

Software Engineering Department

Braude College of Engineering

Capstone Project Phase B

Spell Runner - Wizard VR



[GitHub Repository](#)

Project Code: 24-1-D-37

Advisor: PhD. Sulamy Moshe - MosheSu@braude.ac.il

Students: Habib Ibrahim - Habib.Ibrahim@e.braude.ac.il

Vitaly Rofman - Vitaly.Rofman@e.braude.ac.il

Contents

Abstract	3
1. Introduction.....	4
2. Development Process	5
2.1 Diagrams.....	6
2.1.1 Deployment Diagram	6
2.1.2 Activity Diagram	6
2.2 Scenes and Flow.....	7
2.3 Convolutional Neural Network	10
2.3.1 Model Architecture	10
2.3.2 Architecture Explanation	11
2.4 Machine Learning	12
2.4.1 Prediction Process	12
2.4.2 Model Fine-Tuning.....	12
2.5 Challenges and Solutions	14
2.6 Technologies and Platforms Used	16
2.7 Testing.....	16
3. User Manual	17
3.1 Getting Started	17
3.1.1 System Requirements	17
3.1.2 Connecting VR Headset	17
3.1.3 Game Installation	17
3.2 In-game Guide	17
3.2.1 Main Menu Scene	17
3.2.2 Game Controls	19
3.2.3 Game Interface.....	20
3.2.3 Gameplay.....	23
4. Maintenance Manual	24
4.1 Game Compilation.....	24
4.2 Fitting CNN Model.....	25
4.3 Adding Spells.....	26
4.4 Adding Enemies	26
5. Results and Further improvements	27
5.1 Results.....	27
5.2 Further Improvements.....	28
6. References	29

Abstract

Spell Runner is an immersive virtual reality game that places players in a magical world where they assume the role of a wizard. The game centers around a unique spell casting system, where players draw specific shapes in a 3D environment using hand gestures tracked by a VR headset. These drawings are analyzed by a Convolutional Neural Network (CNN) model, which predicts and categorizes the spells based on the accuracy of the player's input. The CNN model continuously fine-tunes itself based on the player's drawing patterns and style, improving its accuracy over time. The game also features an adaptive feedback system that displays prediction array and allows players to correct misrecognized spells which falls below the prediction threshold, ensuring a fluid and engaging gameplay experience. Spell Runner combines advanced machine learning with intuitive gameplay to create a challenging and enchanting adventure.

1. Introduction

Spell Runner is an innovative virtual reality (VR) game that transports players into a magical realm where they become wizards. Leveraging machine learning technology, Spell Runner interprets hand-drawn shapes made by players through a VR headset, determining the spells to be cast based on these drawings. The core of Spell Runner is its spell-casting system. Players utilise hand gestures and movements, tracked by the VR headset, to draw shapes that correspond to spells listed in their spell book. Each spell has a distinct draw pattern, thereby challenging players to refine their reaction time and precision.

The spell book serves as the repository of magical knowledge in the game. It contains a single group of seven spells, each with its own unique style. Players navigate through this spell book to master these spells. Spell Runner features a Convolutional Neural Network (CNN) model that analyses and categorizes the hand-drawn shapes made by players. This model is trained to recognize the relationship between gestures and specific spells. As players cast more spells, the system adapts and improves its recognition accuracy, enhancing the gameplay experience.

The integrated CNN model in Spell Runner is designed to continuously learn and adapt to each player's unique drawing patterns and style, thereby enhancing the user experience over time. In the initial phase, the model utilises a pre-trained dataset to categorise the player's patterns and identify the corresponding spells. However, as the player progresses and repeatedly casts spells, the model accumulates data on the player's distinctive drawing style, including aspects such as curvature and accuracy.

Additionally, players can benefit from an output system that is visible through the VR headset, which displays the predicted spell that has been cast. If the recognition accuracy of the spell falls below the specified threshold, an appropriate message is displayed, thereby allowing the player to select the spell that they intended to draw. This feature guarantees a more fluid gaming experience, allowing players to correct misidentified spells and tolerate the continuity of their magical journey.

2. Development Process

The development of Spell Runner involved several key phases, each of which was selected with ensuring that our game requirements were met, and that no aspect was overlooked.

Phase 1: Drawing in 3D Environment

We began by creating a 3D environment where players could draw spell shapes using their hand movements tracked by a VR headset. This initial phase involved setting up the VR interface and ensuring that hand gestures could be accurately captured and translated into digital images within the 3D space.

Phase 2: Training the CNN Model

The next stage of the project was to develop and train a CNN model. The objective of this model was to analyse hand-drawn shapes and predict the corresponding spells. We collated a significant data set containing a variety of spell patterns that we ourselves drew in the 3D environment we had previously designed. This data set was then used to train the CNN model, ensuring that it could accurately recognise and classify different spell drawings.

Phase 3: Integrating Machine Learning and Communication

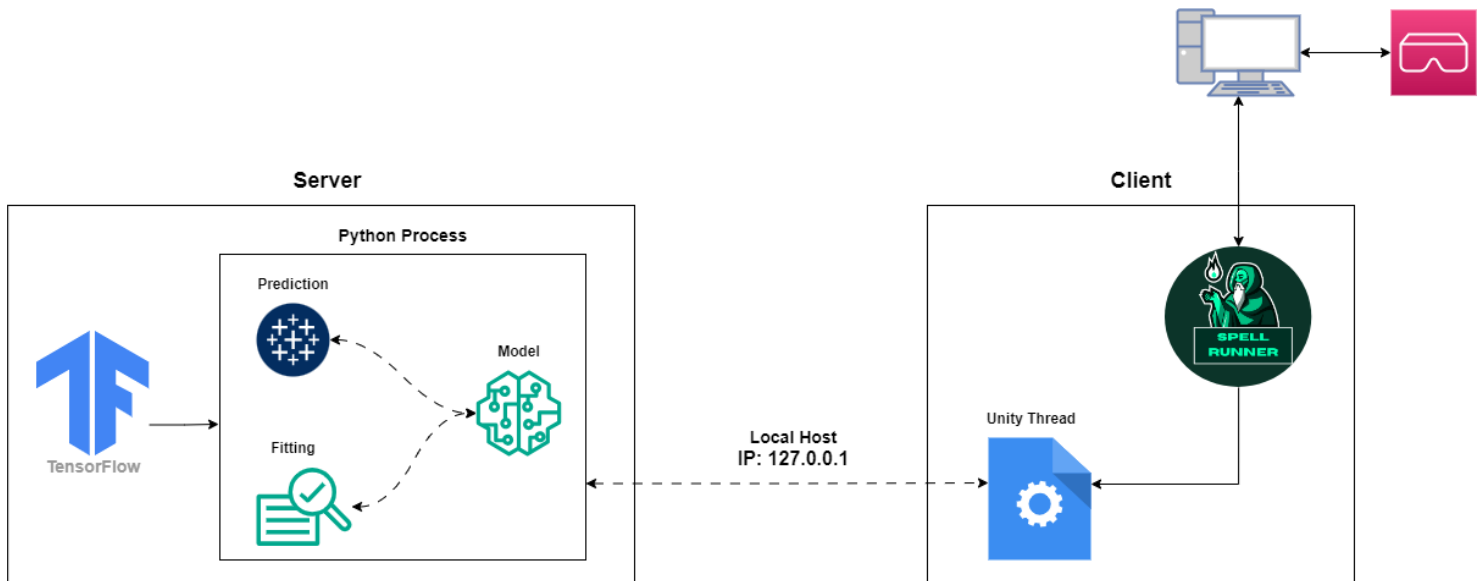
Once the CNN model was trained, we integrated the machine learning system with the game. This required establishing a seamless communication pipeline between the game engine and the Python environment where the CNN model was running, which provided a real-time data exchange between the two processes, allowing the game to send hand-drawn shapes images to the CNN model and receive predictions instantly. This integration was crucial for providing immediate feedback to players on their spell casting accuracy.

