Ergonomics Society, 2007, 49(5), pp. 830–838. DOI: https://doi.org/10.1518/001872007x230208



# Appendix B: Sample Code

# START SCREEN GUI

```
import cv2
import tkinter as tk
from tkinter import   messagebox,Toplevel
from PIL import ImageTk, Image
import threading
from style import *
import sqlite3
from classClient import classClient
from classServer import classServer
import os
# from broadcastingServer import client_list
# from broadcastingClient import server_list

# database connection
connection = sqlite3.connect('airhockey.db')
c = connection.cursor()

empty = ''
k = classClient(empty,empty)
s = classServer(empty,empty)

class GUI:
    def __init__(self):
        self.gui_server_name = ""
        self.gui_client_name = ""
        self.sampleNames = ["Joel", "Jonathan", "Sivaramakrishnan
Subramaniam"]
        self.top_level = None
        self.playerName = None
        self.frame = None

        self.window = tk.Tk()
        self.window.attributes("-fullscreen", True)  # Set the window to
fullscreen

        bg_image = Image.open("Resources/bgm.png")
        bg_image = bg_image.resize((self.window.winfo_screenwidth(),
self.window.winfo_screenheight()))
        self.bg_photo = ImageTk.PhotoImage(bg_image)
        self.bg_label = tk.Label(self.window, image=self.bg_photo)
        self.bg_label.place(x=0, y=0, relwidth=1, relheight=1)

        self.start_button = None
        self.highscore_button = None
        self.quit_button = None
        self.clientMode = None
        self.serverMode = None
        self.back_button = None
        self.textbox1 = None
        self.home_button=None

        self.go_home(1)
        #self.play_video()

        # Bind the Escape key to ask_quit function
        self.window.bind("<Escape>", lambda event: self.ask_quit())

        # Start the tkinter event loop
        self.window.mainloop()

    def create_widgets(self):
        self.start_button = tk.Button(self.frame,    text="Start Game",
command=self.start_game, **button_style)
        self.highscore_button = tk.Button(self.frame,text="  History  ",
command=self.view_match_history, **button_style)
        self.quit_button = tk.Button(self.frame, text="Quit Game",
command=self.ask_quit, **qbutton_style)
        self.clientMode = tk.Button(self.frame, text="Create Room",
command=self.create_room, **button_style)
        self.serverMode = tk.Button(self.frame, text="Join Room",
command=self.show_rooms, **button_style)
        home_image = Image.open("Resources/home.png")
        home_image = home_image.resize((45, 45))  # Adjust the size of
the image if needed
        # Create a PhotoImage from the loaded image
        self.home_icon = ImageTk.PhotoImage(home_image)  # Store a
reference to the PhotoImage
        # Create the home button
        self.back_button = tk.Button(self.window, image=self.home_icon,
command=lambda mode=2: self.go_home(2),  **home_button_style)

        self.textbox1 = tk.Entry(self.frame, **textbox_style)
        self.textbox1.insert(tk.END, "Player 1")

    def position_widgets(self):
        self.start_button.place(relx=0.416, rely=0.49,
anchor=tk.CENTER)
        self.highscore_button.place(relx=0.576, rely=0.49,
anchor=tk.CENTER)
        self.quit_button.place(relx=0.501, rely=0.65, anchor=tk.CENTER)
        self.textbox1.place(relx=0.501, rely=0.65, anchor=tk.CENTER)
        self.clientMode.place(relx=0.416, rely=0.49, anchor=tk.CENTER)
        self.serverMode.place(relx=0.576, rely=0.49,
anchor=tk.CENTER)
        self.back_button.place(relx=0.04, rely=0.052,
anchor=tk.CENTER)

    def go_home(self,mode):
        if mode==2:
            self.textbox1.destroy()
            self.clientMode.destroy()
            self.serverMode.destroy()
            self.back_button.destroy()

        self.create_widgets()
        self.position_widgets()

        self.textbox1.destroy()
        self.clientMode.destroy()
        self.serverMode.destroy()
        self.back_button.destroy()


    def start_game(self):

        self.start_button.destroy()
        self.highscore_button.destroy()
        self.quit_button.destroy()

        self.create_widgets()
        self.position_widgets()

        self.start_button.destroy()
        self.highscore_button.destroy()
        self.quit_button.destroy()


    def show_rooms(self):
        thread = threading.Thread(target=k.client)
        thread.daemon = True  # Set the thread as daemon
        thread.start()
        status="Select a Room to join"
        demo_array = {}
        self.playerName = self.textbox1.get()# player name stored in this
variable
        k.user_name = self.playerName
        roomWindow = tk.Toplevel(self.window)
        roomWindow.title("Available Rooms")
        roomWindow['background'] = "#222E50"
```

```
        roomWindow.geometry("550x400")
        buttons = []  # Initialize buttons list
        statusLabel = tk.Label(roomWindow, text=status,
                               **statusLabel_style)
        statusLabel.pack(fill=tk.X, padx=20, pady=5)
        def buttonClick(btn_name):
            k.gui_server_name = btn_name
            statusLabel["text"]="Sending a request to "+btn_name+"..."
            statusLabel["fg"]="black"
            statusLabel["bg"]="#FCCB06"
            nonlocal status
            status=btn_name

        # Create and display the buttons
        count = 0

        def recButtons():
            nonlocal count
            nonlocal demo_array
            #sampleNames=self.sampleNames
            demo_array = k.server_names # server_list().copy()

            if count > 0:
                for widget in roomWindow.winfo_children():
                    if widget!=statusLabel:
                        widget.destroy()  # Destroy all widgets before
re-creating them
                self.sampleNames=["NoobGamer69","Joel"]
            if len(demo_array) == 0:
                noRoomsLabel = tk.Label(roomWindow, text="No Rooms
Currently Online", bg="#E6F2FF", font=("Helvetica", 14))
                noRoomsLabel.pack(pady=50)
            else:
                for name in demo_array:
                    button = tk.Button(roomWindow, text=name,
                                       command=lambda btn_name=name:
buttonClick(btn_name),
                                       **liveButton_style)
                    button.pack(fill=tk.X, padx=20, pady=5)
                    buttons.append(button)
                if status!="Select a Room to join" and status not in
demo_array:
                    statusLabel["text"]=status+" went offline. Try again"
                    statusLabel["fg"]="white"
                    statusLabel["bg"]="red"
            count += 1
            roomWindow.after(2000, recButtons)

        recButtons()

    def create_room(self):
        thread = threading.Thread(target=s.server)
        thread.daemon = True  # Set the thread as daemon
        thread.start()
        status="Select a Player to accept"
        demo_array = {}
        self.playerName = self.textbox1.get()  # player name stored in
this variable
        s.user_name = self.playerName
        roomWindow = tk.Toplevel(self.window)
        roomWindow.title("Incoming Requests")
        roomWindow['background'] = "#222E50"
        roomWindow.geometry("550x400")
        buttons = []  # Initialize buttons list
        statusLabel = tk.Label(roomWindow, text=status,
                               **statusLabel_style)
        statusLabel.pack(fill=tk.X, padx=20, pady=5)
        def buttonClick(btn_name):
            s.gui_client_name= btn_name
            statusLabel["text"]="Accepting "+btn_name+"..."
            statusLabel["fg"]="black"
            statusLabel["bg"]="#FCCB06"
            nonlocal status
            status=btn_name

        # Create and display the buttons
        count = 0

        def recButtons():
            nonlocal count
            nonlocal demo_array

            #sampleNames=self.sampleNames
            demo_array = s.client_names # server_list().copy()

            if count > 0:
                for widget in roomWindow.winfo_children():
                    if widget!=statusLabel:
                        widget.destroy()  # Destroy all widgets before
re-creating them
            if len(demo_array) == 0:
                noRoomsLabel = tk.Label(roomWindow, text="Waiting for
Requests...", bg="#E6F2FF", font=("Helvetica", 14))
                noRoomsLabel.pack(pady=50)
            else:
                for name in demo_array:
                    button = tk.Button(roomWindow, text=name,
                                       command=lambda btn_name=name:
buttonClick(btn_name),**liveButton_style)
                    button.pack(fill=tk.X, padx=20, pady=5)
                    buttons.append(button)
                if status!="Select a Player to accept" and status not in
demo_array:
                    statusLabel["text"]=status+" went offline. Try again"
                    statusLabel["fg"]="white"
                    statusLabel["bg"]="red"
            count += 1
            roomWindow.after(2000, recButtons)

        recButtons()

    def open_settings(self):
        pass

    def view_match_history(self):
        c.execute('SELECT * FROM scoreset ORDER BY rowid DESC
LIMIT 7')
        rows = c.fetchall()
        highscore_window = Toplevel(self.window)

        highscore_window.title("Match History")
        highscore_window.geometry("430x460")
        highscore_window.configure(bg="#002E63")  # Set background
color of the window

        for row in rows:
            frame = tk.Frame(highscore_window, bg="#D4E6F1",
padx=10, pady=10)
            time_label = tk.Label(frame, text=str(row[4]), fg="#3498DB",
bg="#D4E6F1",
                                  font=("Helvetica", 12, "bold"))
            score_label = tk.Label(frame, text=str(row[2])+" - "+str(row[3]),
**statusLabel_style)
            player1_label = tk.Label(frame, text=str(row[0]), fg="red",
bg="#D4E6F1",
                                     font=("Helvetica", 12,"bold"))
            vs_label = tk.Label(frame, text= " VS ", fg="#2ECC71",
bg="#D4E6F1",
                                     font=("Helvetica", 12,"bold"))
            player2_label = tk.Label(frame, text= str(row[1]), fg="blue",
bg="#D4E6F1",
                                     font=("Helvetica", 12,"bold"))
            frame.pack(fill=tk.X, padx=10, pady=5)
            time_label.pack(side=tk.LEFT)
            score_label.pack(side=tk.LEFT, padx=10)
            player1_label.pack(side=tk.LEFT)
            vs_label.pack(side=tk.LEFT)
            player2_label.pack(side=tk.LEFT)
```

```
    def play_video(self):
        ret, frame = self.cap.read()
        if ret:
            # Retrieve screen resolution
            screen_width = self.window.winfo_screenwidth()
            screen_height = self.window.winfo_screenheight()

            # Resize video frame to match screen resolution
            frame = cv2.resize(frame, (screen_width, screen_height))
            frame = cv2.cvtColor(frame, cv2.COLOR_BGR2RGB)
            image = ImageTk.PhotoImage(Image.fromarray(frame))
            self.canvas.create_image(0, 0, anchor=tk.NW, image=image)
            self.canvas.image = image
            self.window.after(30, self.play_video)
        else:
            self.cap.set(cv2.CAP_PROP_POS_FRAMES, 0)
            self.play_video()

    def ask_quit(self):
        if messagebox.askokcancel("Quit", "Are you sure you want to
quit?"):
            self.window.destroy()

gui = GUI()
```

