



Software Engineering Department
ORT Braude College

Capstone Project Phase A – 61998

House of Riddles using AI and ML

Supervisors:

Moshe SulamY

MosheSu@braude.ac.il

Students:

Seren Hanany- seren.hanany@e.braude.ac.il

Maisalon Safory- maisalon.safory@e.braude.ac.il

Abstract

The proposed game "House of Riddles using AI and ML" offers players a multi-faceted experience designed to engage their brains and foster cognitive stimulation.

The game features a complex system with multiple puzzles locked behind a secure frame, and players compete to open all the cells and solve all the puzzles before their friends. This competitive element adds another layer of excitement and motivation, as players aim to outwit their friends and achieve the highest score or fastest completion time.

Artificial intelligence enhances the gaming experience by providing personalized recommendations, subtle hints and dynamic difficulty adjustment to keep players interested and engaged. The integration of ML within the game not only helps improve the player's aim but also provides an engaging method for learning.

The goal is to offer a gaming experience that adds value by stimulating cognitive activity and providing a more engaging alternative to traditional idle games.

Table of Contents

1. Introduction	4
2. Literature review	5
3. Expected Achievements	7
3.1 Outcomes	7
3.2 Unique Features	8
3.2.1 ML - Player behavior analysis	8
3.2.2 Unity Engine – Modeling a 3D environment	8
3.2.3 Personalized Player Experience	9
3.3 Criteria for Success	9
4. Engineering Process	10
4.1 Research- Unity and game development	10
4.1.1 Constraints and Challenges	10
4.1.2 Conclusions from Research	11
4.2 Research- Machine Learning	11
4.2.1 Constraints and Challenges - Machine Learning	12
4.3 Methodology and Development Process	12
5. Product	14
5.1 Requirements	14
5.2 Architecture overview	15
5.2.1 Machine learning Application	16
5.2.2 Scenes	17
5.3 Diagrams	21
5.3.1 Use Case	21
5.3.2 Activity Diagram	22
5.3.3 Class Diagram	22
6. Verification and Evaluation	23
6.1 Evaluation	23
6.2 Verification	24
7. References	26

1.Introduction

Our project introduces a puzzle game developed in UNITY employing machine learning techniques, designed to engage players' cognitive faculties and adapt challenges to their behaviors and preferences. It features a sophisticated framework with numerous puzzles inaccessible until unlocked, where players strive to solve them faster than their peers. Utilizing machine learning, the game offers personalized recommendations, subtle hints, and dynamic difficulty adjustments to sustain player interest and engagement. By fostering cognitive activity and presenting a compelling alternative to traditional idle games, our aim is to enrich the gaming experience while addressing common challenges encountered in puzzle-solving games, including fixed difficulty levels, the need for personalized experiences, overcoming difficult puzzles, and ensuring player engagement and retention, thus delivering a more satisfying gaming experience overall.

Machine learning (ML) offers solutions to these problems by leveraging data-driven insights to tailor the gaming experience to individual players. For difficulty adaptation, ML algorithms analyze player behavior and preferences, adjusting the challenges dynamically to ensure an appropriate level of difficulty. This solves the problem of fixed difficulty levels by providing a personalized and engaging experience. ML also enables personalized recommendations and hints based on player profiles, addressing the need for a tailored experience and enhancing player connection to the game. Additionally, ML algorithms can detect patterns indicating player frustration or difficulty, offering subtle hints and assistance to help players overcome obstacles and progress smoothly. By continuously analyzing engagement metrics, ML optimizes gameplay elements to maximize player engagement and retention, ultimately leading to increased satisfaction and long-term player commitment. ML's ability to adapt, personalize, and optimize the gaming experience makes it an effective solution to the challenges faced in puzzle-solving games.

Furthermore, there are different types of players who are the main stakeholders in the game, each with unique characteristics, preferences and play styles. Some players may be casual gamers who enjoy solving puzzles in their free time without seeking intense challenges. These players prefer relaxed gameplay with straightforward puzzles and may appreciate gentle hints and guidance to help them progress smoothly. On the other hand, there are hardcore gamers who thrive on difficult challenges and complex puzzles that test their skills and problem-solving abilities. These players may prefer minimal assistance and enjoy the satisfaction of overcoming tough obstacles on their own. Additionally, there are completionists who aim to fully explore and master every aspect of the game, meticulously solving every puzzle and

uncovering every secret. They may appreciate additional content, such as bonus levels or hidden challenges, to extend their gameplay experience. By recognizing and accommodating these different player types, your game can offer a more inclusive and enjoyable experience for all players, regardless of their preferences and skill levels.