Phase 4: Enhancing the 3D Game

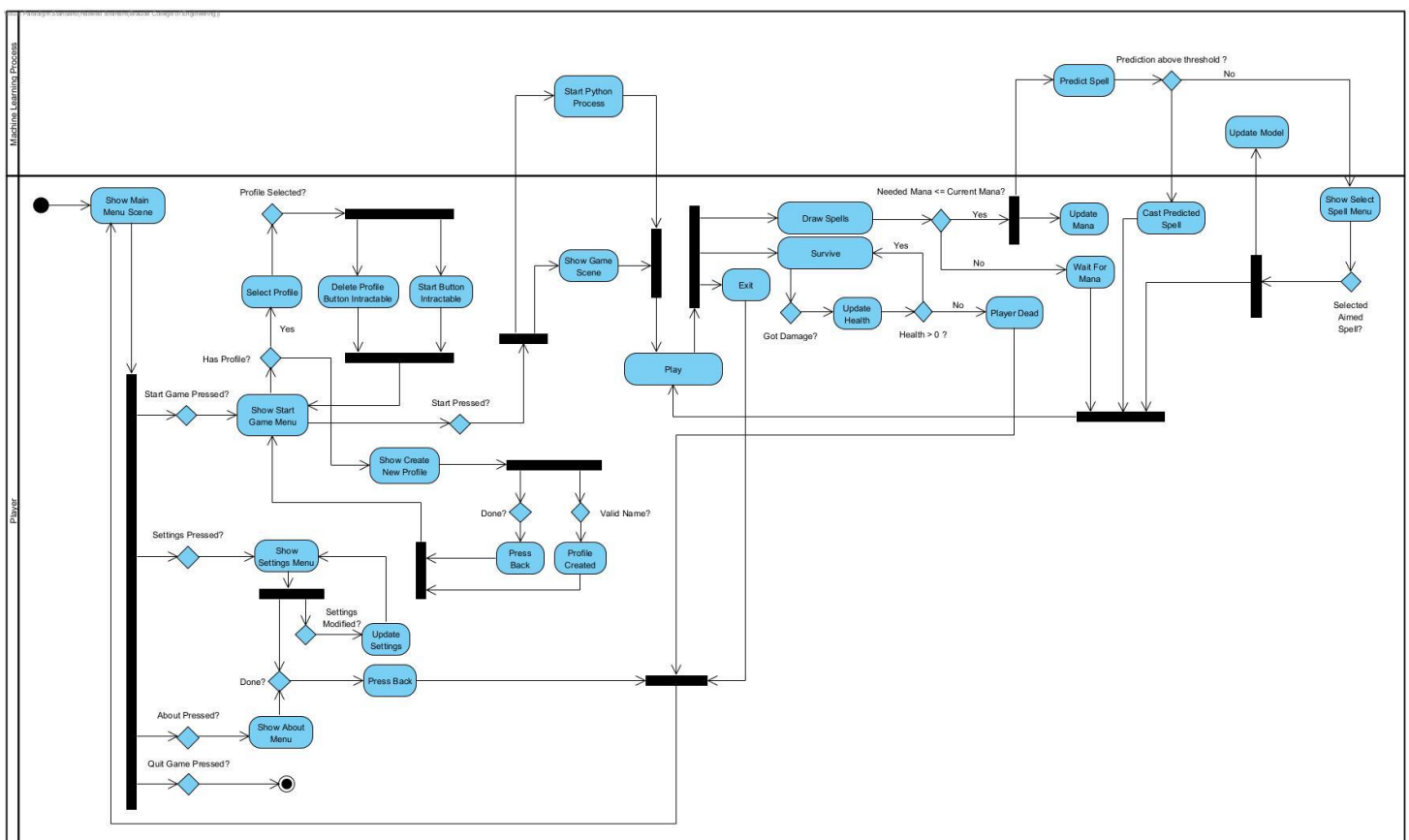
With the machine learning system in place, we turned our attention back to the 3D game interface. We refined the visual elements and user interactions to ensure a smooth and engaging experience. This included designing the spell book, creating visual effects for successful and unsuccessful spell casts, and implementing the output system that displays the predicted spell and accuracy feedback through the VR headset. We also added a feature to allow players to manually select the intended spell if the recognition accuracy fell below the threshold, ensuring a user-friendly experience.

2.1 Diagrams

2.1.1 Deployment Diagram



2.1.2 Activity Diagram



2.2 Scenes and Flow

Main Menu Scene

Upon entering the game, the user is presented with the main menu, which allows them to commence gameplay, modify settings, review instructions, or exit the game.



It is essential to select a profile before entering the game, as each player is assigned a CNN model. New players will receive an initial model.



Game Scene

Once the player has selected their profile and initiated the game, they will be transported to a new scene, which serves as the game environment. In this environment, the player must engage in combat with enemy characters and survive.



The primary objective for the players is to follow the green arrow, which will direct them to the interior of the large building housing a big port. It is imperative that they destroy this port to survive. During their journey to the building, they will encounter a multitude of enemies, which they must draw to attack and destroy.

