

Software Engineering Department

Braude College of Engineering

Capstone Project Phase A

Spell Runner - Wizard VR



Project Code: 24-1-D-37

**Advisor:** PhD. Sulamy Moshe

MosheSu@braude.ac.il

**Students:**

Habib Ibrahim - [Habib.Ibrahim@e.braude.ac.il](mailto:Habib.Ibrahim@e.braude.ac.il)

Vitaly Rofman - [Vitaly.Rofman@e.braude.ac.il](mailto:Vitaly.Rofman@e.braude.ac.il)

Table Of Content

[Abstract 3](#_Toc165897369)

[1. Introduction 4](#_Toc165897370)

[2. Related Work 5](#_Toc165897371)

[3. Background 6](#_Toc165897372)

[3.1 VR 6](#_Toc165897373)

[3.2 Machine Learning 7](#_Toc165897374)

[3.2.1 Introduction to Machine Learning 7](#_Toc165897375)

[3.2.2 Backpropagation 7](#_Toc165897376)

[3.2.3 CNN 7](#_Toc165897377)

[3.3 Unity game engine 9](#_Toc165897378)

[4 Expected achievements 10](#_Toc165897379)

[4.1 Outcomes 10](#_Toc165897380)

[4.2 Unique features 10](#_Toc165897381)

[4.2.1 VR 10](#_Toc165897382)

[4.2.2 Casting recognition using machine learning. 10](#_Toc165897383)

[4.2.3 Personalized model for each individual player 11](#_Toc165897384)

[4.3 Criteria for success 11](#_Toc165897385)

[5. The Process 11](#_Toc165897386)

[5.1. Research - Personalized game development. 11](#_Toc165897387)

[5.1.1. Constraints and challenges 12](#_Toc165897388)

[5.2. Research – Virtual reality game development. 12](#_Toc165897389)

[5.2.1. Constraints and challenges 12](#_Toc165897390)

[5.3. Methodology and Development Process 13](#_Toc165897391)

[6. Product 14](#_Toc165897392)

[6.1. Requirements 14](#_Toc165897393)

[6.1.1. Functional 14](#_Toc165897394)

[6.1.2. Non-functional 14](#_Toc165897395)

[6.2. Architecture overview 15](#_Toc165897396)

[6.3. VR-Interfaces and in-game simulation flow 16](#_Toc165897397)

[6.4. Diagrams 18](#_Toc165897398)

[6.4.1. Use-Case Diagram 18](#_Toc165897399)

[6.4.2. Activity Diagram 19](#_Toc165897400)

[6.4.3. Sequence Diagram 20](#_Toc165897401)

[7. Verification and Evaluation 21](#_Toc165897402)

[7.1. Evaluation 21](#_Toc165897403)

[7.2. Verification 21](#_Toc165897404)

[8. References 23](#_Toc165897405)

# Abstract

Our project is to create a virtual reality game that uses machine learning to personalize the gaming experience. By adapting to each player's drawing patterns, the game aims to increase engagement and satisfaction. Developed with Unity for VR and ML algorithms, using a CNN model to learn and predict players' drawings, based on the Oculus Quest 2 VR set, our goal is to provide an enjoyable gaming experience tailored to individual players.

# Introduction

Spell Runner - Wizard VR is an innovative virtual reality (VR) game that immerses players in a magical world where they assume the role of a wizard. Spell Runner incorporates machine learning technology to interpret hand-drawn shapes made by the player through a VR headset, determining the type of spell to be cast based on these drawings.

Spell Casting System

The core gameplay mechanic of Spell Runner revolves around the spell casting system. Players apply hand gestures and movements tracked by the VR headset to draw specific shapes corresponding to spells found in a spell book. Each spell is associated with a unique draw pattern or combination of patterns, challenging players to master their reaction and accuracy.

Spell book and Spell Groups

The spell book serves as the source of magical knowledge within the game. Spells are organized into separate groups, each containing spells of similar features or effects. Players must navigate through the spell book to access different spell groups, with progression to unlock more powerful and complex spells.