To tackle these challenges, our project utilizes Unity and machine learning to craft a dynamic puzzle game that tests players' cognitive abilities, adjusting to their preferences and behaviors. With a range of puzzles locked within a secure framework, players vie to unlock and solve them faster than their peers. Machine learning algorithms offer tailored hints and recommendations, enriching gameplay and maintaining player interest. By addressing common hurdles like fixed difficulty levels and the need for personalized experiences, our game promises a fulfilling and immersive gaming experience, stimulating cognitive activity and offering an engaging alternative to traditional idle games.

2. Literature review

Machine learning provides computer systems with the ability to learn and improve automatic experience. Machine learning can be defined as a system or "machine" that can learn from its experiences without following specific written instructions. This is done through the use of algorithms and models so that the system can analyze patterns in the data. It has seen rapid development in recent years, driven by advances in algorithms, computational power, and data availability. Researchers and developers are constantly exploring new techniques and applications, including deep learning, reinforcement learning, and natural language processing.

The Unity game engine was developed by Unity Technologies in Denmark. Unity combines a custom rendering engine with the nVidia PhysX physics engine and Mono, Microsoft's open-source application. NET libraries. The benefits of using Unity are many. So the key features of Unity that make it a great robot simulation engine. Problem-solving within this domain involves tasks such as designing game mechanics, balancing gameplay, optimizing performance, and addressing bugs, utilizing methods like prototyping, playtesting, user feedback analysis, and continuous integration/development. Various tools aid in these endeavors; game engines like Unity offer integrated development environments, asset management, visual scripting, and deployment utilities, while version control systems like Git facilitate collaborative development and version management. Resources for game developers span diverse mediums, including books such as "The Art of Game Design: A Book of Lenses" by Jesse Schell, online tutorials, game design

blogs, and academic papers covering topics like game design patterns and level design principles.

Unity is a cross-platform game engine designed to support and develop 2D and 3D video games, computer simulations, virtual reality, consoles and mobile devices platform, equipped with a visual editor, C# scripting API, physics engine, asset store and extensive support for different platforms and devices. Troubleshooting within Unity includes tasks such as scripting game mechanics, integrating AI behaviors, optimizing performance, and integrating third-party plugins, using methods such as object-oriented programming, component-based architecture, and event-driven programming. The Unity Asset Store serves as a treasure trove of assets, plugins, and tools, and includes AI solutions, visual scripting tools like Bolt, and performance optimization utilities. The collaboration is made possible through the integration of Unity Collaborate and Git, which enables collaborative development and version control. A wealth of literary resources, including Unity documentation, guides, forums, and books such as "Unity in Action" by Joe Hawking, along with online courses and communities, provide ample guidance and support for developers harnessing Unity for their game projects.

Git serves as a distributed version control system pivotal for monitoring changes within code repositories, enabling seamless collaboration, version management, merging alterations, and comprehensive project history tracking. Problem-solving within Git entails employing strategies for branching, resolving conflicts, conducting code reviews, and implementing continuous integration practices, utilizing techniques such as feature branching, pull requests, maintaining commit hygiene, and integrating automated testing. Supported by an extensive array of development platforms, IDEs, and hosting services like GitHub, GitLab, and Bitbucket, Git is bolstered by features facilitating repository hosting, issue tracking, code review, and continuous integration. Various literature resources including documentation, tutorials, books such as "Pro Git" by Scott Chacon and Ben Straub, and online courses comprehensively cover Git's functionalities, best practices in version control, and effective Git workflows.

Puzzle design in games encompasses a diverse array of challenges, from traditional logic puzzles to interactive elements seamlessly integrated into gameplay mechanics, requiring a blend of creativity, problem-solving prowess, and careful consideration of player skill levels and preferences.

Problem-solving within puzzle design entails tasks such as defining objectives, crafting challenges, iteratively refining designs based on playtesting feedback, and ensuring a balanced difficulty progression, utilizing methods like paper prototyping, level editors, and puzzle generation algorithms. Game engines like Unity offer comprehensive tool sets for puzzle

creation and scripting, including level editors, visual scripting capabilities, and asset libraries, while specialized third-party tools and frameworks such as PuzzleScript, Puzzlescript.net, and ProBuilder cater specifically to puzzle generation needs. A wealth of literature resources, including books like "Challenges for Game Designers" by Brenda Brathwaite and Ian Schreiber, as well as academic papers and articles covering puzzle design principles, player engagement, and difficulty balancing, provide valuable insights and guidance for aspiring puzzle designers seeking to hone their craft.