## CLIENT PROGRAM

```
import cv2
import cvzone
from cvzone.HandTrackingModule import HandDetector
import numpy as np
import socket
import threading
import time
import pygame
import sqlite3

pygame.mixer.init()

# Locks for synchronization
client_hand_coordinates_lock = threading.Lock()
server_hand_coordinates_lock = threading.Lock()

client_hand_coordinates = [500, 300, 0, 0]
server_hand_coordinates = [0, 0, 0, 0]
ball_coordinates = [615, 335]
name0 = "levi"
name1 = "ackerman"
left = False
count_start = 0
data = ''
data1 = ''
data2 = ''
sound_bat = 0
sound_wall = 0
sound_goal = 0
sync = True
turn = False
olay = False
score = [0, 0]
restart = 0


def handle_server_communication(client_socket):
    global client_hand_coordinates
    global server_hand_coordinates
    global ball_coordinates
    global left
    global data
    global data1
    global data2
    global sync
    global score
    global sound_goal
    global sound_bat
    global sound_wall
    global restart

    while True:
        # Receive hand coordinates from the server
        if sync:
            client_coordinates = client_hand_coordinates

            # Send the client hand coordinates back to the server
            data = ''
            data1 = ''
            data2 = ''




     data = ','.join(map(str, client_coordinates))
            data = data + ','
            data1 = ','.join(map(str, ball_coordinates))
            data = data + data1 + ','
            data2 = ','.join(map(str, score))
            data = data + data2 + ',' + str(sound_bat) + ',' + str(sound_wall)
+ ',' + str(sound_goal)

            client_socket.sendall(data.encode())
            sync = False

        data = client_socket.recv(1024).decode()
        if not data:
            break

        left = True
        x, y, w, h, restart = map(int, data.split(','))
        if restart == 1:
            print('restart recieved is ' + str(restart))
        with server_hand_coordinates_lock:
            server_hand_coordinates = (x, y, w, h)
        sync = True


def play_hitsound():
    hitsound = pygame.mixer.Sound("Resources/hit.wav")
    hitsound.play()


def play_goalsound():
    goalsound = pygame.mixer.Sound("Resources/goal.wav")
    goalsound.play()


def play_wallsound():
    wallsound = pygame.mixer.Sound("Resources/wall.wav")
    wallsound.play()


def start_game(client_socket):
    global left
    global count_start
    global server_hand_coordinates
    global client_hand_coordinates
    global ball_coordinates
    global turn
    global olay
    global score
    global sound_bat
    global sound_wall
    global sound_goal
    global execcnt
    global start_time
    global value1
    global value2
    global restart

    # database connection
    connection = sqlite3.connect('airhockey.db')
    c = connection.cursor()
    query1 = "insert into scoreset values(?, ?, ?, ?, ?)"
    execcnt = 0

    def close_window(event, x, y, flags, param):
        if event == cv2.EVENT_LBUTTONDOWN:
            cv2.destroyAllWindows()

    # Stopwatch variables
    start_time = time.time()

    # Set the dimensions of the vertical board
    desired_width = 1280
    desired_height = 720

    cap = cv2.VideoCapture(0)
    cap.set(3, desired_width)
    cap.set(4, desired_height)
    ses_end = False
```

```
    check = [0, 0]

    # Importing all images
    imgBackground = cv2.imread("Resources/VSboard.png")
    imgGameOver = cv2.imread("Resources/gameOver.png")
    imgBall = cv2.imread("Resources/Ball.png",
cv2.IMREAD_UNCHANGED)
    imgBat1 = cv2.imread("Resources/Hbat1.png",
cv2.IMREAD_UNCHANGED)
    imgBat2 = cv2.imread("Resources/Hbat1.png",
cv2.IMREAD_UNCHANGED)

    # Hand Detector
    detector = HandDetector(detectionCon=0.8, maxHands=1)

    # Variables
    ballPos = [615, 335]
    speedX = 0
    speedY = 0
    gameOver = False
    rx = 0
    ry = 0
    rw = 0
    rh = 0
    mptW = desired_width // 2
    mptH = desired_height // 2
    count_bounce = 2

    while True:
        activeHands = False
        right = False
        _, img = cap.read()
        img = cv2.flip(img, 1)

        hands, img = detector.findHands(img, flipType=False)

        imgBackground = cv2.resize(imgBackground, (desired_width,
desired_height))
        img = cv2.resize(img, (desired_width, desired_height))
        img = cv2.addWeighted(img, 0.2, imgBackground, 0.8, 0)

        if hands:
            activeHands = True
            for hand in hands:
                x, y, w, h = hand['bbox']
                rh, rw, _ = imgBat1.shape
                tempX = x - rw // 2
                tempY = y - rh // 2

                rx = tempX
                ry = tempY

                ry = np.clip(ry, 360, 694)
                rx = np.clip(rx, 220, 931)
                client_hand_coordinates = (rx, ry, rw, rh)

                if hand['type'] == "Right":
                    right = True

        """if check != score or check == score == [0, 0]:
            check = score
            count_bounce = 2"""

        if rx != 0 and ry != 0:
            client_hand_coordinates = (rx, ry, rw, rh)

            img = cvzone.overlayPNG(img, imgBat1, (rx, ry))

            if count_bounce != 1:
                if rx <= ballPos[0] <= rx + rw and ry - 50 <= ballPos[1] <= ry:
                    speedY = -speedY
                    play_hitsound()
                    count_bounce = 1
                    sound_bat = 1
                    sound_wall = 0
                    sound_goal = 0
                    if count_start == 0:
                        speedX = 30
                        speedY = -30
                        turn = True
                        count_start = 1
                elif rx < ballPos[0] <= rx + rw and ry <= ballPos[1] <= ry + rh:
                    speedY = -speedY
                    play_hitsound()
                    sound_bat = 1
                    sound_wall = 0
                    sound_goal = 0
                    count_bounce = 1

        if left:
            lx, ly, lw, lh = server_hand_coordinates

            ly = ly - 2 * (ly - mptH) - lh
            if lx >= mptW:
                lx = lx - 2 * (lx - mptW) - lw
            elif lx < mptW:
                lx = lx + 2 * (mptW - lx) - lw