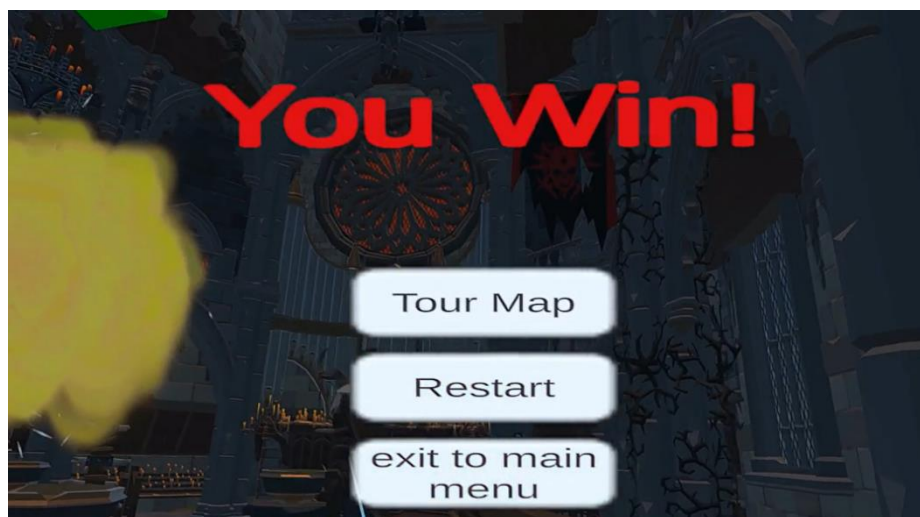


Model Fine Tuning

The player will draw spells from the spell book held in their left hand. If a player draws a specific spell that is not recognised as appropriate by the model, a menu will appear, allowing the player to select an aimed class. The model will then adjust itself based on the class selected by the player.



Upon reaching the port and destroying it, the player will be declared the winner. All enemies will then disappear, and the win menu will be displayed. The player may then select one of three options: to tour the map, to start a new game, or to return to the main menu.



2.3 Convolutional Neural Network

2.3.1 Model Architecture

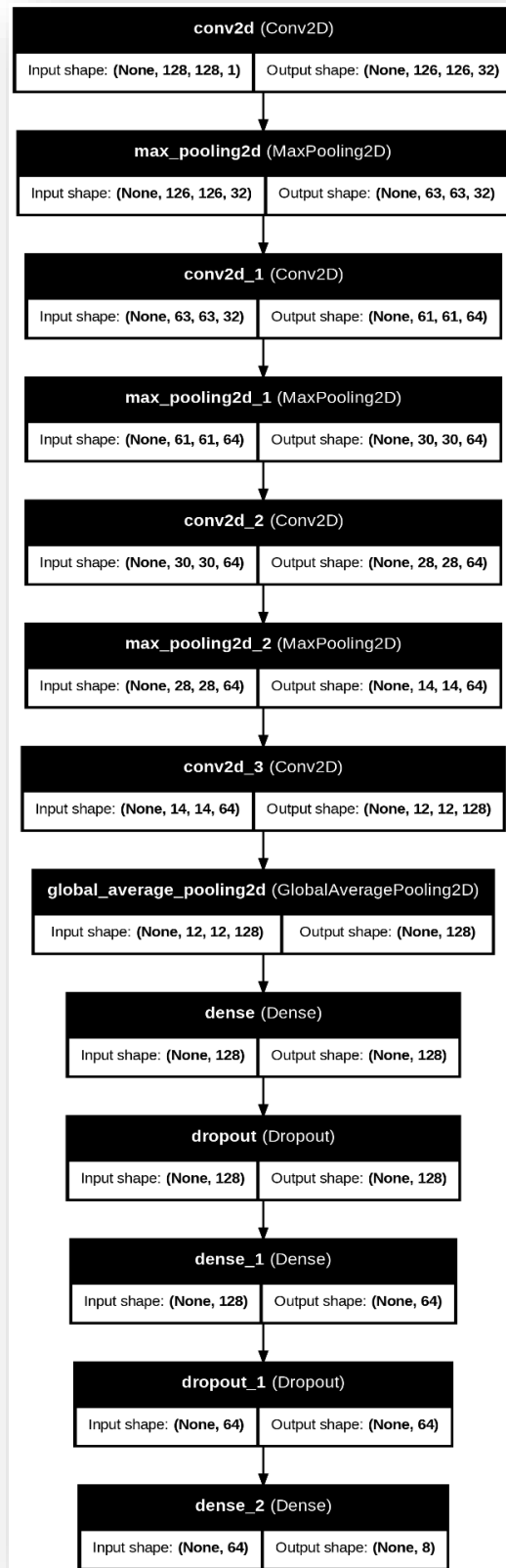


Figure 1: CNN Model Architecture

2.3.2 Architecture Explanation

Convolutional Layers (Conv2D)

The model begins with a sequence of four convolutional layers, each succeeded by a max-pooling layer. Convolutional layers are designed to extract features from the input images:

1. **First Conv2D Layer:** Uses 32 filters.
2. **Second Conv2D Layer:** Uses 64 filters.
3. **Third Conv2D Layer:** Uses 64 filters.
4. **Fourth Conv2D Layer:** Uses 128 filters.

Pooling Layers

Each convolutional layer is followed by a max-pooling layer. The principal function of max-pooling layers is to reduce the spatial dimensions of the feature maps. This down sampling facilitates a reduction in the number of parameters and computational load, thereby enhancing the efficiency of the network. Furthermore, max pooling facilitates the detection of features that are invariant to small translations in the input.

Global Average Pooling

Subsequently, a global average pooling layer is applied following the series of convolutional and pooling layers. This layer performs a reduction of each feature map to a single number, whereby the average of all values within the feature map is computed. This operation markedly moderates the overall number of parameters, thereby assisting in the prevention of overfitting and enhancing the model's robustness.

Dense Layers

Following the global average pooling layer, the model incorporates three fully connected (dense) layers:

1. **First Dense Layer:** Comprises 128 units.
2. **Second Dense Layer:** Comprises 64 units.
3. **Third Dense Layer:** Comprises 8 units

Dropout

To prevent overfitting, dropout is included after the first two dense layers. During training, dropout randomly set a fraction of the input units to zero at each update, which helps in regularizing the model.

2.4 Machine Learning

2.4.1 Prediction Process

One of the key differentiators of our game is its ability to learn and adapt to players' drawing patterns. When a player creates a new image, it is sent to a Python process where the model is waiting for input to predict it.

If the model's prediction score falls within a certain threshold range (between 95% and 100%), it indicates that the model knows this, and it can accurately predict similar images in the future. If the predicted class is between 90% and 95%, the model will self-tune based on predicted class. However, if the prediction is below 90%, the user will be prompted to enter the intended class. This will allow the model to learn that this input was meant to be in the user-entered class.

