A Novel Approach to Improve the Virtual Reality Kayaking Experience

T. I. R. De Zoysa*, G. I. Deshapriya*, R. A. R. L. Ranasinghe*, R. D. P. M. Ranasinghe*, Prof. Dileeka Dias†, Dr. Kasun T. Hemachandra†
*Department of Electronic and Telecommunication Engineering,
University of Moratuwa, Sri Lanka
June 29, 2024

Abstract: This research paper presents the development and implementation of the Virtual Reality (VR) kayaking system. This system offers an immersive experience because of the 360° video surrounding and virtual river. To mask out the river part of the 360° video we used a machine learning algorithm. Then we combined the rest of the 360° video with the virtual river. We used the Unity game engine to develop the system. The system has three modes: competitive mode, free mode and engaged mode. In the competitive mode, the user will be able to compete with the bot players trained using Reinforcement Learning (RL). Free mode shows the 360° as the background. In the engaged mode, in the beginning player will hear an audio description of the kayaking location. Next, the player will be able to chat with the bot. Also, the system give haptic feedback from the platform that we will implement, to the virtual environment. This VR kayaking system is a solution for the lack of infrastructure for real-world kayaking and provides solutions for safety concerns. We can use this system in gaming cafes, and fitness centers and also we can use it to promote tourism. Also, this paper presents the overall project architecture including the Bluetooth communication.

Key Words: Virtual Reality (VR) ,Kayaking System ,360° Video , Reinforcement learning (RL), ReUnity Game Engine , Haptic Feedback, Bluetooth communication

1 Introduction

The integration of Virtual Reality (VR) with real world physical activities increases the significant potential in various fields such as fitness, skill training, entertainment etc. The objective of VR Kayaking which is novel application for this field, is recreation of physical experience in a virtual environment. This project aims to overcome the limitations of recent VR designs while integrating the outdoor physical experience by combining real time virtual river creation and advanced haptic feedback system. Our approach focuses on the VR kayaking simulator that addresses above limitations and providing a comfortable indoor kayaking.

The need of providing an alternative to outdoor kayaking for the various audience arises the motivation of the project. Although outdoor kayaking beneficial for physical health, it introduces challenges such as safety risks, environment limitations, and the requirements of specific weather conditions. Research, like [1], has demonstrated how VR kayaking is improving cognition, muscle strength, and postural balance of elderly individuals. However, there is still a gap in providing VR kayaking solution which has real time Haptic Feedback mechanism. Also, existing VR simulations fail to replicate some aspects fully, including physical power, balance, and coordination, elements that are important for maintaining health benefits in outdoor kayaking and lead to a less satisfying user experience. Therefore, This VR project addresses these issues by developing a kayaking platform that has above unrecognized features like recent development of ergometer design [2] and advanced a Haptic Feedback system [4] by mapping the platform motions using IMU sensors and encoders with the VR environment. Bluetooth was the main communication protocol for data transfer between the platform and VR environment. Not only that, we also mapped the two VR controllers with VR paddle. Unlike most of the existing VR systems with standard controllers or pre-defined environments, we identified the integration of a physical platform as one of the research gaps.

The development of VR applications has improved interactive and immersive experiences. How ever, there are considerable gaps in current VR kayaking applications such as, lack a of score system which has a leaderboard and reward mechanism and Machine Learning Opponents in VR kayaking races. The Current products focus mainly on the gaming environment without a ranking system while missing opportunities to dive into a competitive environment for players. For achieving this, we have developed an advanced scoring system that enhances the player experience, engagement, fostering competition while knowing each other. Also, current VR kayaking applications has introduced virtual races often relying on pre-coded bot players that lack adaptive intelligence instead of using machine learning (ML) agents. Due to this, the players cannot get the real kayaking racing experience in a Single Mode race. So, by integrating ML agents with human-like behavior, we enable more realistic and challenging races, as a result, we have designed diverse and VR kayaking maps readily. This feature promotes the Single Mode races between players.

In the recent, most of VR applications are entirely artificial. As a result the players cannot experience the real world feelings.So, for resolving the issue, we have combined the

VR environment with actual geographical reality. It includes the development of a 360° real video based VR kayaking environment. This method combines the real world video environment with virtual objects such as the river, rocks, aquatic animals, water particles etc, providing users with a realistic and interactive experience, as demonstrated in the paper [3]. Additionally, as explored by the paper [5], about the incorporation of 360° video technology, allows a proper alignments and transitions between real and virtual environments.

Since, the recent VR systems haven't the feature of real time conservation with a Agent while playing the game. So we have developed a system to chat with a bot for conservation about the game functionality. At the beginning, the player will hear an audio description of the kayaking location. After that, the player will be able to have a voice conversation with the system. The system is capable to respond the user referring to previous conversations.

In conclusion, the VR kayaking project represents a significant improvement in the field of VR applications. This project bridges the gap between traditional outdoor activities and existing virtual applications by providing a better solution by addressing the limitations of existing solutions and integrating current technologies. At last, this project introduces a novel way to research of virtual environment design as a major outcome.

2 Materials & Methodology

The methodology chapter explains how to complete the VR kayaking project. It starts by describing the project architecture, identifies the requirements needed, and discusses the system design in terms of integrating major components. Additionally, it mentions the materials and tools needed during the implementation process to provide an understanding of the project's technical details.

2.1 Project Architecture

The VR kayaking system consists of two main components.platform and loacal computer. Platform data, including orientation and wheel speed, transmits to an ESP32 using I2C communication. The ESP32 sends this data to the VR headset over Bluetooth. Communication between the Local Computer and the VR headset operates bidirectionally, allowing for interactive data exchanges, while other data flows are unidirectional to maintain efficiency. Figure 1 illustrates the system's architecture and component interactions.

The architecture of the project consists of several key components:

- **IMU:** Provides orientation data of the platform to ESP32 board via I2C.
- **Rotary Encoder:** Measures the speed of the wheel and sends this data to the ESP32.
- **ESP32:** Receives encoder data, and IMU data transfers it to the VR headset via Bluetooth.
- **Local Computer:**Video prepossessing part run in it.

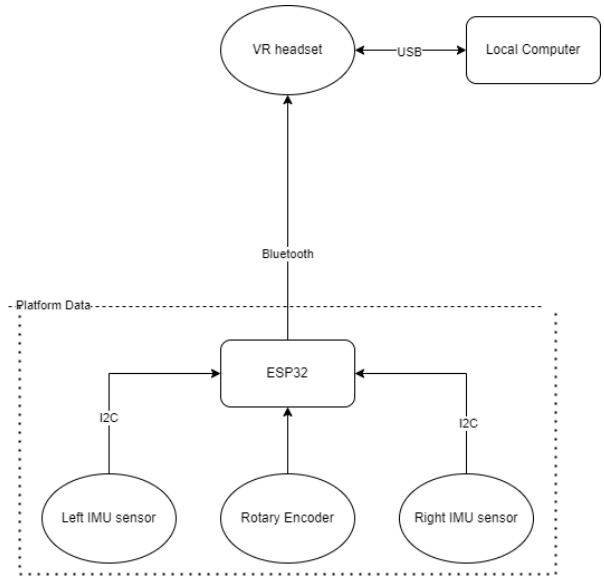

Fig. 1. System Architecture Diagram

- **VR Headset:** Visualizes the environment to the user based on the processed data from local computer.

2.2 Platform and Communication

Previous related projects on VR kayaking [7] failed to incorporate a physical kayaking platform, we use an accessorily designed platform that is intended to provide more realism and immersion to its end-user. The platform shall permit the simulation of natural movement in kayaking, for example, an ergometer mechanism providing resistance as if one is paddling through water-natural movements. With mounted sensors such as IMUs and rotary encoders, it will track the orientation and wheel speed with precision. This information feeds the VR system and ensures that all movements on the platform will be replicated in the virtual environment.