            img = cvzone.overlayPNG(img, imgBat2, (lx, ly))

            if count_bounce != 0:
                if lx <= ballPos[0] <= lx + lw and ly >= ballPos[1] >= ly - 50:
                    speedY = -speedY
                    play_hitsound()
                    sound_bat = 1
                    sound_wall = 0
                    sound_goal = 0
                    if count_start == 0:
                        speedX = -30
                        speedY = 30
                        count_start = 1
                    count_bounce = 0

                elif lx <= ballPos[0] <= lx + lw and ly >= ballPos[1] >= ly - lh:
                    speedY = -speedY
                    play_hitsound()
                    sound_bat = 1
                    sound_wall = 0
                    sound_goal = 0
                    count_bounce = 0
                    if count_start == 0:
                        speedX = -30
                        speedY = 30
                        count_start = 1

        cv2.putText(img, str(int(score[0])).zfill(1), (75, 148),
cv2.FONT_HERSHEY_COMPLEX, 1.5, (18.9, 9.7, 111.6), 3)
        cv2.putText(img, str(score[1]).zfill(1), (1133, 148),
cv2.FONT_HERSHEY_COMPLEX, 1.5, (18.9, 9.7, 111.6), 3)
        cv2.putText(img, str(name1).zfill(1), (1123, 78),
cv2.FONT_HERSHEY_SIMPLEX, 1.5, (18.9, 9.7, 111.6), 3)
        cv2.putText(img, str(name0).zfill(1), (75, 78),
cv2.FONT_HERSHEY_SIMPLEX, 1.5, (18.9, 9.7, 111.6), 3)

        if (ballPos[0] < 780 and ballPos[0] > 508 and ballPos[1] <= 11):
            ses_end = True
            play_goalsound()
            sound_goal = 1
            sound_bat = 0
            sound_wall = 0
            score[0] = score[0] + 1
            print("you score")
        elif (ballPos[0] < 780 and ballPos[0] > 508 and ballPos[1] >=
670):
            ses_end = True
            play_goalsound()
            sound_goal = 1
            sound_bat = 0
```

```
            sound_wall = 0
            score[1] = score[1] + 1
            print("opp score")

        if ses_end:
            ballPos = [615, 335]
            speedX = 0
            speedY = 0
            ses_end = False
            count_start = 0
            count_bounce = 2

        if score[1] == 2 or score[0] == 2:
            gameOver = True

        if gameOver:
            img = imgGameOver
            count_start = 0
            count_bounce = 2
            cv2.putText(img, str(score[0]).zfill(2), (370, 360),
cv2.FONT_HERSHEY_COMPLEX, 2.5, (200, 0, 200), 5)
            cv2.putText(img, str(score[1]).zfill(2), (800, 360),
cv2.FONT_HERSHEY_COMPLEX, 2.5, (200, 0, 200), 5)
            cv2.putText(img, str(score[1]).zfill(2), (800, 360),
cv2.FONT_HERSHEY_COMPLEX, 2.5, (200, 0, 200), 5)
            cv2.putText(img, str(score[0]).zfill(2), (370, 360),
cv2.FONT_HERSHEY_COMPLEX, 2.5, (200, 0, 200), 5)
            if (score[0] > score[1]):
                cv2.putText(img, str(name0) + " wins".zfill(1), (566, 360),
cv2.FONT_HERSHEY_SIMPLEX, 1,
                            (31.3, 182, 194.1), 3)
            else:
                cv2.putText(img, str(name1) + " wins".zfill(1), (566, 360),
cv2.FONT_HERSHEY_SIMPLEX, 1,
                            (31.3, 182, 194.1), 3)

            if start_time is not None:
                elapsed_time = time.time() - start_time
                formatted_time = time.strftime("%H:%M:%S",
time.gmtime(elapsed_time))
                print("Elapsed Time:", formatted_time)
                start_time = None

            if execcnt == 0:
                c.execute(query1, (name0, name1, score[0], score[1],
formatted_time))
                connection.commit()
            execcnt += 1

        else:
            if ballPos[0] >= 1010 or ballPos[0] <= 230:
                speedX = -speedX
                play_wallsound()
                sound_wall = 1
                sound_bat = 0
                sound_goal = 0
                if olay == False:
                    olay = True
                count_bounce = 2

            if ballPos[1] >= 670 or ballPos[1] <= 10:
                speedY = -speedY
                play_wallsound()
                sound_wall = 1
                sound_bat = 0
                sound_goal = 0
                if olay == False:
                    olay = True
                count_bounce = 2

            ballPos[0] += speedX
            ballPos[1] += speedY

            ballPos[0] = np.clip(ballPos[0], 220, 1010)
            ballPos[1] = np.clip(ballPos[1], 0, 670)

            if (ballPos[0] < mptW):
                ball_coordinates[0] = ballPos[0] - 50 + 2 * (mptW -
ballPos[0])
            elif (ballPos[0] >= mptW):
                ball_coordinates[0] = ballPos[0] - 50 - 2 * (ballPos[0] -
mptW)
            if (ballPos[1] >= mptH):
                ball_coordinates[1] = ballPos[1] - 50 - 2 * (ballPos[1] - mptH)
            elif (ballPos[1] < mptH):
                ball_coordinates[1] = ballPos[1] - 50 + 2 * (mptH -
ballPos[1])

            img = cvzone.overlayPNG(img, imgBall, ballPos)

        cv2.imshow("Image", img)
        if restart == 1:
            print('Entered restart')
            ballPos = [100, 100]
            speedX = 15
            speedY = 15
            gameOver = False
            score = [0, 0]
            #imgGameOver = cv2.imread("Resources/gameOver.png")
            sound_bat=0
            sound_wall=0
            sound_goal=0
            start_time = time.time()  # Reset stopwatch
            print("Game Restarted")
            execcnt = 0
        key = cv2.waitKey(1)
        if key == ord('q'):
            break

        if cv2.getWindowProperty('Image', cv2.WND_PROP_VISIBLE) <
1:
            break

    cap.release()
    cv2.destroyAllWindows()


def client_program(username, oppname, ip):
    global name0
    global name1
    name0 = username
    name1 = oppname
    host = ip
    port = 2000

    client_socket = socket.socket()
    client_socket.connect((host, port))

    game_thread = threading.Thread(target=start_game,
args=(client_socket,))
    game_thread.start()

    server_comm_thread =
threading.Thread(target=handle_server_communication,
args=(client_socket,))
    server_comm_thread.start()

    game_thread.join()
    server_comm_thread.join()

    client_socket.close()
```