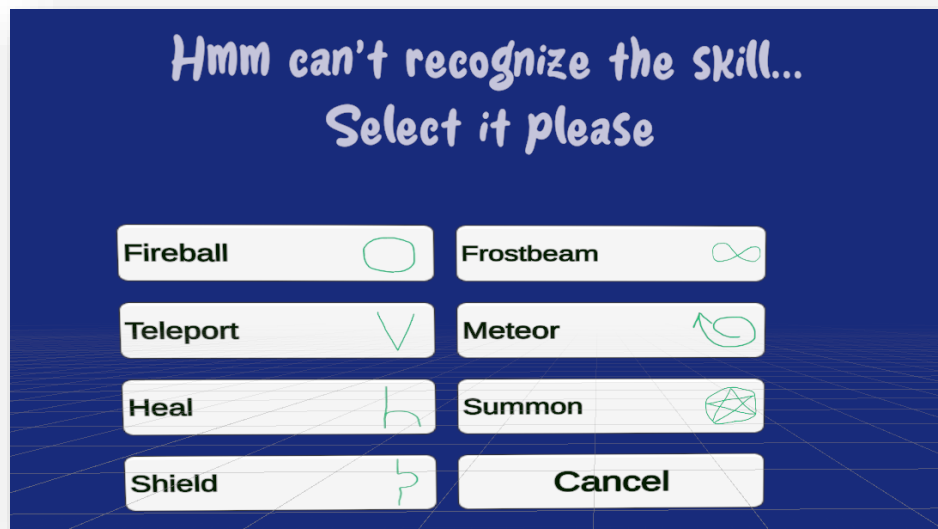


Figure 2: Bad Prediction Menu Where Player Selects Intended Skill

2.4.2 Model Fine-Tuning

To prevent the model from becoming over-fitted on a specific skill, we have implemented a buffer system to prepare images for each epoch. This approach ensures that the model receives a diverse set of images from different classes, allowing it to learn new images without becoming overly biased towards a particular class.

We used buffers to prepare the data(images) to fine-tune the model during gameplay, for each skill, a buffer is created that includes the images that represent that skill. At the outset, all buffers are populated with dataset images, images that have been drawn as part of the data set and used to train the model.

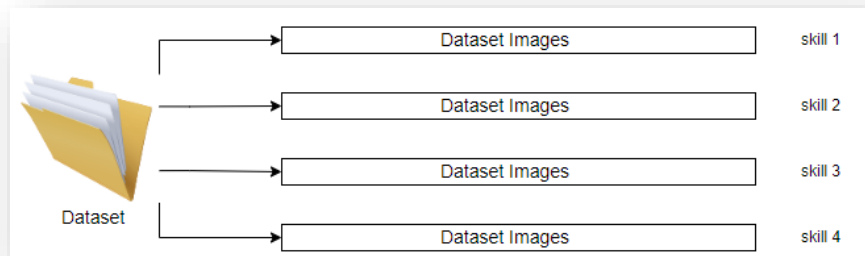


Figure 3: Initially Buffers Full of Dataset Images

While the player is drawing new images, the dataset images will be removed, and the player images will be loaded into the buffer. This is because we want the model to learn the player's drawing patterns, and it is preferable to always have access to his images.

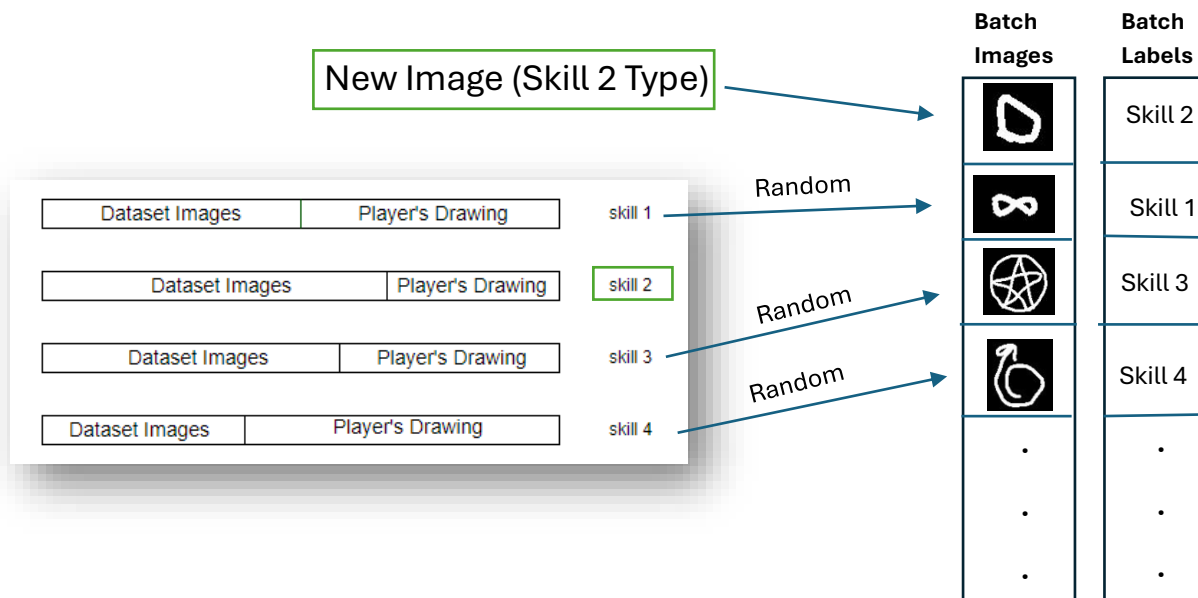


Figure 4: Buffers will be split between dataset images and new drawings

When a new image is required for the model to learn, a batch of images is prepared for entry into the learning session for one epoch, as illustrated in Figure 4. The images in the batch will include the new image and an old image from each other class. This approach is designed to prevent overfitting, ensuring that the model sees all types of images from the class during each new learning session.

This approach guarantees that the model will consistently learn new player images. Furthermore, it prevents the model from overfitting to a specific class, as each epoch of learning incorporates a diverse set of images, including the new ones.

2.5 Challenges and Solutions

Challenge 1

What method should be used to obtain a data set of the requisite shapes?

Solution

A three-dimensional environment was designed to simulate the drawing of shapes at varying angles and patterns. These shapes were then saved and sorted into categories, with each category bearing the name of its respective class.

Challenge 2

How to learn the model new images without over-fitting on a particular class?

Solution

We explained the solution in 2.4.2, where model Fine-Tuning was explained.

Challenge 3

Implementing communication between python (ML) to game (Unity Engine).

Solution

A connection was applied on the local host for the purpose of providing communication between unity and Python process. This enabled the transfer of skill images, the reception of prediction arrays.

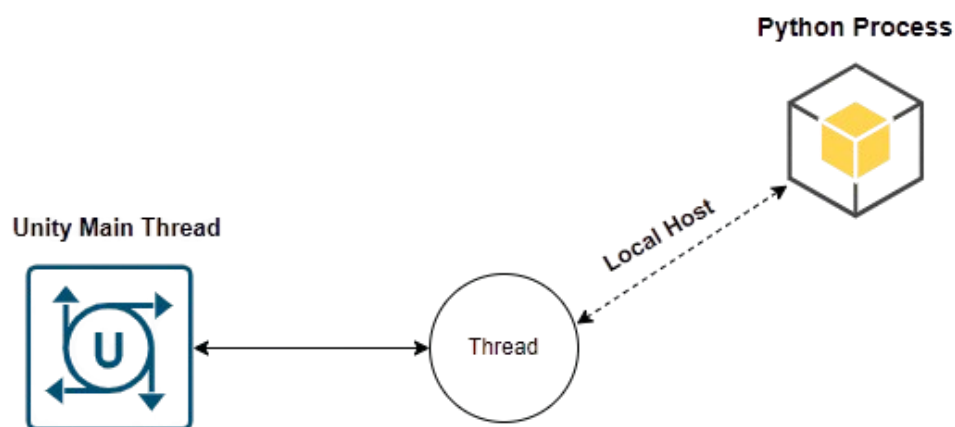


Figure 5: Communication Between Unity Engine and Python Process

Challenge 4

We need to implement a damage system for enemies. For some skills, it should damage multiple enemies, not just one.