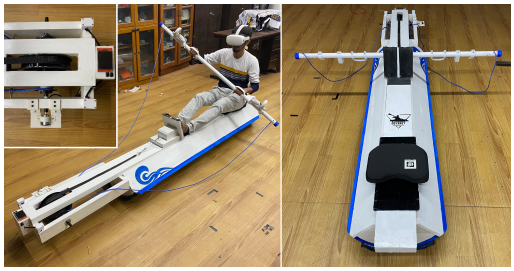

Fig. 2. Our Kayak platform: It includes an ergometer mechanism, which is made using pulleys and a wooden structure.

In our project, we want to map the physical kayaking platform motion to the unity side. Then we use IMU sensors and rotary encoder data to map. Therefore we want to transfer the sensor data and encoder data to the unity side. Then we use Bluetooth as a communication protocol because Bluetooth has some key advantages for our VR kayaking project. Like Low Power Consumption, Real-Time Data Transmission, Security, Ease Of implementation, Flexibility, and Scalability.

2.2.1 Platform data transmitted via Bluetooth

First, we create the BLE server on the ESP32, the device collects data from the sensors and the encoder processes it, and then formats it into packets. These packets, containing information such as orientation and speed, transmit to the VR headset via the established BLE connection.This real-time data transfer makes sure that the VR environment matches the user's movements.

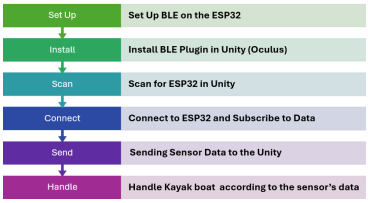

Fig. 3. Block diagram for mapping platform using IMU sensors and Rotary encoder via Bluetooth

Fig. 4. Our platform data packet: These packets contain information such as orientation and speed. The first three values represent the x, y, and z orientation data, and the final value represents the speed data.

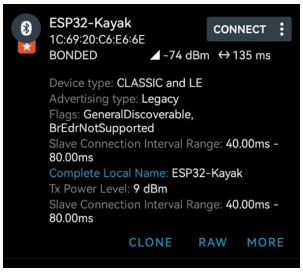

Fig. 5. Our BLE server information:latency is 135ms and slave connection interval range is 40ms-80ms

In Bluetooth communication, typical latency varies depending on factors such as connection interval settings and environmental conditions. For good Bluetooth communication, latency usually ranges from 4 to 20 milliseconds. Our VR kayaking system has real-time data transfer, then we also want good Bluetooth communication. In more applications, latency might be higher like it can range 20 to 50 milliseconds. That is also manageable for real-time data transmission. If our latency exceeds 50 milliseconds then it potentially impacts our user experience in the VR environment. So we must manage our latency at least 50 to 75 milliseconds.

In Bluetooth communication, the RSSI (Received Signal Strength Indicator) plays a very crucial role. It holds the signal power level of the received signal at our Bluetooth server. While we cannot visually observe the Bluetooth connection quality between the VR headset and the ESP32 used in our VR kayaking case, we need to track RSSI values to assess the overall connection quality and its reliability [8]. A normal Bluetooth signal, for that matter a strong Bluetooth signal normally measures in the area of 0dBm. Nevertheless, Bluetooth has a good signal strength of -30dBm also a bad signal strength of -100dB RSSI values. To minimize packet loss and ensure low latency, we need to keep the RSSI healthy and above -70 dB. We can detect potential issues such as interference and range limitations by monitoring changes in RSSI, which may affect the stability of real-time data transmission and the quality of the VR. In this case we use the client specifically the nrf connect app to get the RSSI values. After, we change the position of the client from the Server by moving it closer or further away.

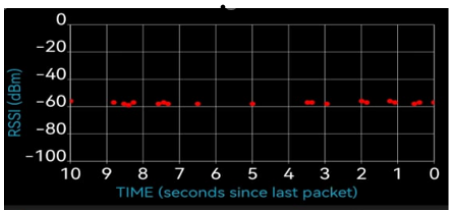

Fig. 6. RSSI variations for the 0–50 cm. It shows how the RSSI varies regularly between -60 dB to -40 dB, showing good strength with less attenuation .

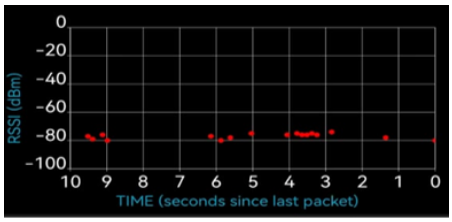

Fig. 7. RSSI variations for the 100–150 cm range.It shows how the RSSI varies regularly between -75 dB to -85 dB. The increased distance introduces moderate signal degradation.

2.2.2 Bluetooth Delay Calculation

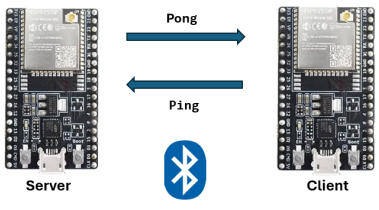

Fig. 8. Bluetooth delay analysis in the VR kayaking system conducted using two ESP32 modules, configured as a client and a server.

We calculated the delay in two phases: ping-pong communication and data packet transmission. In the ping-pong phase, the client sent a "ping" message to the server every 100 milliseconds, and the server responded with a "pong"

message.We considered it as packet loss if the RTT exceeded 40 milliseconds or if there was no response.

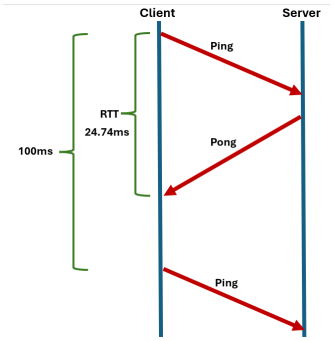

Fig. 9. Ping and Pong data packet communication:The one-way delay averaged 12.37 ms in this phase, which is lower than the usual latency and causes negligible overhead in this phase.

During the second phase, the client sent packets with values like "2.3, 4.5, 5.0, 30.0" to the server. From the analysis, we found that the average one-way delay for the transmission of data packets was about 23.99 ms.

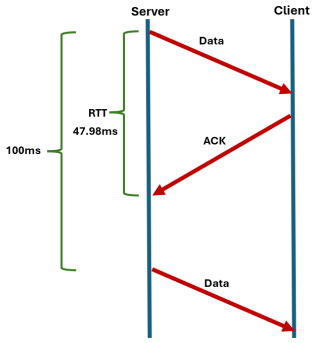

Fig. 10. Data packet communication:The one-way delay averaged 23.99 ms in this phase.This delay is higher than that obtained in the ping-pong phase due to the higher size of the payload and the extra processing.

We compared these findings with the literature on Bluetooth latencies concerning Virtual Reality applications. We found that previous works on latency requirements for interactive VR systems suggest the range of 20 to 50 ms for one-way transmission.; delays greater than this reduce the feeling of immersion and responsiveness. The one-way delay observed in the ping-pong phase was 12.37 ms and 23.99 ms during data transmission, which is within this acceptable range, hence the system is suitable for VR applications. Also, we calculated a time difference between data coming to the unity side and motion start using that sensor data time which value was equal to 0.632ms. Also we calculated a total delay of 24,622ms. That delay was in the acceptable range in VR applications. Furthermore, we confirm reliable communication through negligible packet loss and steady RTTs, which is an essential requirement to keep up the synchronization in VR environments. These findings form a great basis for Bluetooth as one of the viable communication methods in low-latency real-time VR systems.

3 Unity Development

In the Unity Development process, we have focused to brige the gap between modern user needs and the traditional VR systems in main key areas.

3.1 ML-Agents Training in VR Kayaking Races

3.1.1 Reinforcement Learning for Autonomous Kayak Racing

In virtual kayaking races, there are Machine Learning (ML) Agents trained by ML-Agents Toolkit in Unity as bot players in races. As demonstrated by [10], the Unity ML-Agents Toolkit is an open-source plugin to the Unity engine, facilitating reinforcement learning (RL) for Artificial Intelligence (AI) behavior training.