Machine Learning Integration

Spell Runner incorporates machine learning algorithm to analyze and categorize the hand-drawn shapes made by players. This algorithm is trained to recognize patterns and associations between gestures and specific spells. As players continue to cast spells, the machine learning system adapts and improves its recognition accuracy, enhancing the overall gaming experience.

Adaptive Machine Learning Model

The machine learning model integrated into ‘Spell Runner’ continuously learns and adapts to the individual player's spell drawing patterns and style, enhancing the user experience over time. Initially, the model relies on a pre-trained dataset to learn the player's patterns and classify them into specific spells. However, as the player progresses through the game and casts spells repeatedly, the model begins to gather data on the player's unique drawing style, such as curvature and accuracy.

# Related Work

The concept of personalized gaming, where player models influence various components of gameplay, has been the subject of extensive research. One important aspect is the adaptation of game mechanics [1]. While games such as "Braid" and "Max Payne 3" have demonstrated mechanic adaptation based on player skill, recent studies have explored model adaptation and emergent gameplay generation.

Narrative adaptation [1] has long been studied in interactive storytelling, focusing on tailoring content to maximize player enjoyment while maintaining plot coherence. Techniques such as chaining appropriate actor actions and planning for learning objectives have been extensively researched, with systems such as "PaSSAGE" and "SSAU". In the area of music and sound adaptation [1], games such as "Banjo Kazooie" have exemplified interactive music that adapts to player actions and enhances immersion.

Artificial intelligence (AI) in games would provide satisfactory and effective game experiences for players regardless of gender, age, capabilities, or experience [2], it would allow for the creation of personalized games, where the game experience is continuously tailored to fit the individual player. Indeed, it is argued that we are now at a unique point where modern computer technology, simulation, and artificial intelligence (AI) have opened up the possibility that more can be done with regard to on-demand and just-in-time personalization [3].

One approach is published in the ‘Towards Challenge Balancing for Personalized Game Spaces’ paper, where they have enhanced the game (Mario Bros) such that its process for procedural-content generation allows the game spaces (i.e., levels) to be personalized during play of the game.[4]

Difficulty scaling techniques [1] aim to adapt game challenges to player skill dynamically. While traditional difficulty settings exist, recent research has explored advanced methods such as co-evolutionary algorithms and real-time measurement of player enjoyment to achieve balanced game experiences.

In summary, personalized gaming is at the intersection of various fields such as interactive storytelling, music/sound design, multiplayer matchmaking, and difficulty scaling. Our game will focus on using an AI model to personalize gameplay by analyzing the player's drawing or casting spells to adapt to their unique and individual profession.

# 3. Background

## 3.1 VR

“Virtual reality, the use of computer modeling and simulation that enables a person to interact with an artificial three-dimensional (3D) visual or another sensory environment. VR applications immerse the user in a computer-generated environment that simulates reality using interactive devices, which send and receive information and are worn as goggles, headsets, gloves, or body suits”.[5]

“Meta Quest is our family of immersive headsets delivering interactive experiences in 3D spaces that go beyond the physical world, allowing you to do what you love in new ways—whether it’s gaming, entertainment, fitness and wellness, travel, design or hanging out and playing virtually with friends.” [6]



Figure 1: Oculus quest 2 VR set

## 3.2 Machine Learning

### 3.2.1 Introduction to Machine Learning

“Machine learning is a branch of [artificial intelligence (AI)](https://www.ibm.com/topics/artificial-intelligence) and computer science which focuses on the use of data and algorithms to imitate the way that humans learn, gradually improving its accuracy.”[6]

The process begins with the collection and preprocessing of data, which involves cleaning, transforming, and organizing information to prepare it for use. Once the data is ready, a machine learning model is trained on this data, using various algorithms and techniques to identify patterns and make predictions based on the input. After training, the model is then tested against new data to evaluate its accuracy and performance. Predictions are made by the model based on the input it receives, which may include data that has not been seen before. The model then receives feedback in the form of corrections or adjustments to its predictions, allowing it to learn and adapt to new data and improve its performance over time. This continuous learning process is a key aspect of machine learning, enabling models to become more accurate and effective as they are exposed to larger and more diverse datasets.