3. Expected Achievements

3.1 Outcomes

Our project aims to advance exposure to puzzle-solving and adapt the learning process to suit diverse learners, encompassing individuals from various fields and ages. This entails crafting a wide array of puzzles that challenge problem-solving skills in diverse ways, catering to different learning preferences and cognitive styles. To achieve this, we'll integrate features like adjustable difficulty levels, hint systems, and multiple progression paths, accommodating players with varying skill levels and learning needs¹. Our design will prioritize inclusivity, ensuring clarity in game mechanics, instructions, and feedback for players of diverse backgrounds, including those with disabilities or language barriers. Additionally, we'll seamlessly embed educational content into the game environment, offering opportunities for players to learn and apply knowledge while solving puzzles. By focusing on these achievements, our game will not only deliver an engaging gaming experience but also serve as a valuable educational resource accessible to learners of all ages and backgrounds.

¹https://www.researchgate.net/profile/Jeffrey-Craighead/publication/265284198_Using_the_Unity_Game_Engine_to_Develop_SARGE_A_Case_Study/links/55d33fdf08aec1b0429f32f4/Using-the-Unity-Game-Engine-to-Develop-SARGE-A-Case-Study.pdf

3.2 Unique Features

3.2.1 ML - Player behavior analysis

Machine learning enhances player behavior analysis in the game by providing personalized challenges, dynamically adjusting difficulty levels, offering adaptive hints, and enabling predictive analytics². Through continuous monitoring of player performance and preferences, machine learning algorithms tailor challenges to each player's skill level and gameplay style, ensuring an engaging experience. The system adapts the difficulty of puzzles based on player progress, offers subtle hints when needed, and predicts future behavior to anticipate player needs, ultimately enhancing the overall gaming experience.

3.2.2 Unity Engine – Modeling a 3D environment

Creating a captivating 3D environment is pivotal to our game's success. We must carefully choose the setting that best compliments our post-apocalyptic narrative and meticulously craft it using Unity, paying close attention to lighting, object placement, and scale to enhance immersion³. Moreover, optimizing our environment to run smoothly within hardware constraints are crucial. Balancing the number of rendered objects to maintain a stable framerate is essential to prevent user discomfort like nausea or motion sickness, ensuring an enjoyable gaming experience.

²https://www.theseus.fi/bitstream/handle/10024/496604/Breugelmans_Arno.pdf?sequence=2&isAllowed=y

³<https://isprs-annals.copernicus.org/articles/IV-4-W4/161/2017/isprs-annals-IV-4-W4-161-2017.pdf>

3.2.3 Personalized Player Experience

The integration of machine learning algorithms to analyze player behavior and preferences is another key achievement. This system should tailor challenges, provide hints, and adjust gameplay elements to suit individual players, enhancing immersion and satisfaction.

3.2.4 Improve mental skills

Creating a dynamic puzzle game that not only activates the mind but also adjusts its challenges based on player behavior and preferences. The game features a complex system with multiple puzzles locked behind a secured frame, and players compete to open all cells and solve all puzzles before their peers.

3.3 Criteria for Success

1. Clear Guidance for Flexible Gameplay: Helping Players Understand and Adapt to the Game.
2. A definitive game that can be used for learning while entertaining.
3. Creating an intuitive and reliable 3D scene.
4. Incentivizing Continued Play: Earn Houses for Each Victory, Accumulating Points to Aid Progress in Future Puzzles.
5. We aim to leverage Machine Learning to ensure optimal functionality, providing timely and engaging prompts that sustain player enthusiasm throughout the gaming experience.

4.Engineering Process

4.1 Research- Unity and game development

Regarding expanding our understanding of the game, we focused on answering the following questions:

- How does the game affect the expansion of the knowledge of the players?
- Who are the stakeholders?
- Which age group is most eager to play?
- Bottlenecks exist today in providing rapid screening and assessment to our game.
- What options exist for collecting puzzles that are not technology based and how reliable and relevant they are in today's world.

To tackle these questions and expand our knowledge, we used various resources, from scientific papers and articles to videos. After going through the available resources, we met to discuss our findings and share the main points we should focus on when developing the app. Some of the conclusions we reached while reading about puzzle games when it comes to software development, is that the puzzle needs to be adjusted to the age of the player himself so that he gets excited about the game and continues to play in the future.