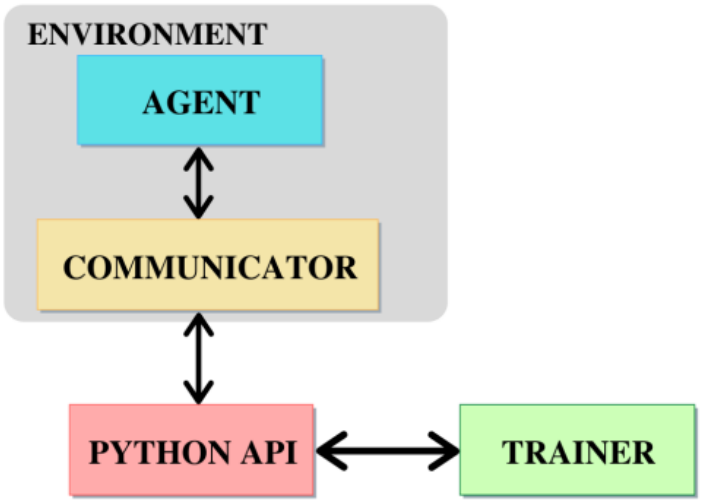

Fig. 11. There are 3 main components in the ML-agents toolkit: agent, environments and actions. The policy of each agents must access the neural network (NN) for sharing the experiences in the training. They use a communicator for connecting with the trainer through Python API.

The toolkit includes three key components such as agents, environments and actions as in Figure 11. ML agents interact with the environment to collect observations, execute actions, and based on their performance, receive rewards. The attached Agent script assigns rewards and observations while Behavior Parameter script behave as the brain of the agent and communicates rewards and observations to the trainer through python API. We designed virtual racing tracks as training environments for our VR kayaking system, similar to a kart racing setup [12]. So, we use a pre-trained model and again trained for a kayaking virtual setup. Actually, we trained 24 karts in a virtual environmnet as in Figure 12. Then, we attached the trained kart object at the bottom of the kayak object, so kayak agents move and rotate relative to the kart object. But kayak object also rotates itself according to the water system dynamics.

In reinforcement learning (RL) for kayak racing, agents (a_i) navigate the track over discrete time steps (t =

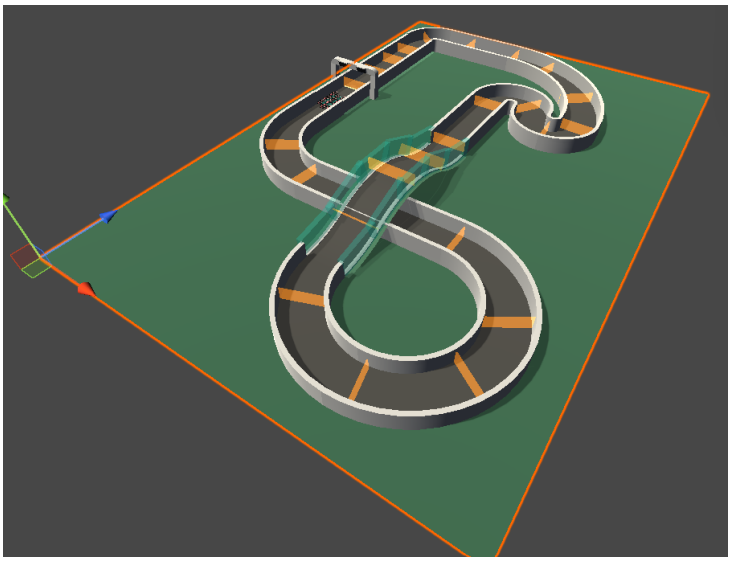

Fig. 12. The Virtual Training Environment which has various type of junctions, heights and checkpoints

$1, 2, ..., T$). An agent takes an action ($a_{i,t}$) such as acceleration, braking, or steering at each time step to transition to a new state (s_{t+1}) while providing a reward ($r_{i,t}$). The goal is to learn a policy $\pi_i(a_{i,t} \mid s_t)$ for evaluating state-value ($V_{\pi_i}(s_t)$)

$$V_{\pi_i}(s_t) = E_\pi \left[\sum_{k=0}^{T-t-1} \gamma^k r_{i,t+k} \mid s_t \right],$$

and action-value ($Q_{\pi_i}(s_t, a_{i,t})$) functions

$$Q_{\pi_i}(s_t, a_{i,t}) = E_\pi \left[\sum_{k=0}^{T-t-1} \gamma^k r_{i,t+k} \mid s_t, a_{i,t} \right].$$

to minimize time to the finish line while avoiding collisions and penalties. A policy is a probability distribution over actions, that maximizes cumulative rewards:

$$J(\pi_i) = E_\pi \left[\sum_{t=0}^{T-1} \gamma^t r_{i,t} \right].$$

This model uses the Multi-Agent Proximal Policy Optimization (MA-PPO) algorithm, which is an extension of PPO, to find the policy parameter θ_i that maximizes the cumulative reward over time.

3.1.2 Reward Mechanism of the RL Process

The following steps [12] explains how end the episodes and when add rewards and punishments.

- Agents start at the starting position, and the ML-agent begins listening to input and performing actions.
- When a agent Passes a checkpoint, earn a reward of 0.5/n, where n is the total number of checkpoints.
- Exceeding 30s to reach the next checkpoint results in a −1 punishment, and the agent respawns at the start position due to the end of episode.
- Reaching the final checkpoint adds a reward of 0.5, the agent respawns at the start position and ends of episode.
- Agents receive a small punishment of −0.001 to incentivize speed.

3.2 Sensor Feedback Mechanism

We have attached the two VR controllers at the end point of the physical paddle as in the paper [13] and assigned Left and Right Controller positional object to KayakPaddleController script. Then it calculates the midpoint between the left and right VR controllers to position the paddle's center, adding an initial offset stored at the start. For taking proper directional orientation of virtual paddle, it takes the vector between the left and right controllers. Then it calculates the rotation values from each axis by Vector product between the normalized difference vector and x, y and z unit vectors and apply this rotation to the paddle mid point. This process update in real-time for precise paddle movement.

We attach a IMU sensor at the midpoint of the physical kayak platform and fix the rotary encoder align with ergometer to touch the rotary wheels each other. Since, attached IMU sensors to the platform detect real kayak platform rotations, as well as the encoder measures the ergometer rotating speed by enabling Haptics Feedback [14], [15], they synchronize with Unity's virtual kayak rotating motions and top speed of the Kayak respectively. Also, our developed communication system utilizes sensor data received from an ESP32 via Bluetooth Low energy (BLE).

3.3 Mathematical Analysis and Modeling of Paddle Motion in Virtual Kayaking Simulation

3.3.1 Step 01: VR Kayak Paddle Mapping Using Two Controllers

In virtual reality (VR) simulations, precise mapping of user interactions is essential for immersion. This paper focuses on modeling a VR kayaking paddle controlled by two hand-held controllers, ensuring natural paddle positioning and orientation.

i) Paddle Position Calculation

The paddle's position is determined using the midpoint between the left and right hand controllers. At the **START** (1) and (2) are calculated, then at each frame **UPDATE** (3) to (7).

Midpoint Computation Let $\mathbf{P}_{LC}$ and $\mathbf{P}_{RC}$ represent the positions of the left and right controllers, respectively. The center's position of the hand controllers is given by:

$$\mathbf{P}_{\text{midC}} = \frac{\mathbf{P}_{LC} + \mathbf{P}_{RC}}{2} \tag{1}$$

To maintain a consistent offset from the initial position, an **initial offset** $\mathbf{O}_{\text{init}}$ is computed at the start:

$$\mathbf{O}_{\text{init}} = \mathbf{P}_{\text{midP}} - \mathbf{P}_{\text{midC}} \tag{2}$$

where $\mathbf{P}_{\text{midP}}$ represents the center's position of the virtual paddle.