Solution

An interface (IDamageable) was developed whereby all enemies were required to implement the TakeDamage function. Afterward, a verification process was conducted for each object that had been hit and implemented this interface, thus requiring it to take damage.

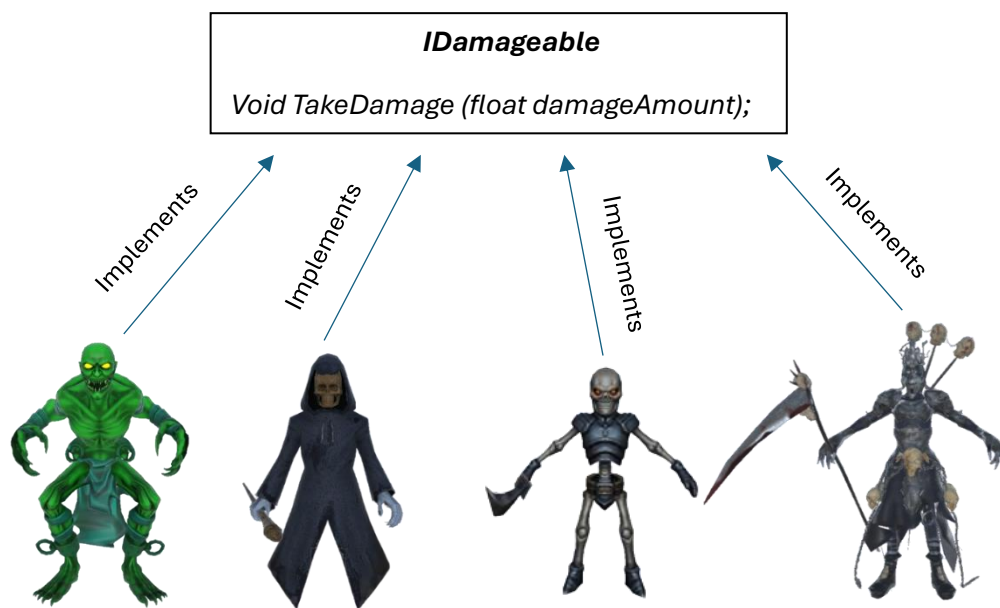


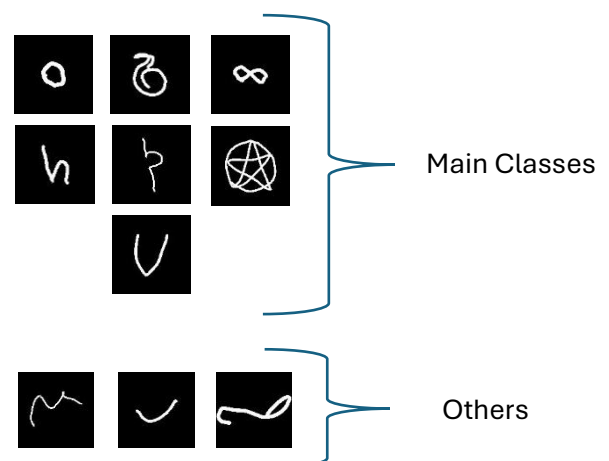
Figure 6: Enemies Implements IDamageable Interface

Challenge 5

If skill is not relevant to any of the class, to what class it should be classified?

Solution

A new class, 'Others', has been added to the dataset to include different styles and patterns of shapes that are not relevant to any of the spell classes. The model has been trained to classify this type of image, which has also helped to prevent irrelevant patterns being classified in the wrong class.



2.6 Technologies and Platforms Used

We employed a variety of software tools, including Unity for game development, Visual Studio for coding, Visual Studio Code for lightweight editing, Git and Git LFS for version control, and GitHub for repository hosting. These tools facilitated a more efficient workflow, enabling effective project management.



Figure 7: Technologies and Platforms

2.7 Testing

Test no.	Module	Test Description	Result
1	Player Controller	The test verifies whether the player can commence the drawing process solely when the Python process is connected and prepared to commence drawing prediction.	Passed ✓
2	Player Controller	The test ascertains whether the game interface is functioning as intended, specifically in relation to the main menu, pause menu, prediction menu and winning/losing menu.	Passed ✓
3	Player Controller	The test ascertains whether a new model is created when a player creates a new profile. It is essential that each player has a model that reflects their draw patterns.	Passed ✓
4	Spell Cast System	The test verifies that the player can cast spells of varying types (e.g., fireball, lightning bolt) and that the intended spell effect is triggered when the CastSpell function is invoked.	Passed ✓
5	Python Connector	The test checks that the Python process is running, and a connection has been made to the local host, and that it is ready to receive data.	Passed ✓
6	Python Connector	The test checks that the Python process is opening the right profile CNN model, successfully making a prediction and sending right format of prediction array.	Passed ✓
7	Enemies Controller	Tests if the enemy AI properly follows and chases the player character when given the player's current position as a parameter.	Passed ✓
8	Feedback Controller	The test checks whether the prediction array is displayed correctly as the data received from Python after model prediction.	Passed ✓
9	Feedback Controller	The test checks that the correct data is passed when the player selects the desired skill from the 'BadPredictionMenu'.	Passed ✓

3. User Manual

Welcome To Spell Runner: Wizards VR

Mastering the art of drawing spells is essential to your survival and success in the world of wizards. Each spell requires you to accurately draw specific shapes using your hand gestures, tracked by the VR headset. These shapes represent powerful magical incantations, and the precision of your drawings directly impacts the effectiveness of the spells you cast.

3.1 Getting Started

3.1.1 System Requirements

- Memory - 8 GB+ RAM.
- Graphics Card - NVIDIA GeForce GTX 970 / AMD 400 Series.
- Processor - Intel i5-4590 / AMD Ryzen 5 1500X or greater.
- Operating System - Windows 10, Windows 11.

3.1.2 Connecting VR Headset

- Full guide how to set-up [Meta Quest](#) Headset.

3.1.3 Game Installation

1. **Join Mega Link:** <https://mega.nz/folder/8XokQLzQ#Dy7y0sugydKLJf8EQjb4IA>
2. **Download the installer, run it and follow the on-screen instructions.**
3. **Launch the executable file of the game, WizardVR.exe.**

3.2 In-game Guide

3.2.1 Main Menu Scene

Once successfully launched the game, the main menu will be displayed. This menu will allow to start the game, modify settings or exit the game.

1) Game Main Menu



Users will have the option to either select an existing profile or create a new one. All profiles will start with a basic model based on the data set.