### 3.2.2 Backpropagation

Backpropagation is a method for teaching a neural network how to make correct predictions by adjusting the connections between its neurons based on the errors it makes. It calculates the error at each step and uses this information to adjust the weights of the network, helping it learn from its mistakes and improve over time. In simpler terms, backpropagation is a way for a neural network to learn and correct itself by analyzing the errors it makes and adjusting its internal connections accordingly.

### 3.2.3 CNN

A Convolutional Neural Network (CNN) is a specific type of neural network commonly used in image recognition and processing tasks. CNNs are designed to learn features automatically and adaptively from images, making them particularly effective for identifying patterns and objects within visual data. At a high level, a CNN consists of multiple layers that process image data in a hierarchical manner.

The first layer typically applies filters to the input image to extract basic features such as edges and lines. These features are then passed through subsequent layers, each of which performs increasingly complex transformations on the data. One key feature of CNNs is their use of local connections and shared weights. This means that each neuron in each layer only receives input from a small, local region of the previous layer.

As data passes through the layers of a CNN, more complex features are extracted and identified. The final layer of the network typically consists of fully connected neurons that make the final prediction based on the extracted features.

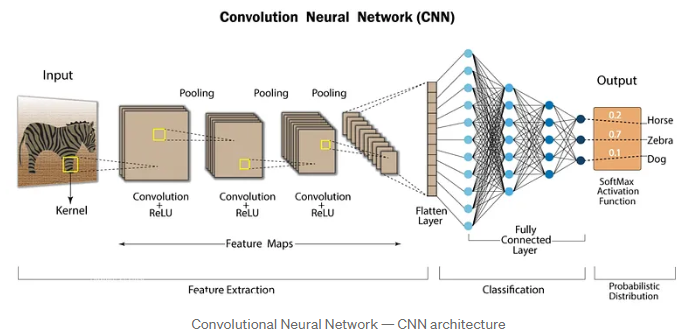


Figure 2: CNN training process

## 3.3 Unity game engine

Unity is a user-friendly game development platform that allows you to create games and interactive experiences for various devices. It provides an intuitive editor, scripting in C#, and a vast asset store for graphics, sound, and more. With built-in physics and networking capabilities, Unity is ideal for academic projects requiring game development or interactive simulations. Its performance optimization tools ensure smooth experiences across different platforms, making it a popular choice for educational and research purposes.