## SERVER

```
import cv2
import cvzone
from cvzone.HandTrackingModule import HandDetector
import numpy as np
import socket
import threading
import time
import sqlite3
import os
import pygame

count_start = 0
score = [0,0]
pygame.mixer.init()
client_hand_coordinates_lock = threading.Lock()
server_hand_coordinates_lock = threading.Lock()
client_hand_coordinates = [700, 500, 0, 0]
server_hand_coordinates = [500, 300, 0, 0]
ball_coordinates = [615, 335]
start = 0
restart = 0
countGame =0
right = False
sync = False
turn = False
data1 = ''
data = ''
olay = False
value1="KD"
value2="Joel"
sound_bat = 0
sound_wall = 0
sound_goal = 0
def handle_client_communication(conn):
    global data1
    global data
    global olay
    global client_hand_coordinates
    global server_hand_coordinates
    global ball_coordinates
    global right
    global sync
    global start
    global score
    global sound_bat
    global sound_wall
    global sound_goal
    global restart
    while True:
        # Receive hand coordinates from the client
        # print("in function")
        data = ''
        start = time.time()
        data = conn.recv(1024).decode()
        #print(time.time()-start)
        if data == '':
            print("no data received")
        # print("received")
        if not data:
            break

        right = True
        # Parse the received coordinates
        sync = True
        x, y, w, h, ball_coordinates[0],
ball_coordinates[1],score[1],score[0],sound_bat,sound_wall,sound_go
al = map(int, data.split(','))

        client_hand_coordinates = (x, y, w, h)
        if (sync):
            data = ''
            server_coordinates = server_hand_coordinates
            # Send the server hand coordinates back to the client
            if server_coordinates:
                server_data = ','.join(map(str, server_coordinates))

                #data1 = ','.join(map(str, ball_coordinates))
                server_data = server_data + ',' + str(restart)
                # server_data = server_data + data1
                if restart == 1:
                    print('restart sent is ' + str(restart))
                # print("Data sent: "+server_data+"\n")
                conn.sendall(server_data.encode())
                restart = 0
                sync = False


def start_game(client_socket):
    global turn
    global olay
    global server_hand_coordinates
    global client_hand_coordinates
    global right
    global count_start
    global ball_coordinates
    global sound_bat
    global sound_wall
    global sound_goal
    global restart
    global execcnt
    global start_time
    global query1
    global countGame
    global score


    # database connection
    connection = sqlite3.connect('airhockey.db')
    c = connection.cursor()
    query1 = "insert into scoreset values(?, ?, ?, ?, ?)"
    execcnt = 0

    def close_window(event, x, y, flags, param):
        if event == cv2.EVENT_LBUTTONDOWN:
            cv2.destroyAllWindows()

    # Stopwatch variables
    start_time = time.time()

    # Set the dimensions of the vertical board
    desired_width = 1280
    desired_height = 720
    cap = cv2.VideoCapture(0)
    cap.set(3, desired_width)
    cap.set(4, desired_height)

    # Importing all images
    imgBackground = cv2.imread("Resources/VSBoard.png")
    imgGameOver = cv2.imread("Resources/gameOver.png")
    imgBall = cv2.imread("Resources/own_ball.png",
cv2.IMREAD_UNCHANGED)
    imgBat1 = cv2.imread("Resources/Hbat1.png",
cv2.IMREAD_UNCHANGED)
    imgAvatar = cv2.imread("Resources/AVATAR.png",
cv2.IMREAD_UNCHANGED)
    imgBat2 = cv2.imread("Resources/Hbat1.png",
cv2.IMREAD_UNCHANGED)

    # Hand Detector
    detector = HandDetector(detectionCon=0.8, maxHands=1)

    # Variables
    ballPos = [615, 335]
```

```
    speedX = 0
    speedY = 0
    gameOver = False
    ses_end = False
    global score
    lx = 0
    ly = 0
    lw = 0
    lh = 0
    count_bounce = 2
    while True:

        # Access and process hand coordinates for each connected
client
        # for client_socket, coordinates in
client_hand_coordinates.items():
        # Process hand coordinates for the client
        # Example: Access individual coordinates
        rx, ry, rw, rh = client_hand_coordinates
        activeHands = False
        left = False
        _, img = cap.read()
        img = cv2.flip(img, 1)
        # imgRaw = img.copy()
        hands, img = detector.findHands(img, flipType=False)  # with
draw
        # Rest of the code...

        # Resize the imgBackground and img to match the desired
dimensions
        imgBackground = cv2.resize(imgBackground, (desired_width,
desired_height))
        img = cv2.resize(img, (desired_width, desired_height))
        #score[0] = 0
        #score[1] = 1

        img = cv2.addWeighted(img, 0.2, imgBackground, 0.8, 0)
        if hands:
            for hand in hands:
                x, y, w, h = hand['bbox']
                lh, lw, _ = imgBat1.shape
                tempX = x - lw // 2
                tempY = y - lh // 2

                lx = tempX
                ly = tempY
                ly = ly + 300
                # ly = ly + 500
                # print(y1)
                ly = np.clip(ly, 350, 694)
                lx = np.clip(lx, 220, 931)
                server_hand_coordinates = (lx, ly, lw, lh)
                # if hand['type'] == "Left":
                #     left = True
                #     server_hand_coordinates = (lx, ly, lw, lh)

        # if  left or activeHands is False:
        if lx != 0 and ly != 0:

            # print("left  x: "+str(x)+"y: "+str(y)+"w: "+str(w)+"h: "+str(h))
            img = cvzone.overlayPNG(img, imgBat1, (lx, ly))
            if lx <= ballPos[0] <= lx + lw and ly - 50 <= ballPos[1] <= ly and
count_bounce != 0:
                speedY = -speedY
                count_bounce = 0