Thus, after (1) and (2) are calculated, the following equations (3) to (7) are executed at each frame. Compute the center's position of the hand controllers same as in **START**:

$$\mathbf{P}_{\text{midC}} = \frac{\mathbf{P}_{LC} + \mathbf{P}_{RC}}{2} \tag{3}$$

and the updated virtual paddle position is:

$$\mathbf{P}_{\text{midP}} = \mathbf{P}_{\text{midC}} + \mathbf{O}_{\text{init}} \tag{4}$$

ii) Paddle Orientation Calculation

The paddle's orientation must align with the user's hand direction. This is achieved using **quaternion transformations**.

Hand Direction Vector The primary direction of the hand controllers is defined as the vector between the left and right controllers:

$$\mathbf{D}_{\text{C}} = \mathbf{P}_{RC} - \mathbf{P}_{LC} \tag{5}$$

The unit direction vector is:

$$\hat{\mathbf{D}}_{\text{C}} = \frac{\mathbf{D}_{\text{C}}}{|\mathbf{D}_{\text{C}}|} \tag{6}$$

This presents a mathematical model for mapping a VR kayak paddle using two controllers. By computing the midpoint for positioning and using quaternion-based alignment for orientation, we ensure smooth and realistic paddle interactions in VR.

3.3.2 Step 02: Virtual Kayaking Simulation based on the Virtual Paddling

Efficiently computing the movement direction of kayak paddle blades is crucial for achieving realistic virtual kayaking simulations. The function `CheckPaddleDirection()` determines the influence of paddle movements on the kayak's dynamics by analyzing the positional changes of the left and right paddle blades, computing forces and torques accordingly.

Let $P_L(t)$ and $P_R(t)$ represent the positions of the left and right virtual paddle blades at time t, respectively. The previous recorded paddle blade's positions relative to the center of mass of the kayak, are given by $P_{LC_K}(t-1)$ and $P_{RC_K}(t-1)$. The kayak's center of mass is denoted as $C_K(t)$, and the force application height offset is given by h_f.

i) Initialize the Function

At the **START**, the following are calculated for initialize the function.

$$P_{LC_K}(t-1) = P_L(t) - C_K(t)$$

$$P_{RC_K}(t-1) = P_R(t) - C_K(t)$$

Then, the global variables $WP_L(t)$ and $WP_R(t)$ which are represent the left paddle blade in water and right paddle blade in water respectively, are set using the function `IsInWaterPlane()`, where it returns the status of the blade is in the water or not is detected.

$$WP_L(t) = \text{IsInWaterPlane}(P_L(t)), \quad WP_R(t) = \text{IsInWaterPlane}(P_R(t))$$

Then, only after the **fixed time duration** has passed in **UPDATE**:

- `Direction Computation`, `Torque Computation` and `Force Computation` are executed, if left or right virtual paddle blade is inside the water (based on the status given by $WP_L(t)$ and $WP_R(t)$).
- Finally, `Water Detection and State Update` is updated.

ii) Direction Computation

If either paddle blade is submerged, its displacement vector relative to the kayak center is computed as:

$$D_L(t) = (P_L(t) - C_K(t)) - P_{LC_K}(t-1) \quad \text{if left paddle blade in water}$$

$$D_R(t) = (P_R(t) - C_K(t)) - P_{RC_K}(t-1) \quad \text{if right paddle blade in water}$$

The directional output is determined using the function `GetCardinalDirection()`, which maps the displacement vectors $D_L(t)$ or $D_R(t)$ to a predefined set of movement directions. The function is mathematically defined as:

$$G(D(t)) = (D_x, D_z)$$

where $G(D(t))$ is the computed cardinal direction for displacement $D(t)$. Here, the function extracts only the horizontal components x and z of the displacement vector, ignoring the vertical y component. Also, the returned `GetCardinalDirection()` direction is stored as the `outputDirection` globally.

$$O(t) = (O_x(t), O_y(t)) = G(D(t))$$

iii) Torque Computation

If the left paddle blade is submerged, the torque applied to the kayak is computed as:

$$T_L(t) = \tau(P_{LC_K}(t), P_{LC_K}(t-1), T_P)$$

where $\tau(a, b, c)$ represents a torque computation `AddTorqueKayak()` function based on the position difference around the specific torque position T_P (In this case T_P = Vector3(0,0,0)). Similarly, for the right paddle blade:

$$T_R(t) = \tau(P_{RC_K}(t), P_{RC_K}(t-1), T_P)$$

The function `AddTorqueKayak()` computes torque for turning the kayak. Given the current and previous paddle blade's positions relative to the kayak's center of mass, projected onto the horizontal plane:

$$P(t) = (P_x(t), 0, P_z(t)), \quad P(t-1) = (P_x(t-1), 0, P_z(t-1))$$

$$T_p = (T_x, 0, T_z)$$

Cross Product Calculation for Torque Computation

The function `CalculateCrossProduct()` is used to determine the perpendicular force vector contributing to the kayak's turning motion. Given three points:

- A = $P_{C_K}(t-1)$
- B = $P_{C_K}(t)$

- S = T_P

$$A = (A_x, A_y, A_z), \quad B = (B_x, B_y, B_z), \quad S = (S_x, S_y, S_z)$$

A direction vector is defined as:

$$V = B - A$$

A projection of $S - A$ onto V is computed as:

$$Proj_V(S - A) = \frac{(S - A) \cdot V}{V \cdot V} V$$

The perpendicular vector is:

$$V_\perp = (S - A) - Proj_V(S - A)$$

The cross-product is then computed as:

$$C = V \times V_\perp$$

which represents the torque-inducing vector applied to the kayak.

The following graph visualizes the cross-product calculation for torque computation. It illustrates the vectors $V = B - A$, $S - A$, the projected component $\text{Proj}_V(S - A)$, the perpendicular vector $V_\perp$, and the final torque-inducing cross product $C = V \times V_\perp$.

The function `CalculateCrossProduct()` calculates the cross product between the force vector ($P_c - P_p$) and perpendicular vector from T_p to the force vector. So, the cross-product computation for torque is given by:

$$U(t) = \lambda(P(t), P(t-1), T_p)$$

where U_y is extracted as the up-vector component. The turning power factor is calculated as:

$$T_f(t) = ||P(t) - P(t-1)|| + O_t$$

where O_t is a tuning offset. The applied torque to the kayak using **Unity's AddTorque function** (as an `Acceleration Force Mode`) is:

$$T(t) = kayakRigidbody.AddTorque(\frac{U_y(t)P_t}{T_f(t)^2})$$

where P_t is the turning power coefficient.

iv) Force Computation

The force application point on the kayak is computed as:

$$F_p(t) = C_K(t)$$

The function `AddForceKayak()` applies force to the kayak at a specific force position $F_p(t)$. The force direction is computed as:

$$F_d(t) = (-O_x(t), 0, -O_y(t))$$

where $O(t) = (O_x(t), O_y(t))$ represents the `outputDirection` vector. The applied force to the kayak using **Unity's AddForceAtPosition function** (as an `Acceleration Force Mode`) is given by:

$$F(t) = kayakRigidbody.AddForceAtPosition(F_d(t) \cdot P_e)$$

where P_e is the engine power coefficient. The force is applied at $F_p(t)$ using positional force application.

v) Water Detection and State Update

The function checks if each paddle blade is submerged in the water using:

$$WP_L(t) = \text{IsInWaterPlane}(P_L(t)), \quad WP_R(t) = \text{IsInWaterPlane}(P_R(t))$$

where `IsInWaterPlane()` is defined as:

$$WP(P(t)) = \begin{cases} 1, & P_y(t) < W_L \\ 0, & P_y(t) \geq W_L \end{cases}$$

where $WP(P(t))$ determines if the paddle blade $P(t)$ is below the water level W_L.