2) Selecting/Creating Profile Menu



Once you have selected your profile, please click on the 'Start Game' button to begin playing. You will then be transferred to a new scene, 'Game Play Scene'(3). At this point, you should prepare yourself for the challenge ahead. Once the 'Get ready' text disappears, you will be ready to start drawing and playing.

3) Game Play Scene



On the left-hand side of the screen, you will see a book that displays the available spells. The amount of mana required to cast each spell is shown beneath it. Your HP and mana bars can be seen below the spells.

3.2.2 Game Controls

Once the player has selected their profile and initiated the game, they should be ready to move and draw by holding the corresponding controller. The right controller thumbstick (1) enable the player to move and on click it enable prediction values bar, while left controller thumbstick (1) enable the player to rotate camera, the right controller grip button (5) must be held to draw, the left controller menu button (2) opens menu in gameplay.

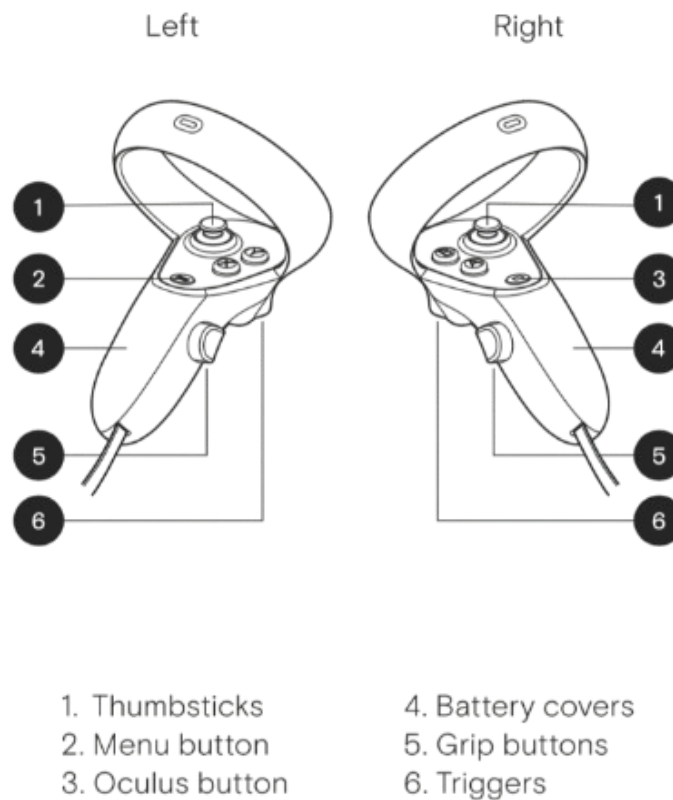


Figure 8: Meta Quest Controllers

3.2.3 Game Interface

As you progress through the game, you will encounter various menus, each of which will be a result of an action taken during gameplay.

1) Drawing Process

To defend yourself from enemies, you must draw spells and attack them. The process of drawing is straightforward: you must be standing and stationary and adopt an optimal position.



Then hold on the right controller grip button and start drawing the desired spell, upon completion of the drawing, the button should be released.



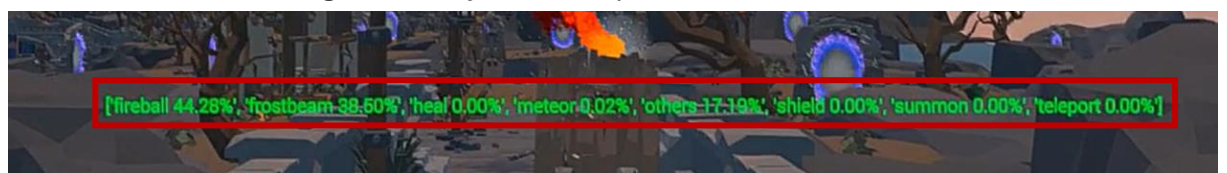
Once the button has been released, the draw will be captured and transmitted to the processing stage. Upon the completion of the prediction, the user will be presented with an appropriate skill animation that aligns with the desired skill.

Skill Animation

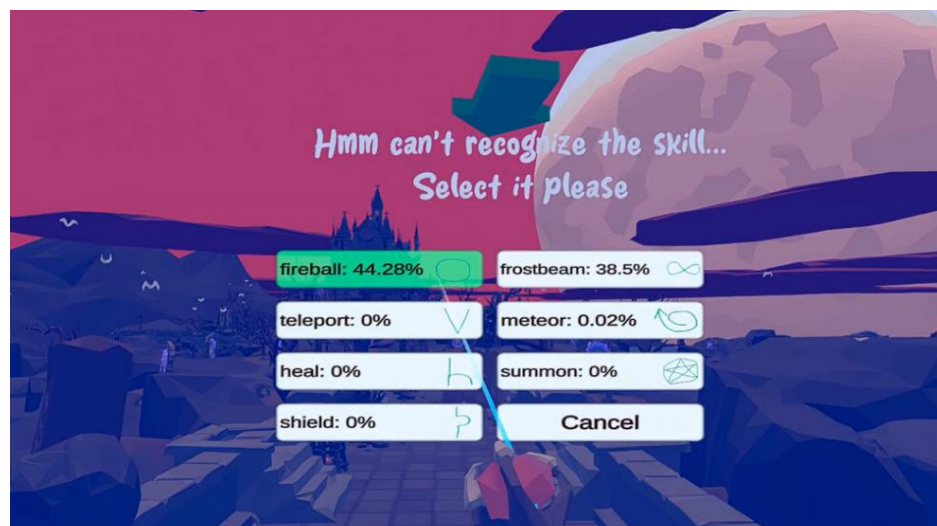


2) Prediction Bar

As you play the game and cast different spells, you can press the right controller thumbstick to show a prediction value bar. This includes the probability value of each skill class, which helps you see how well your model is doing and how you can improve it.



3) Selecting Aimed Spell Menu



This menu will appear when the model prediction accuracy for a given drawing is not satisfactory. The game will then provide the opportunity to select the desired skill, which will then be applied and enable the model to learn about it.

4) Pause Menu



By pressing the menu button on the left-hand control device, the game will be placed in a state of suspension, and a pause menu will be displayed. The player may then elect to either resume play when they are ready or to exit the game and return to the main menu.

5) Winning Menu



Upon completion of the game and survival, the win menu will be displayed, offering the player the option to explore the game map, restart the game, or exit to the main menu.

6) Losing Menu



In the event of a loss, and a reduction in health points to zero, the player will be unable to continue, and a losing menu will be displayed. The player will then have the option to either restart the game or exit to the main menu.

3.2.3 Gameplay

The objective of the game is to survive. The world is full of enemies who will try to attack and damage. In the world, there is a large building with a large port inside. Once you destroy this port, all enemies will disappear, and enemy ports will be broken.