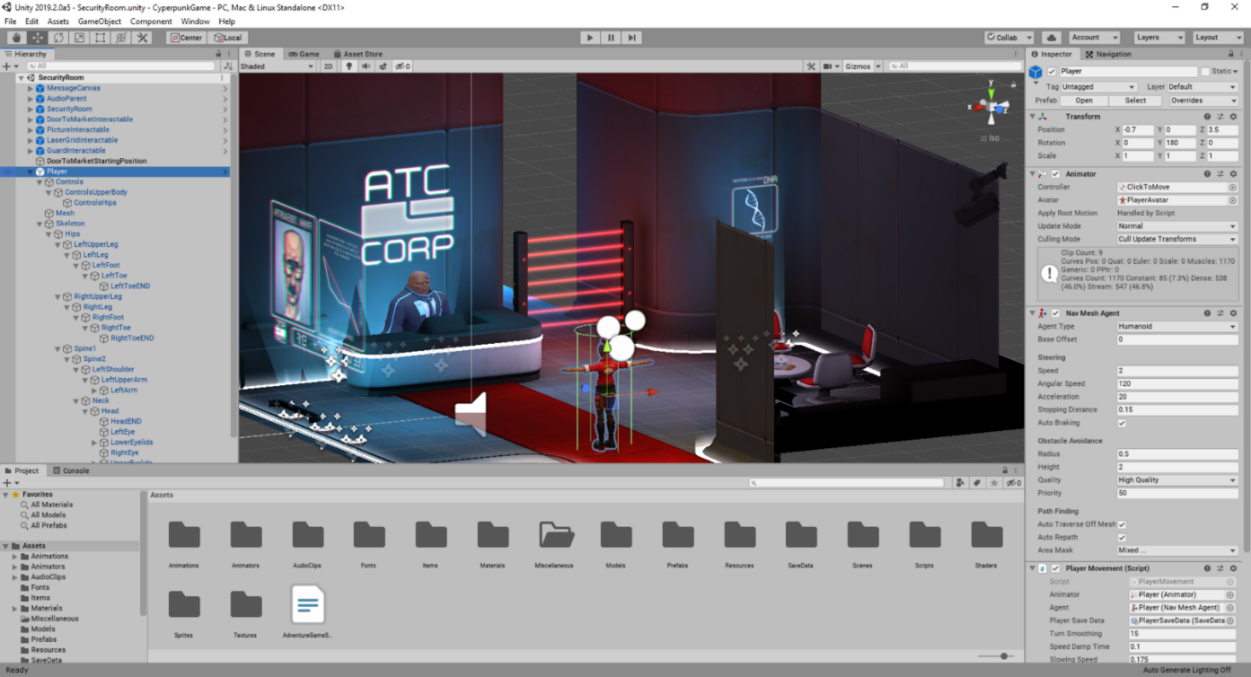


Figure 3: Unity engine environment

# 4 Expected achievements

## 4.1 Outcomes

The primary objective of this project is to develop personalized gameplay, featuring a machine learning model that learns and adapts to each player's unique drawing style for casting spells.

By continuously refining the model based on individual players, we will create more personalized and engaging gameplay experiences with improved accuracy in recognizing and classifying unique spell patterns. This advanced level of customization enhances the overall player experience, adopts a strong sense of personal investment, and encourages ongoing exploration within the game world.

The main challenge of this project is to balance fast spell recognition times with high accuracy. Our goal is to deliver a game that overcomes all challenges and enhances user experience.

## 4.2 Unique features

### 4.2.1 VR

VR technology delivers an unmatched gaming experience, where players can immerse themselves fully in a rich and interactive 3D environment, exploring, interacting, and experiencing the game world in ways previously unimaginable.

What sets our VR implementation apart is its integration with Unity3D, a well-known game development engine known for its robust 3D graphics capabilities. Unity3D stands out due to its user-friendly interface, extensive library of assets, and powerful tools for creating immersive environments.

### 4.2.2 Casting recognition using machine learning.

Our game will feature a unique casting recognition method using Convolutional Neural Networks (CNNs) and machine learning, which will improve player interaction by allowing them to cast spells and perform actions. This innovative feature will enhance immersion, engagement, and interactivity, providing players with a dynamic and personalized gaming experience unlike anything they've experienced before.

### 4.2.3 Personalized model for each individual player

Our game implements a personalized model for each player using machine learning. By analyzing gameplay casting patterns, style, and shape, we tailor the gameplay experience to suit individual styles. This approach ensures dynamic adaptation and raises deeper player engagement, setting our game apart with its innovative and immersive features.

## 4.3 Criteria for success

* Responsive interface: The game interface should provide timely and clear feedback to players, acknowledging successful actions and alerting them to errors.
* Engaging Gameplay: The game should provide compelling mechanics and challenges to keep players engaged.
* Technical Stability: Smooth performance and minimal bugs that ensure the game runs smoothly without interruptions increase player satisfaction and overall enjoyment.
* Clear Goals: Players should have well-defined goals and objectives to guide their progress.
* Model Efficiency: The ML algorithm should facilitate balanced gameplay by assisting players as the model learns the pattern of drawing and casting spells.

Our main criteria for success in game development are ensuring player satisfaction, engagement, technical stability, balanced gameplay, and adaptation to player behavior.

# 5. The Process

## 5.1. Research - Personalized game development.

Our research on personalized games began with a thorough investigation of the benefits of using this methodology in game development and its reliability. While the development of personalized games can be tricky and complicated, research showed that AI could make it easier and more understandable. Through our exploration of the combination between AI and game development, we explored the benefits, challenges, and usability of it, and we found that personalization models can increase player immersion by adapting gameplay mechanics to individual preferences, despite the challenges of deploying in-game personalized models that will leverage user experience and satisfaction. In addition to a variety of models available today, a specific AI model can fit the idea of personalization that developers are seeking, for example, in our case we will use CNN model for image recognition and classification to learn the drawing pattern for each player.