Then, the previous paddle blade's positions are updated for subsequent calculations as:

$$P_{LC_K}(t-1) = P_L(t) - C_K(t), \quad P_{RC_K}(t-1) = P_R(t) - C_K(t)$$

This function systematically calculates paddle motion, determines directionality, and applies forces and torques to simulate realistic kayaking dynamics. By utilizing displacement vectors, torque computations, and force applications, it ensures accurate movement representation in the virtual environment.

3.4 Voice Commands Recognition

The voice command system was fully integrated into a Unity-based VR kayaking application using a combination of the Meta XR All-in-One SDK and a BERT-based NLP model. The process involves both real-time voice input from the user and model inference for command classification.

3.4.1 Step 01: Speech to Text with Meta XR SDK

The Meta XR All-in-One SDK was installed in Unity to enable voice capture on compatible VR headsets (e.g., Meta Quest 2). This SDK provides APIs to access the microphone and convert voice to plain text using built-in Automatic Speech Recognition (ASR).

- The Unity script listens for user input using the Meta Voice SDK.
- When a user speaks, the SDK captures the audio and internally processes it through ASR.
- The output of this process is a plain text string of the user's spoken command.
- This text is passed to the NLP model for classification.

3.4.2 Step 02: Text Classification with BERT-base-uncased Model

The transcribed text is processed using a fine-tuned BERT-base-uncased NLP model on a custom dataset containing voice commands for classification to 4 type of predefined classes as shown in Table I. This step determines the specific command and triggers the corresponding function in the application.

- **BERT-base-uncased Model:** Used for robust text classification.
- **WordPiece Tokenizer:** Tokenizes the input text efficiently.

- **Optimizer:** AdamW (Adam with Weight Decay) ensures stable convergence.
- **Loss Function:** Cross Entropy Loss is used for multi-class classification tasks.

3.4.3 Command Classes and Functions

The classified text corresponds to one of the predefined command classes as shown in Table I. Each class is linked to a specific function within the system.

TABLE I
CLASSES & FUNCTIONS

Class	Function
`pause_menu`	Trigger the Pause menu
`main_menu`	Exit to Main menu
`align_arrow`	Align the boat along the arrow
`update`	Change the virtual environment

3.5 Score System

The LeaderBoard script manages the leaderboard functionality for a Unity game for displaying and updating high scores. The SetHighscoreEntry function updates the leaderboard by getting the high score table from PlayerPrefs, using the player's first name. Because the entire system identifies each users considering users First name field as the key identity (ID). There are common saving points for both leaderboard and reward system for all the players to add scores when achieving in the game. So, the function adds the current player's marks to their existing score of the player or creates a new entry if the player is not on the table. It uses a binary search algorithm to determine the correct position for new entries and sort the leaderboard. After updating, it stores the updated table back to PlayerPrefs. For allowing users to filter leaderboard entries by name, it includes search functionality. A special Congratulation object appears to celebrate players who achieve the first rank while highlighting the top three positions with visual elements like trophies and colors.

The RewardManager script tracks and unlocks predefined achievements for gaining additional scores. When a user completes an achievement reward manager stores its index in PlayerPrefs common current store point. Then the manager combines current and saved reward indexes and stored them to corresponding player. Meanwhile, it updates UI elements like buttons and text to unlock the corresponding rewards for collection. Once collected, the reward's score is added to the user's total leaderboard score. The reward item is then reset and locked again in future achievements.

Passing each checkpoint, winning given number of races, finish the race before a target time etc., are some of examples for gaining scores and the system shows them as Hints.

4 Kayaking Environment Background

The reason why we use both real and virtual worlds: always to keep the control of the player's movements in the game environment. We cannot do this using only real-world environments, since they rely on pre-recorded videos which cannot change their course of action dynamically with respect to the players' actions. For example, while thinking that the player paddles to the left, the environment should reflect this action. Using only real-world environments, requires recording various of videos in various directions, captures frames for every possible direction a player could face, and making transitions between them. This requires many computer resources and is an extremely complicated job. We can integrate the virtual river, where, with every action of the player, environmental aspects change dynamically for a smooth and realistic experience.

To realize this concept effectively, we followed a structured workflow that combined both real-world video data and dynamic virtual elements. The following workflow outlines the key stages involved in creating our VR Kayaking experience.

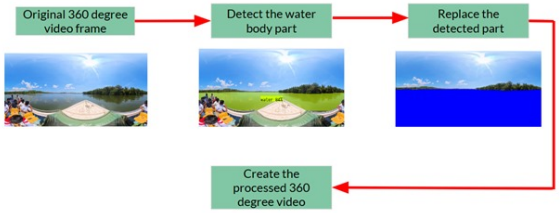

Fig. 13. Video Preprocessing Work Flow

4.0.1 Original 360 degree video frame(Extracting the Equirectangular Projection)

The 360-degree video I worked with was recorded using an equirectangular projection method. This approach takes in the entire spherical view all around the camera—from left to right (360 degrees across) and from top to bottom (180 degrees up and down)—which is then "flattened" into a rectangular image.

This flattening allows us to represent and process the entire environment as a 2D video frame whose width coincides with the entire horizontal rotation and whose height corresponds to the vertical angle. Utilizing equirectangular projection has the added benefits of being able to more easily use computer vision and machine learning tools (where a frame becomes a standard image) and still maintain the 360-degree view for VR.

Fig. 14. Equirectangular Projection

4.0.2 Splitting the Video into Frames

Rather than trying to process the entire video at once, IŽ019ve used PythonŽ019s OpenCV library to loop over a video and output individual frames. By working on one frame

at a time (sampling every few frames to save processing time), I could analyze and modify each image separately, making the entire process faster and more efficient.

4.0.3 Detecting the Water Region Using AI with Roboflow

For detecting water areas in each frame, I used a pre-trained computer vision model from Roboflow, which is a popular platform to deploy computer vision models via API. The model was trained to detect the ”water” class in images.

4.0.4 Methodology for Accessing the Roboflow Model

- First, create a free Roboflow account at roboflow.com.
- Get your API key on the Roboflow dashboard.
- Utilize the given model id, in this instance ”vr-kayaking/5,” to pass image data to the Roboflow Application Programming Interface (API).
- The model outputs predictions that contain the coordinates of the water regions identified.

For each frame:-

- Resized the image to the model's expected input size (512x512 pixels).
- Encoded the image in base64 format to send securely over the web.
- Sent the encoded frame to Roboflow2019s inference API.
- Received back the water region coordinates in the form of polygons.

Model Output:-

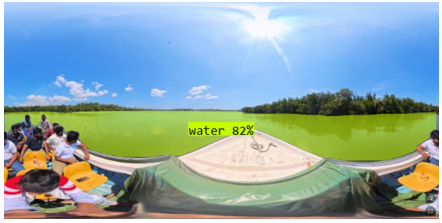

Fig. 15. Model Output image

Performance Analysis of the model:-


```
{
  "predictions": [
    {
      "x": 738,
      "y": 455,
      "width": 1474,
      "height": 172,
      "confidence": 0.819,
      "class": "water",
      "points": [
        {
          "x": 326.723,
          "y": 370.5
        },
        {
          "x": 324.406,
          "y": 371.658
        },
        {
          "x": 291.966,
          "y": 371.
```


Fig. 16. Model Output Values

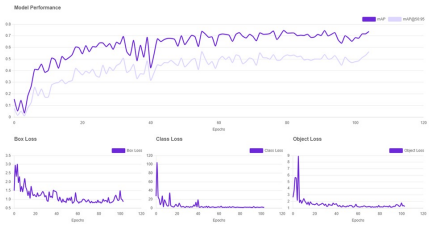

Fig. 17. Model Performance analysis

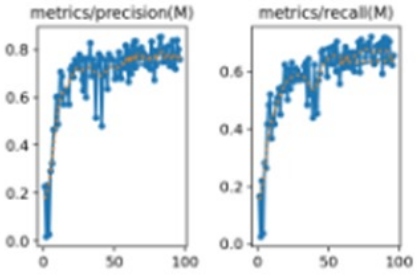

Fig. 18. Model precision and recall

Applying Water Mask and Custom Coloring

Once I obtained the water coordinates, I developed a mask emphasizing the specified regions. A semi-transparent blue overlay (RGBA: 0, 0, 255, 255) was applied to predominantly highlight the water pixels. This made the water stand out visually and helped separate it from the rest of the scene.