                if (count_start == 0):
                    # turn = True
                    # speedY = -20
                    # speedX = 20
                    count_start = 1

            elif lx <= ballPos[0] <= lx + lw and ly <= ballPos[1] <= ly + lh
and count_bounce != 0:
                speedY = -speedY
                count_bounce = 0
                if (count_start == 0):
                    # turn = True
                    # speedY = 20
                    # speedX = 20
                    count_start = 1

            # ballPos[0] += speedX
            # ballPos[1] += speedY
            # with server_hand_coordinates_lock:
            # server_hand_coordinates = (lx, ly, lw, lh)
            # print(server_hand_coordinates)

        if right:
            # print("left  x: "+str(x)+"y: "+str(y)+"w: "+str(w)+"h: "+str(h))
            mptW = desired_width // 2
            mptH = desired_height // 2
            ry = ry - 2 * (ry - mptH) - rh
            if rx >= mptW:
                rx = rx - 2 * (rx - mptW) - rw
                # print("first if")
            elif rx < mptW:
                rx = rx + 2 * (mptW - rx) - rw
                # print("Second if")

            img = cvzone.overlayPNG(img, imgBat2, (rx, ry))



        # Game Over
        cv2.putText(img, str(int(score[0])).zfill(1), (75, 148),
cv2.FONT_HERSHEY_COMPLEX, 1.5, (18.9, 9.7, 111.6), 3)
        cv2.putText(img, str(score[1]).zfill(1), (1133, 148),
cv2.FONT_HERSHEY_COMPLEX, 1.5, (18.9, 9.7, 111.6), 3)
        cv2.putText(img, str(value2).zfill(1), (1123, 78),
cv2.FONT_HERSHEY_SIMPLEX, 1.5, (18.9, 9.7, 111.6), 3)
        cv2.putText(img, str(value1).zfill(1), (75, 78),
cv2.FONT_HERSHEY_SIMPLEX, 1.5, (18.9, 9.7, 111.6), 3)

        if ses_end:
            # ballPos = [615, 335]
            speedX = 0
            speedY = 0
            ses_end = False
            count_start = 0
            count_bounce = 2
        #print(str(score[0]) + " " + str(score[1]))
        if (score[0] == 2 or score[1] == 2):
            gameOver = True

        if gameOver:

            img = imgGameOver


            cv2.putText(img, str(score[0]).zfill(2), (370, 360),
cv2.FONT_HERSHEY_COMPLEX, 2.5, (200, 0, 200), 5)
            cv2.putText(img, str(score[1]).zfill(2), (800, 360),
cv2.FONT_HERSHEY_COMPLEX, 2.5, (200, 0, 200), 5)

            # get elapsed time for database update
            if start_time is not None:
                elapsed_time = time.time() - start_time
                formatted_time = time.strftime("%H:%M:%S",
time.gmtime(elapsed_time))
                print("Elapsed Time: ", formatted_time)
                start_time = None  # Reset start_time to None

            # update database
            if execcnt == 0:
                c.execute(query1, (value1, value2, score[0], score[1],
formatted_time))
                connection.commit()
```

```
            execcnt = execcnt + 1

            if (score[0] > score[1]):
                cv2.putText(img, str(value1) + " wins".zfill(1), (566, 360),
cv2.FONT_HERSHEY_SIMPLEX, 1,(31.3, 182, 194.1), 3)

            else:
                cv2.putText(img, str(value2) + " wins".zfill(1), (566, 360),
cv2.FONT_HERSHEY_SIMPLEX, 1,(31.3, 182, 194.1), 3)
        # if gameOver:
        # img = imgGameOver
        # cv2.putText(img, str(score[1] + score[0]).zfill(2), (585, 360),
cv2.FONT_HERSHEY_COMPLEX,
        # 2.5, (200, 0, 200), 5)

        # If game not over move the ball
        else:
            ballPos = ball_coordinates



            ballPos[0] = np.clip(ballPos[0], 220, 1010)
            ballPos[1] = np.clip(ballPos[1], 0, 670)
            #print("X ball " + str(ball_coordinates[0]) + "y ball: " +
str(ball_coordinates[1]))
            img = cvzone.overlayPNG(img, imgBall, ballPos)

            # sound = [sound_bat, sound_wall, sound_goal]
            # soundPlay(sound_bat,sound_wall,sound_goal)
            # if sound_bat == 1:
            #     play_hitsound()
            # elif sound_wall == 1:
            #     play_wallsound()
            # elif sound_goal == 1:
            #     play_goalsound()
            '''
            thread1 = threading.Thread(target=soundPlay, args=sound)
            thread1.daemon = True
            thread1.start()
            '''



        # print(str(count_bounce) + "\n")
        # cv2.moveWindow("Image", 0, 0)
        cv2.imshow("Image", img)
        key = cv2.waitKey(1)
        if key == ord('r'):
            print('entered restart')
            ballPos = [615, 335]
            speedX = 0
            speedY = 0
            restart = 1
            count_bounce = 2
            count_start=0
            gameOver = False
            score = [0, 0]
            print('entered restart')
           # imgGameOver = cv2.imread("Resources/gameOver.png")
        if key == ord('q'):
            break
        if cv2.getWindowProperty('Image',cv2.WND_PROP_VISIBLE)<1:
            break

    cap.release()
    cv2.destroyAllWindows()


def server_program(username, oppname):
    global value1
    global value2
    host = socket.gethostname()
    port = 2000
    value1 = username
    value2 = oppname
    server_socket = socket.socket()
    server_socket.bind((host, port))
    server_socket.listen(2)

    print(host + " Server started. Waiting for connections...")

    while True:
        client_socket, addr = server_socket.accept()
        print("Connected with", addr)

        # Start the game in a separate thread
        game_thread = threading.Thread(target=start_game,
args=(client_socket,))
        game_thread.start()

        # Create a new thread to handle the client connection
        client_thread =
threading.Thread(target=handle_client_communication,
args=(client_socket,))
        client_thread.start()

        game_thread.join()
        client_thread.join()
        client_socket.close()
```