Therefore, the suitability of puzzle games for a certain age group depends on the individual's cognitive development, interests and preferences. Some people may enjoy puzzles from a very young age, while others may not find them engaging until later in life. The key is to find puzzles that match the skill level of the player and provide an appropriate level of challenge and enjoyment.

4.1.1 Constraints and Challenges

Learning how to use Unity presents several constraints and challenges, primarily due to the platform's complexity and expansive feature set. Novice developers may find themselves overwhelmed by the multitude of tools, functions, and scripting languages within Unity's ecosystem.

Navigating this steep learning curve requires dedication, patience, and a significant investment of time to grasp the fundamentals of game development and Unity's specific workflows.

Additionally, keeping pace with Unity's frequent updates and evolving technologies can pose a challenge, as developers must continuously adapt their skills and workflows to leverage new features effectively. Moreover, troubleshooting issues and debugging errors within Unity projects can be arduous, particularly for those without prior programming experience. Despite these challenges, overcoming the learning curve in Unity can lead to rewarding opportunities for creative expression and innovation in game development.

4.1.2 Conclusions from Research

The conclusions are that the suitability of puzzle games for a certain age group depends on the individual's cognitive development, interests and preferences. Some people may enjoy puzzles from a very young age, while others may not find them engaging until later in life. The key is to find puzzles that match the skill level of the player and provide an appropriate level of challenge and enjoyment.

4.2 Research- Machine Learning

As we delved into the world of machine learning algorithms, we quickly realized the vast array of options available, each tailored to different tasks and problem areas. However, as our journey progressed, we found ourselves drawn into the realm of reinforcement learning. This decision was partly influenced by the Unity Machine Learning Agents (ML-Agents) toolset, which prioritizes reinforcement learning algorithms. With its seamless integration into Unity projects and support for various reinforcement learning approaches. So the agents are modeled in the environment and receive rewards based on their actions⁴.

4

https://books.google.co.il/books?hl=ar&lr=&id=OMNiDwAAQBAJ&oi=fnd&pg=PP1&dq=Reinforcement+Learning+Algorithms+with+unity+game&ots=CkBONy-A92&sig=OHdCTANtyNth_N5-Auc_sdUXZq8&redir_esc=y#v=onepage&q=Reinforcement%20Learning%20Algorithms%20with%20unity%20game&f=false

4.2.1 Constraints and Challenges - Machine Learning

- **Unity Integration:** Understanding how to integrate machine learning algorithms within Unity's ecosystem can be a challenge for newcomers. This includes installing and setting up the appropriate plugins, understanding how to interface with Unity's game objects and environments, and managing the interaction between machine learning agents and the Unity environment.
- **Unity ML-Agents:** While Unity ML-Agents simplifies the process of integrating machine learning into Unity projects, learning how to use the ML-Agents toolkit effectively still requires familiarity with reinforcement learning concepts and algorithms. Understanding how to configure and train agents, design reward functions, and interpret training results are important skills to develop.
- **Performance Optimization:** Ensuring that machine learning algorithms run efficiently within Unity's runtime environment is crucial, especially for real-time applications such as games. Optimizing computational performance, memory usage, and runtime efficiency while maintaining smooth gameplay can be a challenging task.
- **Unity-Specific Challenges:** Unity's development environment may present its own unique challenges when working with machine learning algorithms, such as compatibility issues with specific versions of Unity, platform-specific considerations (e.g., mobile vs. PC), or limitations imposed by Unity's scripting API.

4.3 Methodology and Development Process

For our development approach, we opted for the Agile methodology, which we believe aligns well with our specific needs. By adopting an iterative approach, we can break down our feature delivery into manageable components, allowing for maximum adaptability to changes. Our development process will be segmented into:

- Building the scene using the Unity3D engine.

- Adding tracking and movement in the scene.
- Adding calculations to measure the interpersonal space of a user with the environment.
- Adding an option to store these calculations.
- Checking the correlations of our data with questionnaires.
- Applying changes to any problem that arises as a result of the results.
- Creating a web-based interface that will be presented to the screening initiators.

After each iteration, we'll conduct an evaluation to assess our progress, making any necessary adjustments before initiating the subsequent cycle. Our priority is to provide functional software that incorporates input from users who will participate in testing our product.