Replacing the Boat and Water Regions

My custom algorithm took this further by coloring the water and lower parts of the frame where water was detected. This step prepared the frames for integration with the virtual river and kayak in Unity. Although the frame still looked

natural, this colored overlay acted as a placeholder for the virtual elements we added later, allowing smooth combination of real and virtual content.

This algorithm is designed to detect and remove the water region (and kayak, if applicable) from a video frame, starting from the top-left corner and working down each column.

Step-by-Step Explanation:

1) **Start from the Top-Left Corner:**
   - Begin scanning the video frame column by column.
   - For each column, start from the top-most pixel and move downward.
2) **Detect the First Water Pixel in Each Column:**
   - While scanning down a column, check each pixel to see if it belongs to the water class (i.e., classified as part of the water body using a segmentation or classification model).
   - Once a pixel belonging to the water class is found, perform an additional check.
3) **Verify with Additional Pixels:**
   - After detecting the first water pixel in the column, check the next two pixels below it.
   - If all three consecutive pixels belong to the water class, it is considered a valid detection.
4) **Remove the Water Region in the Column:**
   - Once confirmed, remove (or mark for removal) all pixels starting from that first detected water pixel down to the bottom of the column.
   - This effectively removes the water region for that vertical slice of the frame.
5) **Repeat for All Columns:**
   - Continue the same process for every column across the width of the frame.

Saving the Processed Frames

Each processed frame—with the highlighted water regions—was saved as a high-quality PNG file with transparency. These files were stored in a folder to be used for video reconstruction.

Reconstructing the 360-Degree Video Using FFmpeg

After removing the water region and kayak from each frame of the 360-degree video, a sequence of processed image frames remains. To recreate the video from these individual frames, FFmpeg—a powerful command-line tool for handling multimedia files—was used.

4.0.5 The FFmpeg Command Explained

```
ffmpeg -framerate 30 -i frame_%05d.png -c:v libx264 -pix_fmt yuv420p output360.mp4
```

Fig. 19. Video reconstruction function

Here2019s what each part of the command means:

`-framerate 30`

- This sets the frame rate to 30 frames per second (fps).
- It tells FFmpeg that each second of video should be made up of 30 image frames.
- This ensures the video playback speed matches the original timing and provides smooth motion.
- For example, if you have 300 processed frames, the final video will be 10 seconds long (300 ÷ 30).

`-i frame_%05d.png`

- This specifies the input file pattern:
  - `frame_00001.png`, `frame_00002.png`, ..., `frame_99999.png`
  - `%05d` means the numbers are zero-padded to 5 digits.
- FFmpeg reads these image files in numerical order and treats each one as a video frame.

`-c:v libx264`

- This sets the video codec to `libx264`, which is a widely used H.264 encoder.
- H.264 is efficient, giving good quality at a relatively small file size.
- It ensures compatibility with most modern video players and platforms.

`-pix_fmt yuv420p`

- This sets the pixel format to `yuv420p`, which is the most compatible format for video playback.
- Many video players and web platforms require this format.
- Without this, some players (especially browsers or mobile devices) may not render the video properly.

`output360.mp4`

- This is the name of the output video file that will be created from your image sequence.
- It will be a smoothly playing, 30 fps 360-degree video, built from the water-removed and processed frames.

Sending the Video to Unity

The final video was imported into Unity, where it was mapped onto the inside of a 3D sphere. The user stands at the center of this sphere, allowing them to look around and experience the real-world environment in 360 degrees.

Along with this video sphere, a virtual river was created inside Unity based on the water masks from the video frames. This mix of real video and virtual objects created an immersive VR kayaking experience, combining reality and virtual design seamlessly.

Technologies and Tools Used

- **Python & OpenCV**: For extracting video frames and image processing.
- **Roboflow API**: For water region detection using a pre-trained machine learning model.
- **FFmpeg**: For video reconstruction from processed frames.
- **Unity 3D**: For creating the VR environment and combining real video with virtual objects.
- **Equirectangular Projection**: To handle and process the 360-degree video format.

This process was key to merging real-world 360-degree videos with interactive virtual environments. By detecting and highlighting water regions, we could integrate a virtual river and kayak, enhancing the realism and engagement of the VR kayaking simulation. Using Roboflow2019s AI model allowed for fast and accurate water detection without building a model from scratch. This integration of real and virtual elements created a unique and immersive VR experience.

5 Engaged Mode

When the player starts the system, system asks player to select one of two modes. The first one is competitive mode and the second one is engaged mode. When a player starts the engaged mode first he will hear an audio that describes the kayaking environment. The audio is 2-3 minutes in length. After that player will be able to chat with the system.

When a player uploads a video, systems askes to fill few details about the kayaking place. The first one is the name of the kayaking place. The second one is the city of the kayaking place. For example, if the player uploads a video of kayaking in Madu Ganga he should fill the kayaking place as “Madu Ganga” and the city as “Balapitiya”. I hosted a backend server as an AWS EC2 instance. The backend is coded in Python language. It has an API endpoint to get the audio description of the kayaking place. When system calls that post-API endpoint, system receives an audio file about the kayaking place.

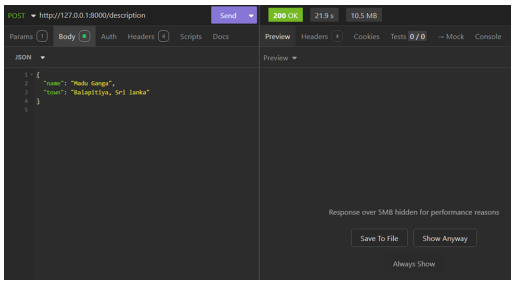

Fig. 20. This figure show the api endpoint call to get the audio file. Api call’s body has two parameters as shown. As the response it receives an audio file.

After that user can have a voice chat with the system. For that, I used Whisper Ai to transcribe the voice of the player. In the related program, when the user starts to speak program will start to listen. When the user is silent for more than two seconds program will stop listening and start transcribing the audio file. After transcribing the audio file system makes an API call for the backend(/chat) with the transcript. That backend API endpoint is integrated with Google Gemini LLM. It will return a response according to the user’s voice transcript. When integrating an LLM to the backend my first consideration was chatgpt API. But it doesn’t have a free tire or a monthly subscription. Its payment method is “pay as you go”. So it is difficult to keep a reliable subscription because if players use the chat more, the account balance will reach zero. Even though we could get a notification when the balance is close to zero dealing with it is somewhat difficult. So I decided to use Google Gemini LLM API. It is free and only need to get an API key to work with. Here I used a special method when integrating. Normally when calling Gemini API it doesn’t have a memory of our previous API calls. This is a problem when a player chats with the system, the system doesn’t have a memory of the previous chats. There is an option to start a chat when integrating Gemini API. I used that method. So the system is capable to refer previous chats when replying to the current prompt of the player.

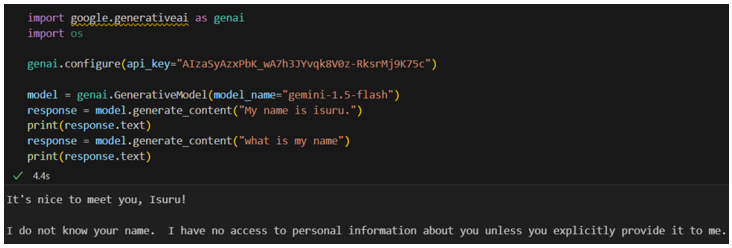

Fig. 21. This is without using the chat method when calling Gemini API. We can see that after telling LLM my name, it can’t remember it in the next conversation.