# Appendix C: CO-PO And CO-PSO Mapping

## COURSE OUTCOMES:

After completion of the course the student will be able to

| SL. NO | DESCRIPTION | Blooms' Taxonomy Level |
|---|---|---|
| CO1 | Identify technically and economically feasible problems (Cognitive Knowledge Level: Apply) | Level 3: Apply |
| CO2 | Identify and survey the relevant literature for getting exposed to related solutions and get familiarized with software development processes (Cognitive Knowledge Level: Apply) | Level 3: Apply |
| CO3 | Perform requirement analysis, identify design methodologies and develop adaptable & reusable solutions of minimal complexity by using modern tools & advanced programming techniques (Cognitive Knowledge Level: Apply) | Level 3: Apply |
| CO4 | Prepare technical report and deliver presentation (Cognitive Knowledge Level: Apply) | Level 3: Apply |
| CO5 | Apply engineering and management principles to achieve the goal of the project (Cognitive Knowledge Level: Apply) | Level 3: Apply |

## CO-PO AND CO-PSO MAPPING

| | PO 1 | PO 2 | PO 3 | PO 4 | PO 5 | PO 6 | PO 7 | PO 8 | PO 9 | PO 10 | PO 11 | PO 12 | PSO 1 | PSO 2 | PSO3 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| CO1 | 3 | 3 | 3 | 3 | | 2 | 2 | 3 | 2 | 2 | 2 | 3 | 2 | 2 | 2 |
| CO2 | 3 | 3 | 3 | 3 | 3 | 2 | | 3 | 2 | 3 | 2 | 3 | 2 | 2 | 2 |
| CO3 | 3 | 3 | 3 | 3 | 3 | 2 | 2 | 3 | 2 | 2 | 2 | 3 | | | 2 |
| CO4 | 2 | 3 | 2 | 2 | 2 | | | 3 | 3 | 3 | 2 | 3 | 2 | 2 | 2 |
| CO5 | 3 | 3 | 3 | 2 | 2 | 2 | 2 | 3 | 2 | | 2 | 3 | 2 | 2 | 2 |