### 5.1.1. Constraints and challenges

During our research, we faced some challenges that make it a good starting point: there's the complexity of implementing AI-driven personalization, which requires significant expertise in both AI and game development. We decided to explore the capabilities of several models to select the one that best fits our needs.

Integrating AI-driven personalization into virtual reality (VR) adds another layer of complexity, as data from VR must be efficiently transferred to the AI model to ensure best integration and optimal performance in learning player patterns and behaviors. This includes storing checkpoints in the model after each training iteration.

Finally, the most challenging element is updating the model quickly and efficiently, which will take our time to study this case in depth to ensure the best performance for the game.

## 5.2. Research – Virtual reality game development.

Our research focuses on Virtual Reality (VR) games within the game development market. We explore their evolution, impact, and prospects by technological advancements, and user preferences. Our aim is to uncover opportunities and challenges in VR game development.

VR technology is gaining importance in various industries, including space experiments, employee training, and manufacturing. Its applications range from imagining engineering systems to enhancing training modules and simulating complex processes for improved efficiency and knowledge.

Recognizing the broad range of applications and benefits of virtual reality (VR) across industries, we decided to apply this technology in game development. The immersive nature of VR aligns with our goal of creating engaging and interactive experiences for users, which drove our decision to develop a VR game.

### 5.2.1. Constraints and challenges

Developing a VR game in Unity presents unique constraints and challenges. Although Unity provides robust tools for VR development, optimizing performance for different VR platforms can be difficult. Unity's cross-platform capabilities require careful implementation to ensure seamless experiences across devices. Our approach to overcoming these challenges involves testing on Quest 2 set and utilizing fixed VR features to create a fully immersive experience.



Figure 4: Virtual reality is used in employee training.

## 5.3. Methodology and Development Process

We chose the Agile methodology for development because of its iterative and flexible nature in project management and software development. It promotes teamwork and flexibility, which enhances the development process.

Our workflow will be divided it into numerous stages:

1. Designing AI model architecture and implementing in appropriate programing language, including testing the model, and train a basic version of it.
2. Scheming game graphics and the virtual environment we will use.
3. Developing the environment and main gameplay elements
4. Integrating the AI model into the game while ensuring compatibility.
5. Testing the general functionality of the game and evaluating the performance of the model.
6. Gathering feedback and refining the game through user testing, involving different players.
7. Performing user experience test to verify that the game meets all specified requirements and operates correctly.
8. Deploying the initial version of the game to the public.

Each stage will be given sufficient time to ensure thorough completion. Following this methodology, we will not proceed to a new stage until the previous one has been acceptably completed to ensure an orderly and thorough development process.

# 6. Product

|  |  |
| --- | --- |
| 1 | The system must be adjusted to virtual reality headset |
| 2 | The system must be responsive |
| 3 | The system graphics must suite for virtual environment |
| 4 | The system learns the players drawing patterns |
| 5 | The system predicts different spell shapes correctly |
| 6 | The system allows the player to navigate across the virtual word |