5.Product

5.1 Requirements

Functional

1	The system should choose puzzles dynamically, adjusting the complexity based on player behavior and preferences.
2	The system should track players' progress, including completed puzzles and overall game achievements.
3	The system should analyze player behavior to provide personalized recommendations and hints.
4	The system stores data in a database
5	The system should have a secure mechanism for locking and unlocking puzzles to ensure fair gameplay.
6	The system should support multiplayer functionality, allowing players to compete against each other in real-time or asynchronously.
7	The system should collect and analyze data on player interactions, puzzle completion rates, and overall engagement.

Non-functional

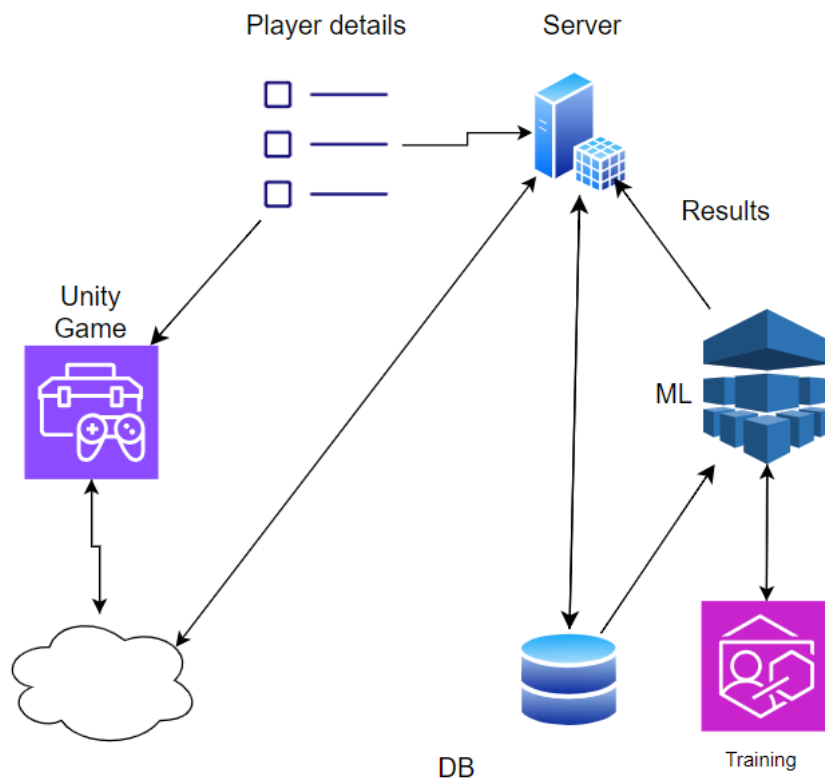
1	Easy to use graphical interface for the user.
2	Players should be able to navigate between different puzzles, levels and game features with ease.
3	Different types of puzzles should be available, offering variety and challenging gameplay.
4	The difficulty of puzzles should be dynamically adjusted based on player performance to maintain engagement.
5	Access to new puzzles should be gated behind previous achievements or challenges.
6	Reports and analytics should be available to help developers understand player behavior and improve the game experience over time.

5.2 Architecture overview

Our architecture consists of several key components:

- **Input:** Receives data from the environment.
- **Preprocessing:** Cleans and formats data for learning.
- **Neural Network:** Core component for pattern recognition.
- **Activation Functions:** Introduces non-linearity for complex learning.
- **Output:** Produces actions or predictions.
- **Loss Function:** Measures performance during training.
- **Optimizer:** Adjusts model parameters to minimize loss.
- **Training Pipeline:** Process for data collection, training, and evaluation.
- **Memory and State:** Retains information for decision-making.
- **Deployment:** Putting the trained model into action for real-time use.

5.2.1 Machine learning Application



Our application will be using the MVC architecture in the following method:

Model

The database will receive data like distance calculations and user details from the Controller. Its main role is to store this information securely.

Controller

The controller layer will dynamically handle scene creation and management based on user interactions, utilizing Unity3D Engine for tasks like texture rendering, lighting computations, and object manipulation. It will also interact with the Model layer to store any necessary data.

View

The View layer displays the user interface, including navigation buttons, players, and objects. Users interact with this interface, and the Controller manages the input and logic for the scene accordingly.

5.2.2 Scenes

We create a variety of social situations for users to experience, from pleasant and familiar settings such as peaceful homes to more challenging environments such as streets. Our goal is to offer a wide variety of scenarios that help users build confidence and adaptability in different social contexts.