Fig. 22. This is with using the chat method when calling Gemini API. We can see that after telling LLM my name, it can remember it in the next conversation.

This engaged mode is useful because the player will be able to talk with someone when kayaking. He can have private conversations with the system. Players can use this to get opinions from the LLM about their problems. Since the Gemini-1.5-flash method has a very good cognition ability this is very useful.

6 Results

6.1 BLE Communication

We measured the communication delay between two ESP32 devices using BLE. The below histograms show Ping and Pong data delay, and Data packet delay. The results on Ping and Pong delay are shown as Averaging 12.37ms of delay, while the delay of the data packet, shown averaged at 23.99ms. These results prove that the BLE communication delay stays within the acceptable range and is suitable for real-time VR applications.

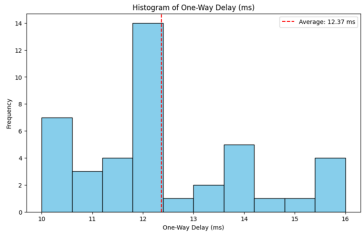

Fig. 23. Histogram for Ping and Pong data delay

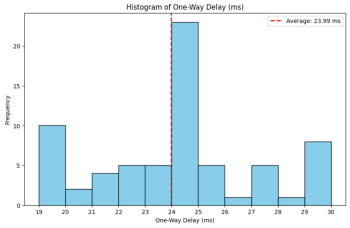

Fig. 24. Histogram for data packet communication

6.2 Unity Development

6.2.1 Architecture of the output of the Unity Application

We have used Unity software for the development of VR kayaking system application. The final output system follows the functional flow as the following given architecture block diagram. We have developed a user interface (UI) for navigating between scenes and each component. The Unity system incorporates four scenes as shown in the architecture diagram. The ”Intro” scene shows the starting loading window for the kayaking system, followed by the ”Main Menu” scene, which afford main entry points of the functionalities.

Single Player Mode

In this menu, users can navigate ”Play” to choose between Single Player or Multiplayer modes. In the “Single Player” panel, the ”Settings” tab allows adjustments to game-related features, including game difficulty (Easy, Normal, Hard), music volume, Racing Mode (On, Off), Haptics Feedback (On, Off), the number of bot players, and kayak color customization. If users enable the Racing Mode, users can participate in virtual kayaking races which have Machine Learning (ML) as the opponent. If users disable the Racing Mode, users can play in Free Mode. So, this mode renders the

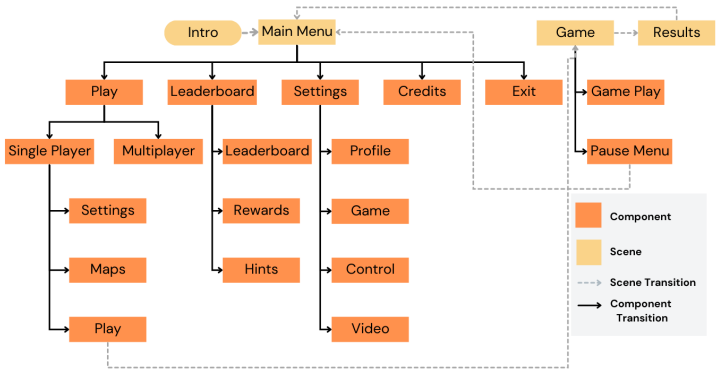

Fig. 25. This architecture shows the final result of Unity Application and mainly focuses on the UI design, because we can explain properly the output functionality by navigation though the UI

360 videos as the background on a Unity sphere. For making only the environment beyond the water visible and others invisible, we already preprocess these videos with image processing techniques, then remove water and kayak visuals. The “Maps” tab shows the virtual kayaking tracks or 360-degree video environments according to the chosen Racing Mode.

By clicking ”Play” users navigate to the ”Game” scene, where the kayaking gameplay begins.

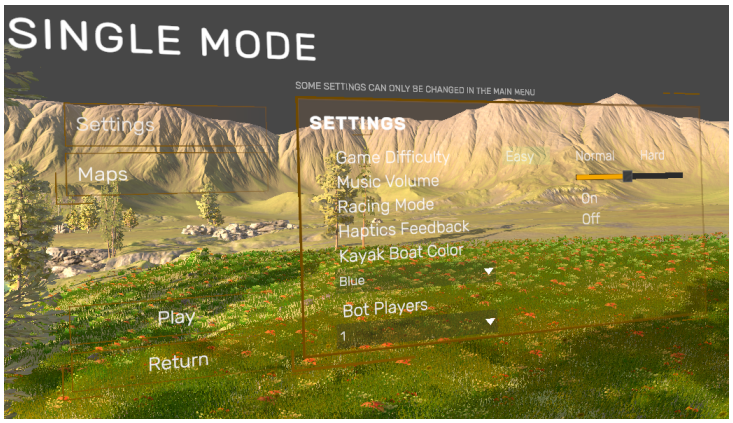

Fig. 26. Single Mode User Interface (UI) with tabs:Settings (change game configuration settings) and Maps(choose between different Maps)

Game Play

While in the ”Game” scene, players can improve kayaking experience with real-time HUD UI feedback, including a map, speed meter, and race ranking updates. By navigating to the ”Pause Menu” the players can configure game options such as enabling or disabling the directional arrow which dynamically indicates the direction to next checkpoints, displaying key bindings, adjusting frame rate visibility and take screenshots. Users can also check the BLE connection with the ESP32 module and return to the Main Menu at any point. As a result of real-time mapping of two VR controller positions to paddle ends, users can move and rotate the kayak based on the physical paddling movements. High-fidelity

water system enhances the water visuals integrating features such as FFT Waves, ripples, reflections, attenuation, particles, buoyancy physics etc. Upon completing the race, the system transitions to the ”Results” scene providing results of the race and after users can navigate to the Main Menu.

Fig. 27. A screenshot of the Game play. The display shows various HUD UI components: the Ranking list, speed meter, map and time remaining etc., Also, there is the arrow showing the direction of next checkpoint.

6.3 Score System

In the Main Menu, ”Leaderboard” panel for promoting the boost engagement and learning outcomes [16] of the users, represents three tabs. The “Leaderboard” tab provides a comprehensive view of player rankings based on their gained scores. The “Hints” tab exhibits various strategies how to increment their scores and rise the leaderboard. The players have the opportunity to complete predefined achievements which the “Rewards” tab shows and unlock them to earn additional scores. By the ”Settings” panel in the Main Menu, players are able to change various system configurations. The ”Profile” tab enables users to view, edit, or create player profile details. The ”Game” tab allows adjustments to game-play settings, such as difficulty level, music volume, HUD and tooltips visibility while the ”Control” tab configures VR controller input keys and sensitivity. In the ”Video” tab, users can customize graphics settings. Additionally, the ”Credits” panel highlights the development history of the VR kayaking system and show details of people who created.

6.4 The training result of PPO Model

After the re-training, the PPO model outputs best results with a mean cumulative reward and value loss of 0.7702 and 2.0040e−-05, respectively. Therefore, these results represent that we are able to use the output model for navigating the ML Agents properly. Figure 28 & Figure 29 show the corresponding plots.