## 6.1. Requirements

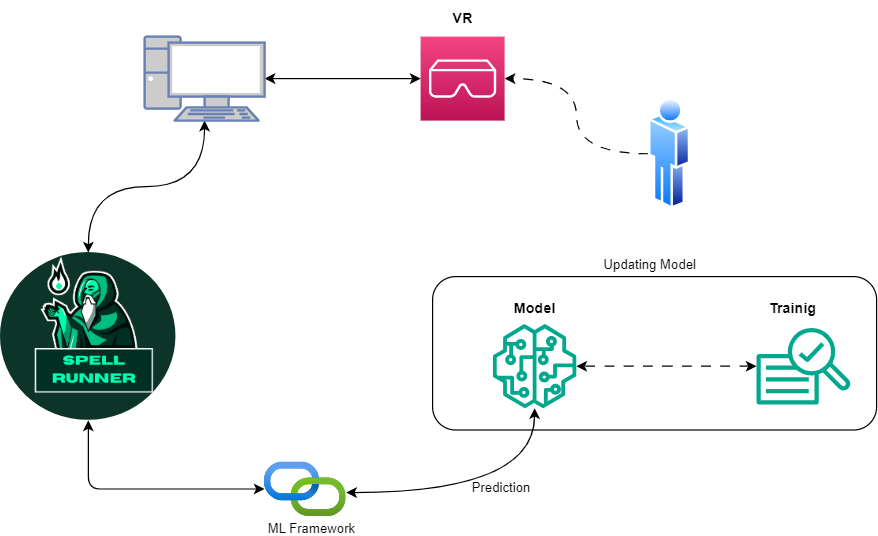
### 6.1.1. Functional

### 6.1.2. Non-functional

|  |  |
| --- | --- |
| 1 | The game must provide an immersive virtual reality experience |
| 2 | The game should maintain stable performance |
| 3 | The game should have minimal loading times between scenes |
| 4 | The game should feature immersive audio effects |
| 5 | The game interface should be user friendly and easy to understand |
| 6 | The game should be challenging. |

## 6.2. Architecture overview

To start gameplay, users must first connect their VR headset to the computer. Once this is done, they can then run the game and begin playing. Meanwhile, on the back-end side, each new input from the user that will be inserted into the game (a new shape), will be sent to the model, which will predict it and learn more about the user's drawing patterns.



## 6.3. VR-Interfaces and in-game simulation flow

Upon entering the game, players will receive clear instructions on how to navigate and interact with the interface, ensuring an easy and intuitive experience without confusion. The user will hold the game accessory with both hands to initiate gameplay interaction with the interface.



After the game begun, player should start navigating the map searching for enemies and attack them to survive and collect point.



Players will utilize a spell book held in their left hand to cast various powers for attacking enemies. Spells are constructed through drawing shapes like circles, triangles, and others, providing intuitive interaction and diverse gameplay experiences.

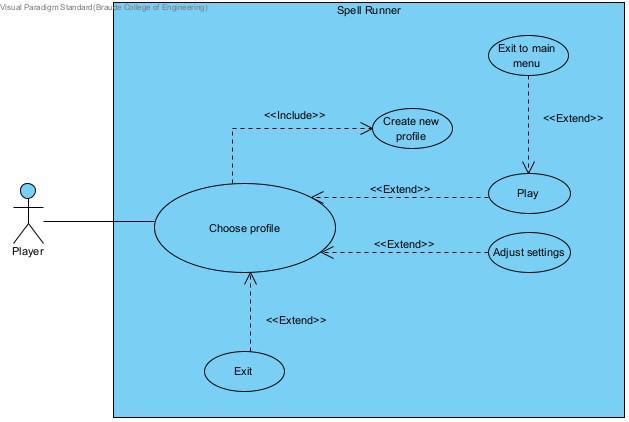


Players choose which spell to cast and begin drawing it. The game's ML model learns the player's behavior and drawing patterns, converting them into specific spell for casting.