4. Maintenance Manual

4.1 Game Compilation

To compile the game, it is first necessary to ensure that the Unity Editor is installed with version 2022.3.32f1. The next step is to clone the project from the GitHub repository to the Unity Editor. It is then essential to check the console for any errors, as these may have originated from the editor rather than the project itself. If any errors are present, it is necessary to resolve them before proceeding to the next stage. Once the project has been cloned and configured in the editor without any errors, navigate to the "Edit" menu, then select "Project Settings" and finally "Player." Here, it is essential to ensure that the scripting backend (configuration) is set to "Mono."

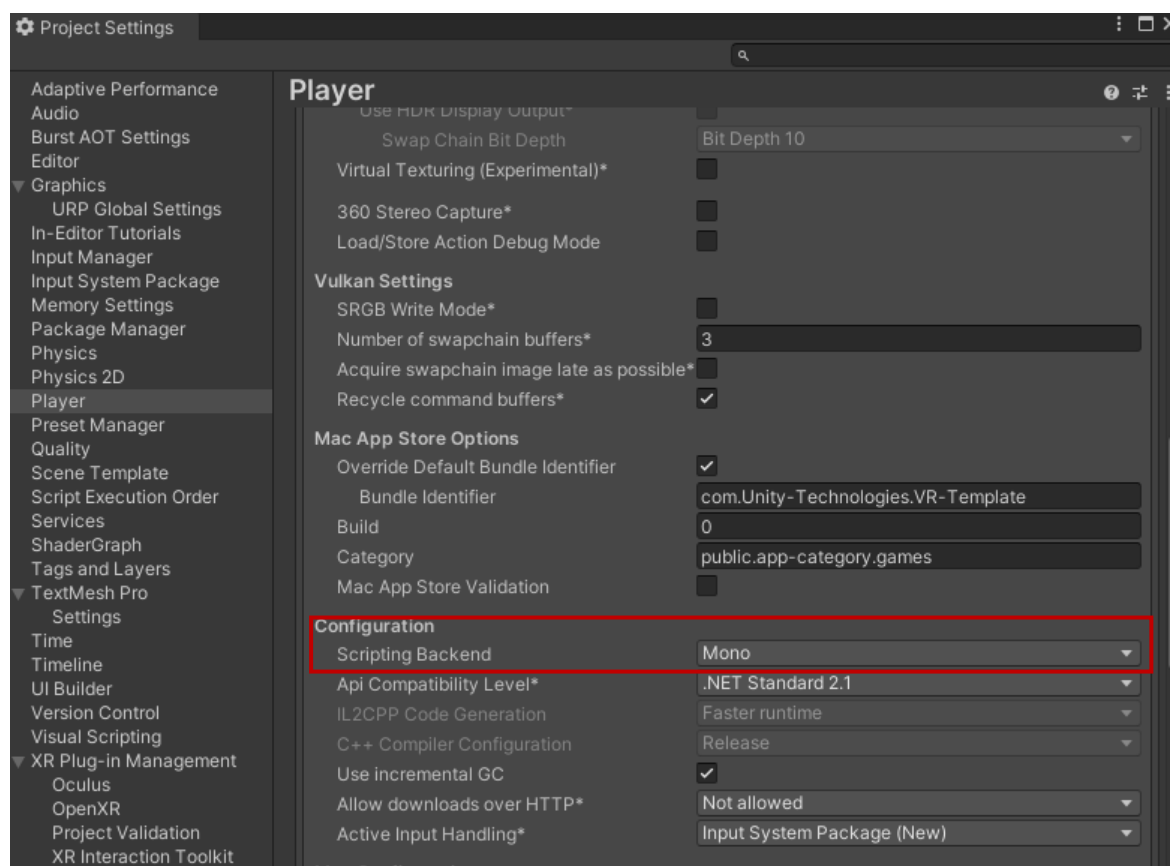


Figure 9: Unity Editor Project Settings Menu

Once the configuration has been established, proceed to the "File" menu, then "Build Settings" option should be selected, ensuring that all game scenes are selected and that the target platform is set to "Windows." Once this has been done, the build and run option should be selected.

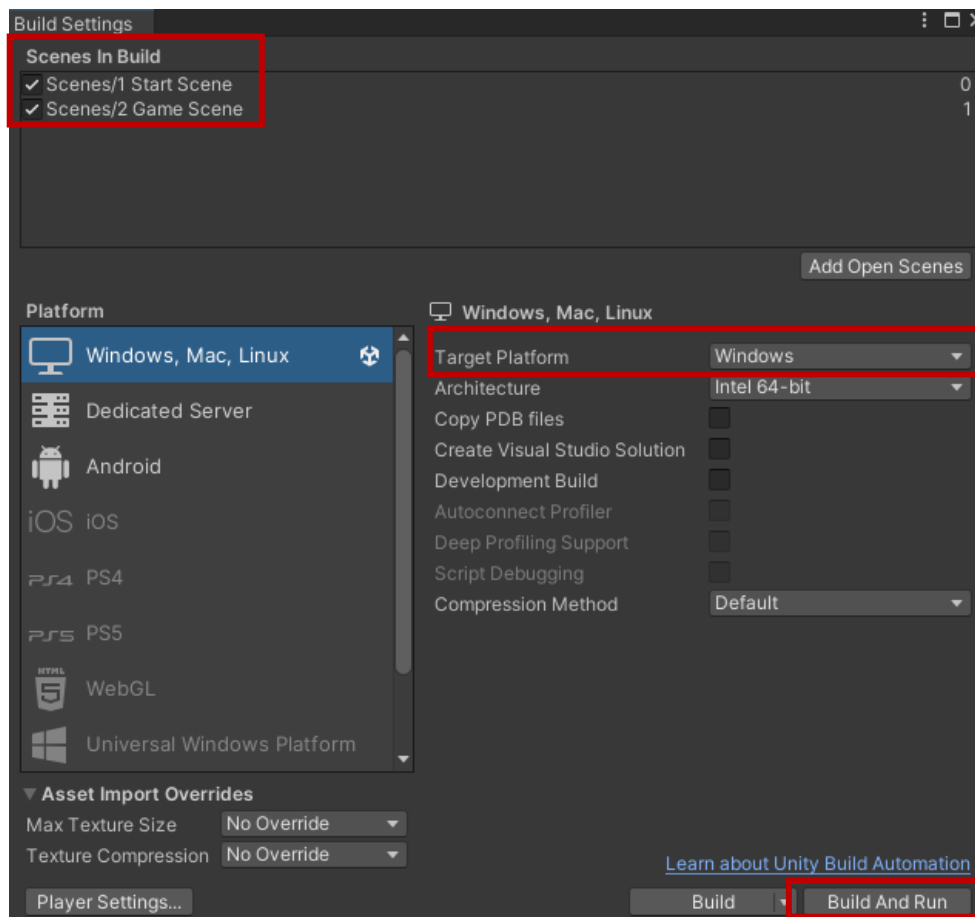


Figure 10: Unity Editor Build/Run Menu

4.2 Fitting CNN Model

The skill images are located in the directory "WizardVR/Assets/Scripts/UnityPython/BufferImages". Each skill is represented by a folder containing various shapes and styles. To create a new model, one can access the trainmodel.py file, which is located in "WizardVR/ModelTrainScript." This file requires access to the skill images, which can be enabled by modifying the "base_path" within the file to "/Assets/Scripts/UnityPython/BufferImages."

```
# Main execution
if __name__ == "__main__":
    base_path = '.' # Assuming the script is in the same directory as the skill folders
    model_save_path = 'spell_recognition_model.h5' # Path to save the model
```

Once the path for the dataset has been configured, the script can be executed, resulting in the generation of a new model that will be saved after training in a file called spell_recognition_model.h5. If needed to replace the existing model with a new one, it is necessary to enter the path "WizardVR/Assets/Scripts/UnityPython/models" and replace the existing model files. It is important to ensure that the new model file is given the same name as the old one.