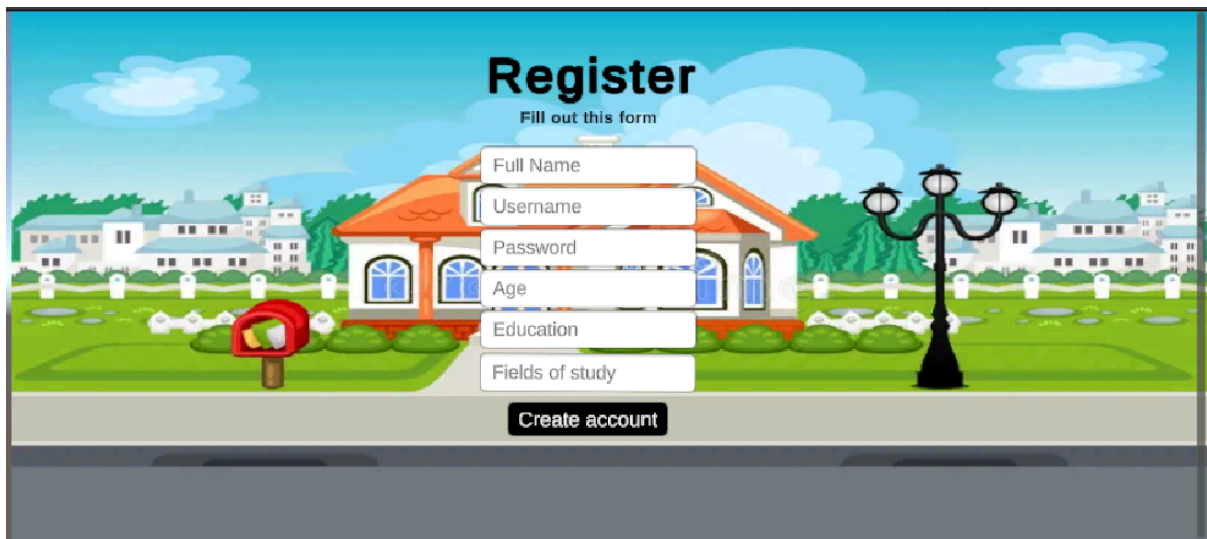
Login screen:

Upon installing the app, users are faced with the initial screen. Returning users can log in by entering their username and password. If they forget their password, they can reset it using the "Forgot Password" option. New users simply click "Register Now" and provide basic information to complete the registration process. With the login feature, any player can conveniently access the game using their saved information.



Registration screen:

When a player embarks on their first gaming session, they'll register by inputting personal information like their full name, username and password to establish an account. Furthermore, they'll provide details about their age, education, and specific areas of interest. This ensures that they are matched with puzzles that align with their preferences and engage their curiosity.



Register
Fill out this form

Full Name
Username
Password
Age
Education
Fields of study

Create account

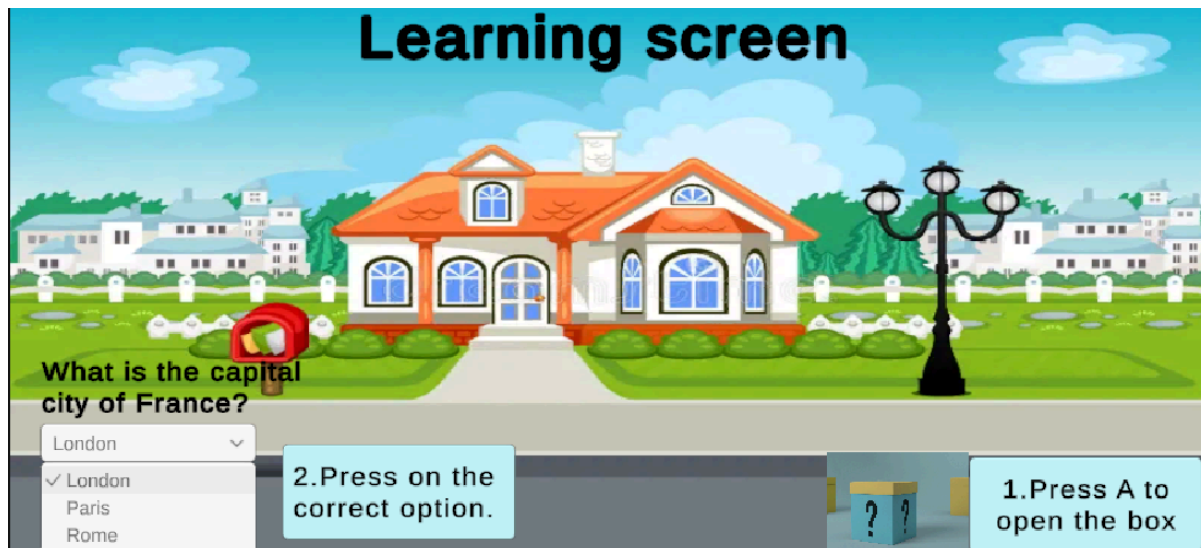
practice prompt:

Once the user inputs their name and age, they'll encounter a screen prompting them to choose whether they'd like to participate in a practice session. This option is particularly encouraged for novice users utilizing Unity.



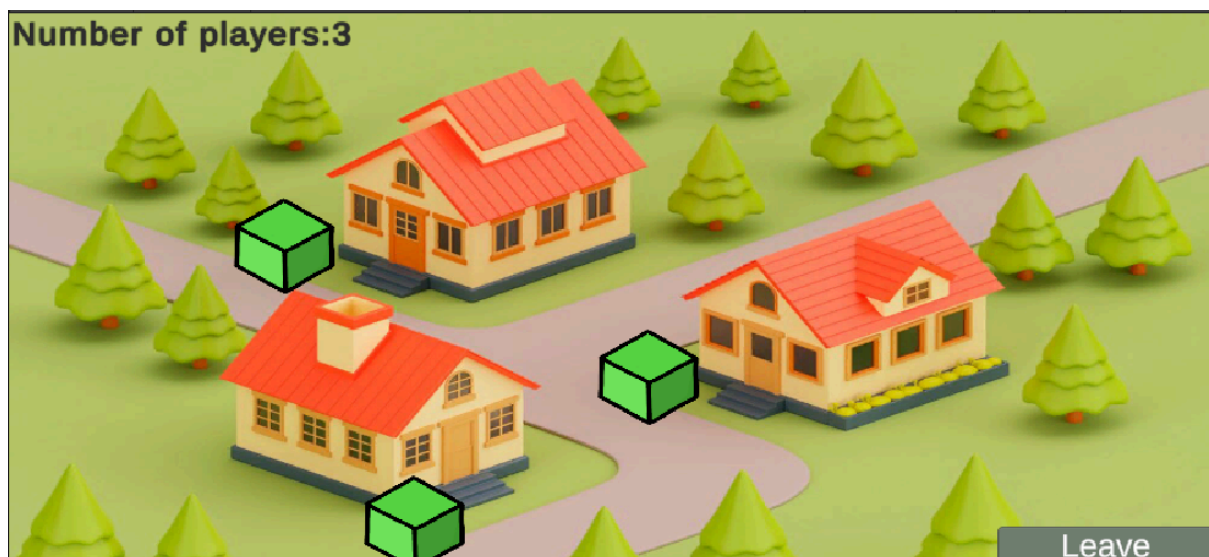
character walking towards the user:

During the practice session, users' input won't be stored in their accounts, distinguishing it from the simulation where all user actions are saved in our database. Users will encounter a figure that approaches and then retreats. They'll be prompted to halt the figure at a comfortable personal space distance. Additionally, they'll find boxes with instructions to open one and solve the puzzle inside.