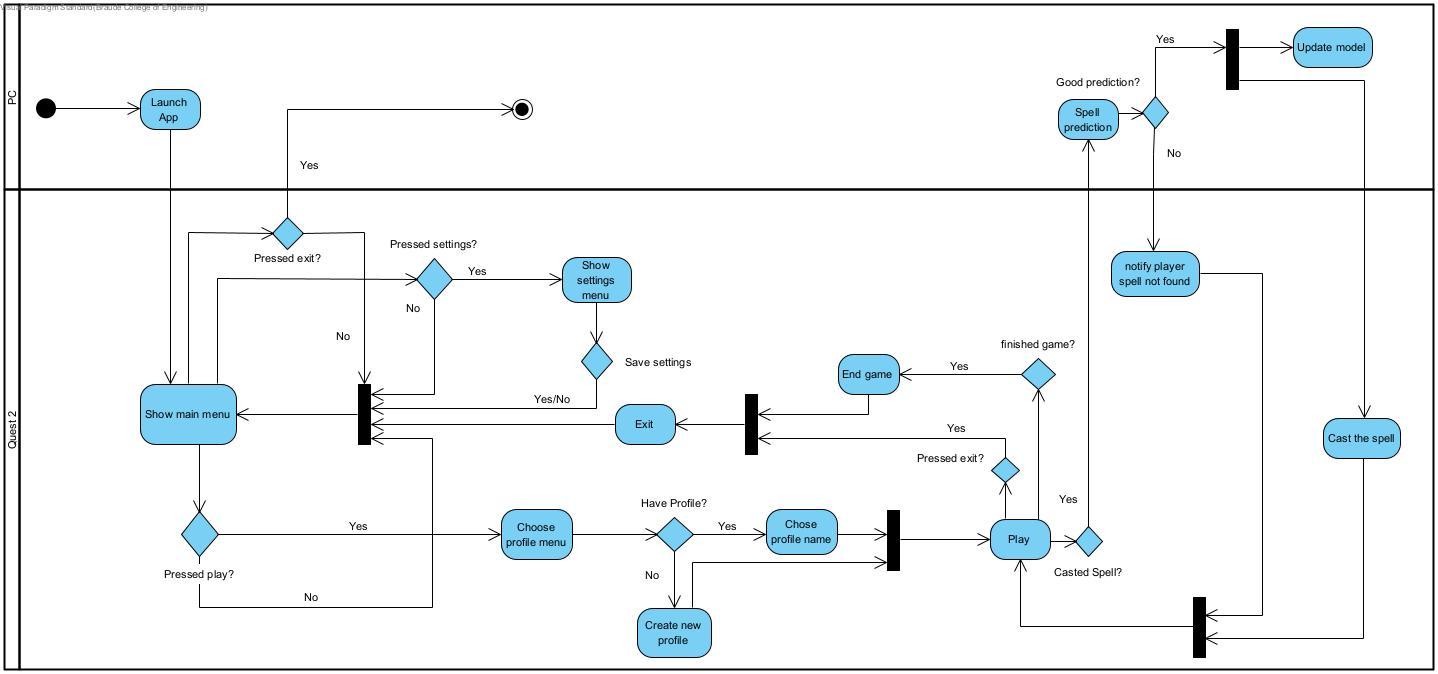


## 6.4. Diagrams

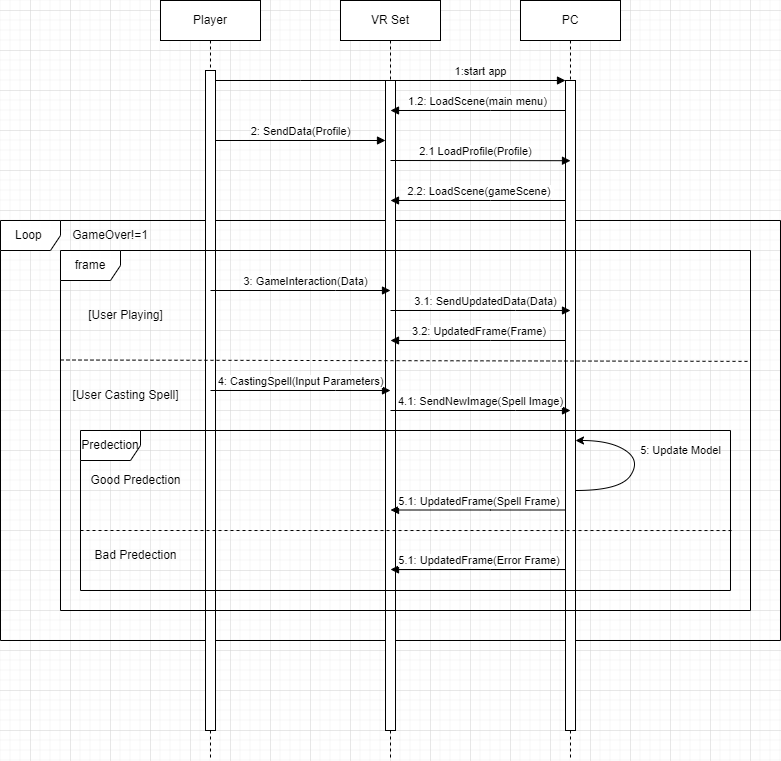
### 6.4.1. Use-Case Diagram



### 6.4.2. Activity Diagram



### 6.4.3. Sequence Diagram



# 7. Verification and Evaluation

## 7.1. Evaluation

Game evaluation is crucial to ensure that our game meets the desired standards of quality, enjoyment, and effectiveness. By assessing various aspects of the game, we can determine its strengths, weaknesses, and overall potential for success upon release. Here's an outline of our evaluation plan:

1. Design Estimation: Review the visual and audio design elements, such as aesthetics, user interface, sound effects, music, and character/world design. Ensure they are immersive, cohesive, and align with the game's theme and narrative.
2. Player in-game feedback: Conduct testing with a sample group of players to observe their interactions with the game. Collect feedback on their experience, engagement level, and suggestions for improvement.
3. Performance Evaluation: Test the game's performance to ensure smooth gameplay, minimal bugs, and optimized performance.

## 7.2. Verification

We are planning to use the Unity Test Framework (UTF) because it is a powerful tool for testing Unity projects, allowing us to create and run automated tests to verify the behavior and functionality of our code. With UTF, we can write test scripts directly within the Unity Editor and execute them to ensure that our game logic, components, and systems are working correctly.

Some of the main features of UTF that it provides:

1. Integrated Testing: UTF is integrated directly into the Unity Editor, making it easy to create, edit, and run tests within the familiar Unity environment.
2. Automated Testing: Developers can automate the execution of tests, allowing for rapid and efficient verification of game functionality.
3. Test Runner: UTF provides a Test Runner window where developers can organize and execute their test suites, view test results, and debug failures.

Here are some examples of tests we are planning to do:

|  |  |  |  |
| --- | --- | --- | --- |
| Test no. | Module | Tested Function | Expected Result |