4.3 Adding Spells

To add a new spell to the game, there are two main steps you need to take. First, in the 3D environment, draw shapes of the new spell from different angles and in different styles, the images will be saved in the path

"C:\Users\YourUserName\AppData\LocalLow\braude project\WizardVR".

When you have finished drawing, copy the new images then open the path "WizardVR/Assets/Scripts/UnityPython/BufferImages", create a new folder and name it with the desired skill, then paste all the new images into this folder.





Second, to learn a new model, use the

"WizardVR\ModelTrainScript\trainmodel.py" file as explained in 4.2. Additional steps must be taken is to change the number of classes from 8 to 9 in

"trainmodel.py" and to add the name of the new skill to the class names list, these 2 additional steps are necessary to have a valid new model.

```
class_names = ['fireball', 'frostbeam', 'heal', 'meteor', 'others', 'shield', 'summon', 'teleport'] + new skill name
```

Then, join the path "WizardVR/Assets/Scripts/UnityPython\models" and look for the spell_recognition_model.h5 file. Replace this with the new model but ensure that the new file is given the same name.

 jub.h5	25/08/2024 22:43	H5 File	1,885 KB
 jub.h5.meta	25/08/2024 22:43	meta	1 KB
 spell_recognition_model.h5	25/08/2024 22:43	H5 File	1,885 KB
 spell_recognition_model.h5.meta	25/08/2024 22:43	meta	1 KB

Once all the steps have been completed, the spell shape can be added to the spell book within the game. The most effective method for doing so is by creating a new image that incorporates all the spells and replacing it with the canvas of the book. Search for "spells canvas" to locate the image canvas within the Unity editor in the game scene.

4.4 Adding Enemies

It is crucial to ensure that the enemy prefab has an organised animator to incorporate a new enemy. Furthermore, As stated in Section 2.5 (Challenge 4), the enemy controller must implement the IDamageable interface to ensure interaction with the game. It is also important to ensure that the nave mesh component is added to the enemy and that the target (player) is set. It is also crucial to simulate the movement of the enemy and make changes if necessary to ensure optimal performance within the game.

To incorporate enemy entities into the in-game spawn ports, it is necessary to integrate the enemy prefab into the "WizardVR/Assets/Scripts/EnemySpawner.cs" script. Additionally, the enemy prefab must be incorporated into the "WizardVR/Assets/CountEnemiesController.cs" script, which is responsible for enumerating the current number of enemies and ensuring the game's fairness.

5. Results and Further improvements

5.1 Results

The development and implementation of Spell Runner have yielded significant outcomes, blending advanced machine learning techniques with an immersive gaming experience. One of the most notable achievements is the successful integration of a Convolutional Neural Network (CNN) into the VR environment, which allows real-time recognition of hand-drawn spell shapes. This feature is the cornerstone of the game, enabling an engaging and interactive spell-casting system that reacts dynamically to player inputs.

During testing phases, the CNN model demonstrated a high degree of accuracy in recognizing spell patterns, with a prediction success rate often exceeding 95% after adequate training with the player's unique drawing style. The model's ability to fine-tune itself based on the player's inputs has led to an increasingly personalized gaming experience, where the system adapts to the nuances of each player's style. This adaptability not only enhances gameplay but also provides a sense of progression as the player's skills improve.

The adaptive feedback system in Spell Runner is designed to maintain fluid gameplay by allowing players to correct misrecognized spells, ensuring a smooth experience. The user-friendly interface, combined with polished 3D visuals and seamless communication between the Unity engine and the Python-based machine learning, creates an immersive environment. The game's challenging spellcasting system adds strategic depth and enhances overall player satisfaction.

5.2 Further Improvements

Despite the positive results, several areas for improvement have been identified, which could further enhance the functionality and user experience of Spell Runner.

Enhanced Enemy AI

While the current enemy AI provides a sufficient challenge, there is potential to develop more sophisticated enemy behaviors. Implementing more complex AI routines that allow enemies to learn from player actions or coordinate in groups could increase the game's difficulty and strategic depth. Additionally, varying enemy types with distinct abilities and weaknesses would add variety and keep gameplay fresh and engaging.

Multiplayer Mode

Introducing a multiplayer mode could significantly enhance the game. Allowing players to compete or collaborate in spellcasting challenges would add a social dimension to the game, making it more appealing to a broader audience. This could involve co-op missions, competitive duels, or even a shared spell-casting arena where players can test their skills against one another.

Visual and Environmental Enhancements

While the current visual design is functional, there is always room for aesthetic improvement. Upgrading the game's graphics to include more detailed environments, dynamic lighting, and weather effects could make the magical world of Spell Runner even more immersive. Additionally, incorporating more interactive elements within the game environment, such as destructible objects or hidden secrets, would encourage exploration and engagement.

6. References

- [1] <https://www.meta.com/help/quest/articles/headsets-and-accessories/oculus-link/set-up-link/>
- [2] Bakkes, S., Tan, C.T. and Pisan, Y., 2012, July. Personalised gaming: a motivation and overview of literature. In Proceedings of the 8th Australasian Conference on Interactive Entertainment: Playing the System (pp. 1-10).
- [3] D. Charles, M. McNeill, M. McAlister, M. Black, A. Moore, K. Stringer, J. Kucklich, and A. Kerr. Player-centered design: Player modeling and adaptive digital games. In DiGRA, pages 285–298, 2005.
- [4] M. Riedl. Scalable personalization of interactive experiences through creative automation. Computers in Entertainment (CIE), 8(4):26, 2010.
- [5] Bakkes, S., Whiteson, S., Li, G., Vişniuc, G.V., Charitos, E., Heijne, N. and Swellengrebel, A., 2014, October. Challenge balancing for personalised game spaces. In 2014 IEEE Games Media Entertainment (pp. 1-8). IEEE.
- [6] <https://www.britannica.com/technology/virtual-reality>
- [7] <https://www.meta.com/quest>
- [8] <https://www.ibm.com/topics/machine-learning>
- [9] <https://www.tensorflow.org/tutorials/images/cnn>
- [10] <https://www.meta.com/help/quest/articles/headsets-and-accessories/oculus-link/requirements-quest-link/>