6.5 Video Pre-processsing Results

We measure the model accuracy in detecting the water region over different kinds of frames in different conditions,

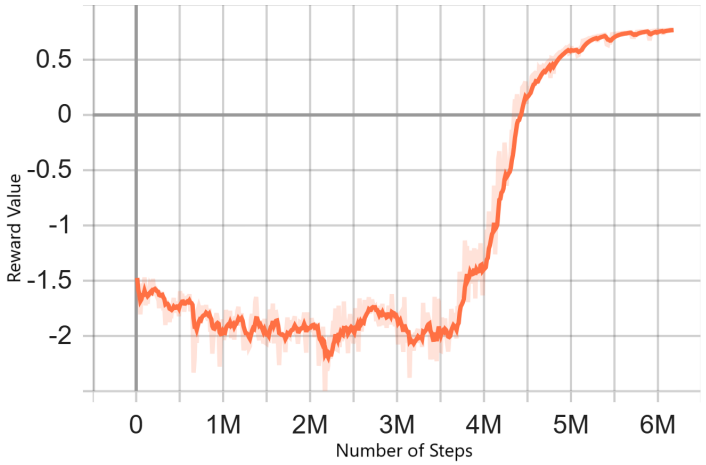

Fig. 28. Default PPO result plots: Reward value vs Number of steps taken to train the NN model

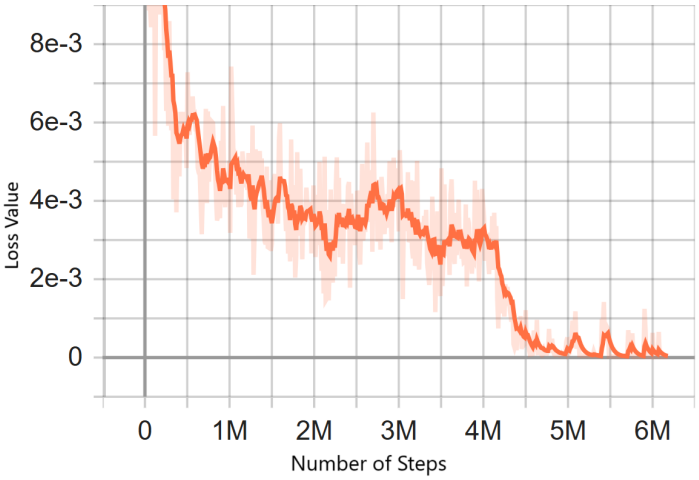

Fig. 29. Default PPO result plots: Loss value vs Number of steps taken to train the NN model

i.e., for daytime and nighttime conditions for both turbulent and normal water. Our model shows a very good detection of water regions, maintaining high accuracy and finding the best performances across the considered scenarios.

Results are in Table II by listing the accuracy of detection for each scenario. Our model proves to be robust and adapted to various conditions.

TABLE II
RESULTS OF WATER DETECTION ACCURACY ACROSS SCENARIOS

Scenario	Accuracy (%)
Daytime - Turbulent Water	85
Daytime - Normal Water	92
Nighttime - Turbulent Water	78
Nighttime - Normal Water	88

7 Conclusion

The VR kayaking system shows how immersive technologies can be used to solve real-world issues raised when kayaking, like lack of infrastructure and safety concerns. The system provides two distinctive modes which are competitive and engaged modes. Also, our system improves user experience and engagement using machine learning algorithms. This system is very useful. We can use this system in gaming cafes, and fitness centers and also we can use this to promote tourism of a country because foreigners will able to experience kayaking places of a country without actually visiting them.

References

[1] H.-J. Kim, Y.-R. Kim, and Y. Lee. (2015). "A new approach to improve cognition, muscle strength, and postural balance in community-dwelling elderly with a 3-D virtual reality kayak program," *Journal of Physical Therapy Science*, vol. 27, no. 11, pp. 3545–3548.

[2] F. Dereszynski, K. Yeadon, and M. Pain. (2005). "Optimised preliminary design of a kayakergometer using a sliding footrest-seat complex," *Engineering of Sport 6*, Springer, pp. 173-178.

[3] A. Jeffrey, S. Faudzi, S. Wan, and Y. Yoshida. (2020). "Virtual Kayaking: A Local Culture-Based Virtual Reality Paddling Experience," *2020 3rd International Conference on Innovative Research and Development (ICIRD)*, Kuala Lumpur, Malaysia, pp. 1-6.

[4] A. Giannakakis, I. Leontaris, and P. Xenidis. (2020). "The Impact of Passive Haptics on the User Experience While Exercising in Virtual Reality," in *Lecture Notes in Computer Science*, vol. 12188, pp. 246-257.

[5] J. Yang, C. Wang, Y. Chen, and C. Wang. (May 2021). "Development of a 360-degree virtual reality video-based immersive cycle training system for physical enhancement in older adults: a feasibility study," *Journal of NeuroEngineering and Rehabilitation*, vol. 18, no. 1, pp. 1-10.

[6] Z. Chen, Y. Liu, and J. Zhang. (2020). "360-degree Image Processing on NVIDIA Jetson Nano," *2020 IEEE International Conference on Consumer Electronics (ICCE)*, Las Vegas, NV, USA, pp. 1-2.

[7] Liu, Kao-Hua & Sasaki, Tomoya & Kajihara, Hiroyuki & Hiyama, Atsushi & Inami, Masahiko & Chen, Chien-Hsu. (2020). Virtual Kayaking: A Local Culture-Based Virtual Reality Paddling Experience. 10.1007/978-3-030-50249-2_45.

[8] Dong, Qian & Dargie, Waltenegus. (2012). Evaluation of the reliability of RSSI for indoor localization. 2012 International Conference on Wireless Communications in Underground and Confined Areas, ICWCUCA 2012. 1-6. 10.1109/ICWCUCA.2012.6402492.

[9] Borella, Michael & Swider, Debbie & Uludag, Suleyman & Brewster, Gregory. (1998). Internet packet loss: Measurement and implications for end-to-end QoS. 3-12. 10.1109/ICPPW.1998.721868.

[10] J. Hanski and K. B. Biçak. (2021). 'An Evaluation of the Unity Machine Learning Agents Toolkit in Dense and Sparse Reward Video Game Environments', Dissertation.

[11] Xi, Wang. (2020). Research on Application of Artificial Intelligence in VR Games. 10.3233/FAIA200704.

[12] Savid, Yusef & Mahmoudi, Reza & Maskeliunas, Rytis & Damaševičius, Robertas. (2023). Simulated Autonomous Driving Using Reinforcement Learning: A Comparative Study on Unity's ML-Agents Framework. Information. 14. 290. 10.3390/info14050290.

[13] Angelos Barmpoutis, (Jan. 2020), "Virtual Kayaking: A Study on the Effect of Low-Cost Passive Haptics on the User Experience While Exercising," Communications in computer and information science, pp. 147–155, doi: https://doi.org/10.1007/978-3-030-50729-9_20.

[14] Liu, Kao-Hua & Sasaki, Tomoya & Kajihara, Hiroyuki & Hiyama, Atsushi & Inami, Masahiko & Chen, Chien-Hsu. (2020). Virtual Kayaking: A Local Culture-Based Virtual Reality Paddling Experience. 10.1007/978-3-030-50249-2_45.

[15] Barmpoutis, Angelos & Faris, Randi & Garcia, Samantha & Li, Jingyao & Philoctete, Joshua & Puthusseril, Jason & Wood, Liam & Zhang, Menghan. (2020). Virtual Kayaking: A Study on the Effect of Low-Cost Passive Haptics on the User Experience While Exercising. 10.1007/978-3-030-50729-9_20.

[16] Park, Sungjin & Kim, Sangkyun. (2019). Development of Leaderboard Design Principles to Improve Motivation in a Gamified Learning Environment (Preprint). JMIR Serious Games. 9. 10.2196/14746.

[17] J. Doe, A. Smith, and B. Brown. (2023). "360° Video Metadata Standards," in *Proc. IEEE VR Conf.*, pp. 45–50.

[18] A. Jones and L. Martin. (2022). "Segmentation Models for Panoramic Videos," *ACM Trans. Multimedia Comput. Commun. Appl.*, vol. 18, no. 4, pp. 123–132.