| 1 | Player Controller | MovePlayer(direction) | Verifies that the player character moves in the correct direction when the MovePlayer function is called with a specified direction vector. |
| 2 | EnemyAI | ChasePlayer(playerPosition) | Tests if the enemy AI properly follows and chases the player character when given the player's current position as a parameter. |
| 3 | Game Manager | StartGame() | Ensures that the game initializes correctly and begins running when the StartGame function is called, initializing game elements such as player health, score, and enemy spawn points. |
| 4 | SpellCastingSystem | CastSpell(spell) | Validates that the player can cast spells of different types (e.g., fireball, lightning bolt) and that the appropriate spell effect is triggered when the CastSpell function is invoked. |
| 5 | UIManager | DisplayGameOverScreen() | Checks if the game over screen is correctly displayed when the player character's health reaches zero, providing options to restart the game or return to the main menu. |
| 6 | SoundManager | PlaySoundEffect(sound) | Tests whether sound effects are played correctly in the game when triggered by specific events, such as player actions or enemy encounters. |
| 7 | MLModelProcessor | ProcessImage(image) | This test unit verifies the MLModelProcessor's ability to correctly process an input image and return the predicted result |

# 8. References

[1] 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).

[2] 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.

[3] M. Riedl. Scalable personalization of interactive experiences through creative automation. Computers in Entertainment (CIE), 8(4):26, 2010.

[4] 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.

*[5] Li, H., 2022. Convolutional Neural Network-Based Virtual Reality Real-Time Interactive System Design for Unity3D. Computational Intelligence and Neuroscience, 2022.*

*[6]*[*https://www.britannica.com/technology/virtual-reality*](https://www.britannica.com/technology/virtual-reality)

*[7]* [*https://www.meta.com/quest*](https://www.meta.com/quest)

*[8]* [*https://www.ibm.com/topics/machine-learning*](https://www.ibm.com/topics/machine-learning)

*[9] https://www.tensorflow.org/tutorials/images/cnn*