3/2/1: high/medium/low

## JUSTIFICATIONS FOR CO-PO MAPPING

| MAPPING | LOW/ MEDIUM/ HIGH | JUSTIFICATION |
|---|---|---|
| 100003/CS622T.1-PO1 | **HIGH** | Identify technically and economically feasible problems by applying the knowledge of mathematics, science, engineering fundamentals, and an engineering specialization to the solution of complex engineering problems. |
| 100003/CS622T.1-PO2 | **HIGH** | Identify technically and economically feasible problems by analysing complex engineering problems reaching substantiated conclusions using first principles of mathematics. |
| 100003/CS622T.1-PO3 | **HIGH** | Design solutions for complex engineering problems by identifying technically and economically feasible problems. |
| 100003/CS622T.1-PO4 | **HIGH** | Identify technically and economically feasible problems by analysis and interpretation of data. |
| 100003/CS622T.1-PO6 | **MEDIUM** | Responsibilities relevant to the professional engineering practice by identifying the problem. |
| 100003/CS622T.1-PO7 | **MEDIUM** | Identify technically and economically feasible problems by understanding the impact of the professional engineering solutions. |
| 100003/CS622T.1-PO8 | **HIGH** | Apply ethical principles and commit to professional ethics to identify technically and economically feasible problems. |
| 100003/CS622T.1-PO9 | **MEDIUM** | Identify technically and economically feasible problems by working as a team. |
| 100003/CS622T.1-PO10 | **MEDIUM** | Communicate effectively with the engineering community by identifying technically and economically feasible problems. |
| 100003/CS622T.1-P011 | **MEDIUM** | Demonstrate knowledge and understanding of engineering and management principles by selecting the technically and economically feasible problems. |
| 100003/CS622T.1-PO12 | **HIGH** | Identify technically and economically feasible problems for long term learning. |
| 100003/CS622T.1-PSO1 | **MEDIUM** | Ability to identify, analyze and design solutions to identify technically and economically feasible problems. |
| 100003/CS622T.1-PSO2 | **MEDIUM** | By designing algorithms and applying standard practices in software project development and Identifying technically and economically feasible problems. |
| 100003/CS622T.1-PSO3 | **MEDIUM** | Fundamentals of computer science in competitive research can be applied to Identify technically and economically feasible problems. |
| 100003/CS622T.2-PO1 | **HIGH** | Identify and survey the relevant by applying the knowledge of mathematics, science, engineering fundamentals. |

| | | |
|---|---|---|
| 100003/CS622T.2-PO2 | **HIGH** | Identify, formulate, review research literature, and analyze complex engineering problems get familiarized with software development processes. |
| 100003/CS622T.2-PO3 | **HIGH** | Design solutions for complex engineering problems and design based on the relevant literature. |
| 100003/CS622T.2-PO4 | **HIGH** | Use research-based knowledge including design of experiments based on relevant literature. |
| 100003/CS622T.2-PO5 | **HIGH** | Identify and survey the relevant literature for getting exposed to related solutions and get familiarized with software development processes by using modern tools. |
| 100003/CS622T.2-PO6 | **MEDIUM** | Create, select, and apply appropriate techniques, resources, by identifying and surveying the relevant literature. |
| 100003/CS622T.2-PO8 | **HIGH** | Apply ethical principles and commit to professional ethics based on the relevant literature. |
| 100003/CS622T.2-PO9 | **MEDIUM** | Identify and survey the relevant literature as a team. |
| 100003/CS622T.2-PO10 | **HIGH** | Identify and survey the relevant literature for a good communication to the engineering fraternity. |
| 100003/CS622T.2-PO11 | **MEDIUM** | Identify and survey the relevant literature to demonstrate knowledge and understanding of engineering and management principles. |
| 100003/CS622T.2-PO12 | **HIGH** | Identify and survey the relevant literature for independent and lifelong learning. |
| 100003/CS622T.2-PSO1 | **MEDIUM** | Design solutions for complex engineering problems by Identifying and survey the relevant literature. |
| 100003/CS622T.2-PSO2 | **MEDIUM** | Identify and survey the relevant literature for acquiring programming efficiency by designing algorithms and applying standard practices. |
| 100003/CS622T.2-PSO3 | **MEDIUM** | Identify and survey the relevant literature to apply the fundamentals of computer science in competitive research. |
| 100003/CS622T.3-PO1 | **HIGH** | Perform requirement analysis, identify design methodologies by using modern tools & advanced programming techniques and by applying the knowledge of mathematics, science, engineering fundamentals. |
| 100003/CS622T.3-PO2 | **HIGH** | Identify, formulate, review research literature for requirement analysis, identify design methodologies and develop adaptable & reusable solutions. |

| | | |
|---|---|---|
| 100003/CS622T.3-PO3 | **HIGH** | Design solutions for complex engineering problems and perform requirement analysis, identify design methodologies. |
| 100003/CS622T.3-PO4 | **HIGH** | Use research-based knowledge including design of experiments, analysis and interpretation of data, and synthesis of the information to provide valid conclusions. |
| 100003/CS622T.3-PO5 | **HIGH** | Create, select, and apply appropriate techniques, resources, and modern engineering and IT tools. |
| 100003/CS622T.3-PO6 | **MEDIUM** | Perform requirement analysis, identify design methodologies and assess societal, health, safety, legal, and cultural issues. |
| 100003/CS622T.3-PO7 | **MEDIUM** | Understand the impact of the professional engineering solutions in societal and environmental contexts and Perform requirement analysis, identify design methodologies and develop adaptable & reusable solutions. |
| 100003/CS622T.3-PO8 | **HIGH** | Perform requirement analysis, identify design methodologies and develop adaptable & reusable solutions by applying ethical principles and commit to professional ethics. |
| 100003/CS622T.3-PO9 | **MEDIUM** | Function effectively as an individual, and as a member or leader in teams, and in multidisciplinary settings. |
| 100003/CS622T.3-PO10 | **MEDIUM** | Communicate effectively with the engineering community and with society at large to perform requirement analysis, identify design methodologies. |
| 100003/CS622T.3-PO11 | **MEDIUM** | Demonstrate knowledge and understanding of engineering requirement analysis by identifying design methodologies. |
| 100003/CS622T.3-PO12 | **HIGH** | Recognize the need for, and have the preparation and ability to engage in independent and lifelong learning in the broadest context of technological change by analysis, identify design methodologies and develop adaptable & reusable solutions. |
| 100003/CS622T.3-PSO3 | **MEDIUM** | The ability to apply the fundamentals of computer science in competitive research and prior to that perform requirement analysis, identify design methodologies. |
| 100003/CS622T.4-PO1 | **MEDIUM** | Prepare technical report and deliver presentation by applying the knowledge of mathematics, science, engineering fundamentals, and an engineering specialization to the solution of complex engineering problems. |
| 100003/CS622T.4-PO2 | **HIGH** | Identify, formulate, review research literature, and analyze complex engineering problems by preparing technical report and deliver presentation. |

| | | |
|---|---|---|
| 100003/CS622T.4-PO3 | **MEDIUM** | Prepare Design solutions for complex engineering problems and create technical report and deliver presentation. |
| 100003/CS622T.4-PO4 | **MEDIUM** | Use research-based knowledge including design of experiments, analysis and interpretation of data, and synthesis of the information to provide valid conclusions and prepare technical report and deliver presentation. |
| 100003/CS622T.4-PO5 | **MEDIUM** | Create, select, and apply appropriate techniques, resources, and modern engineering and IT tools and Prepare technical report and deliver presentation. |
| 100003/CS622T.4-PO8 | **HIGH** | Prepare technical report and deliver presentation by applying ethical principles and commit to professional ethics and responsibilities and norms of the engineering practice. |
| 100003/CS622T.4-PO9 | **HIGH** | Prepare technical report and deliver presentation effectively as an individual, and as a member or leader in teams, and in multidisciplinary settings. |
| 100003/CS622T.4-PO10 | **HIGH** | Communicate effectively with the engineering community and with society at large by prepare technical report and deliver presentation. |
| 100003/CS622T.4-PO11 | **MEDIUM** | Demonstrate knowledge and understanding of engineering and management principles and apply these to one's own work by prepare technical report and deliver presentation. |
| 100003/CS622T.4-PO12 | **HIGH** | Recognize the need for, and have the preparation and ability to engage in independent and lifelong learning in the broadest context of technological change by prepare technical report and deliver presentation. |
| 100003/CS622T.4-PSO1 | **MEDIUM** | Prepare a technical report and deliver presentation to identify, analyze and design solutions for complex engineering problems in multidisciplinary areas. |
| 100003/CS622T.4-PSO2 | **MEDIUM** | To acquire programming efficiency by designing algorithms and applying standard practices in software project development and to prepare technical report and deliver presentation. |
| 100003/CS622T.4-PSO3 | **MEDIUM** | To apply the fundamentals of computer science in competitive research and to develop innovative products to meet the societal needs by preparing technical report and deliver presentation. |
| 100003/CS622T.5-PO1 | **HIGH** | Apply the knowledge of mathematics, science, engineering fundamentals, and an engineering specialization to the solution of complex engineering problems. |
| 100003/CS622T.5-PO2 | **HIGH** | Identify, formulate, review research literature, and analyze complex engineering problems by applying engineering and management principles to achieve the goal of the project. |

| | | |
|---|---|---|
| 100003/CS622T.5-PO3 | **HIGH** | Apply engineering and management principles to achieve the goal of the project and to design solutions for complex engineering problems and design system components or processes that meet the specified needs. |
| 100003/CS622T.5-PO4 | **MEDIUM** | Apply engineering and management principles to achieve the goal of the project and use research-based knowledge including design of experiments, analysis and interpretation of data, and synthesis of the information to provide valid conclusions. |
| 100003/CS622T.5-PO5 | **MEDIUM** | Create, select, and apply appropriate techniques, resources, and modern engineering and IT tools and to apply engineering and management principles to achieve the goal of the project. |
| 100003/CS622T.5-PO6 | **MEDIUM** | Apply reasoning informed by the contextual knowledge to assess societal, health, safety, legal, and cultural issues and the consequent responsibilities by applying engineering and management principles to achieve the goal of the project. |
| 100003/CS622T.5-PO7 | **MEDIUM** | Understand the impact of the professional engineering solutions in societal and environmental contexts, and apply engineering and management principles to achieve the goal of the project. |
| 100003/CS622T.5-PO8 | **HIGH** | Apply ethical principles and commit to professional ethics and responsibilities and norms of the engineering practice and to use the engineering and management principles to achieve the goal of the project. |
| 100003/CS622T.5-PO9 | **MEDIUM** | Function effectively as an individual, and as a member or leader in teams, and in multidisciplinary settings and to apply engineering and management principles to achieve the goal of the project. |
| 100003/CS622T.5-PO11 | **MEDIUM** | Demonstrate knowledge and understanding of engineering and management principles and apply these to one's own work, as a member and leader in a team. Manage projects in multidisciplinary environments and to apply engineering and management principles to achieve the goal of the project. |
| 100003/CS622T.5-PO12 | **HIGH** | Recognize the need for, and have the preparation and ability to engage in independent and lifelong learning in the broadest context of technological change and to apply engineering and management principles to achieve the goal of the project. |
| 100003/CS622T.5-PSO1 | **MEDIUM** | The ability to identify, analyze and design solutions for complex engineering problems in multidisciplinary areas. Apply engineering and management principles to achieve the goal of the project. |

| | | |
|---|---|---|
| 100003/CS622T.5-PSO2 | **MEDIUM** | The ability to acquire programming efficiency by designing algorithms and applying standard practices in software project development to deliver quality software products meeting the demands of the industry and to apply engineering and management principles to achieve the goal of the project. |
| 100003/CS622T.5-PSO3 | **MEDIUM** | The ability to apply the fundamentals of computer science in competitive research and to develop innovative products to meet the societal needs thereby evolving as an eminent researcher and entrepreneur and apply engineering and management principles to achieve the goal of the project. |