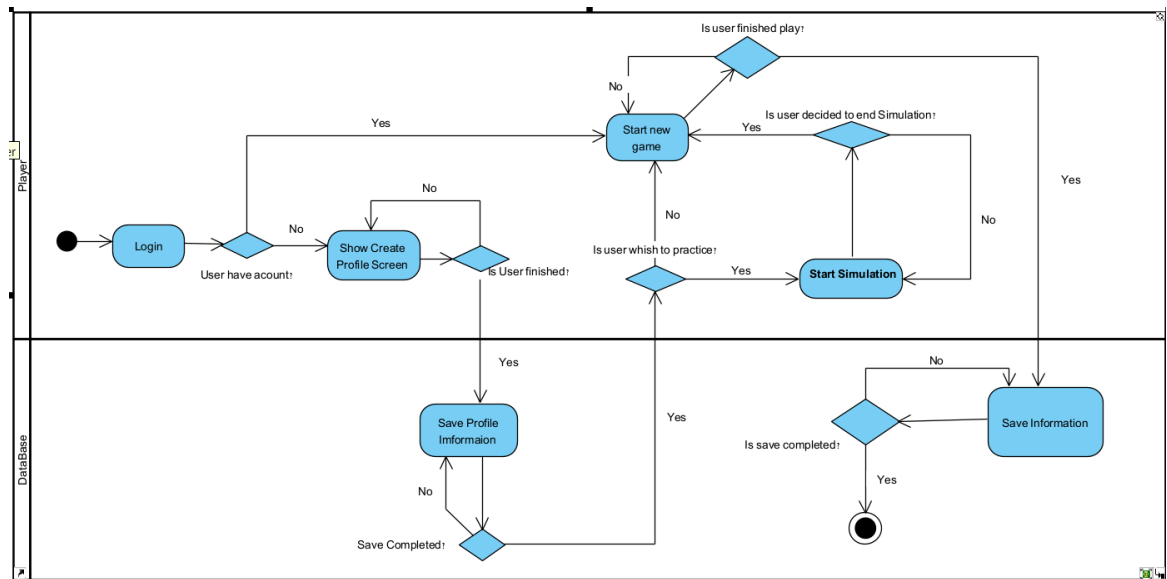


Start Play Screen:

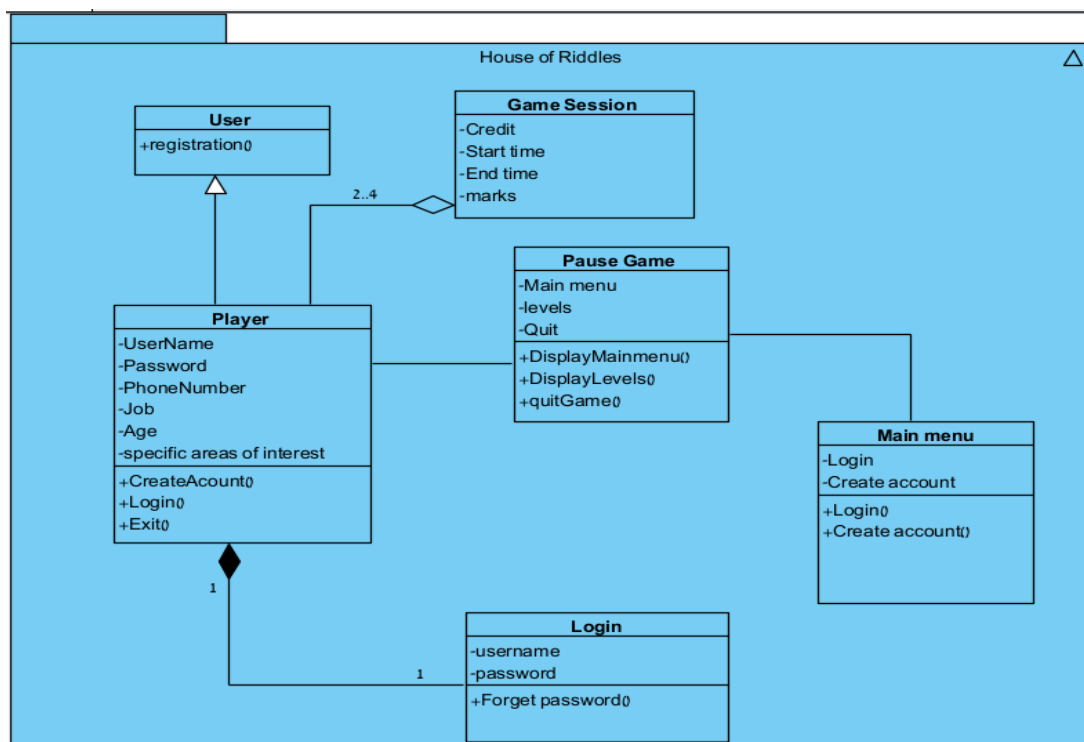
As players commence their gameplay, their progress and data are securely stored in the database. They find themselves in a neighborhood where each house corresponds to a player and features a unique lock and prize. Players endeavor to unlock each house and solve the puzzles within. The first player to finish all puzzles wins the respective house. Each player is assigned puzzle difficulty levels based on the number of houses they've won, ensuring a balanced challenge for all participants.



5.3.2 Activity Diagram



5.3.3 Class Diagram



6. Verification and Evaluation

Verification and evaluation in game development ensure that the game functions as intended and meets quality standards. Verification involves testing the game's features, mechanics, and functionalities to identify and fix any bugs or errors. The evaluation assesses the overall player experience, including gameplay, graphics, audio, and user interface, to ensure the game meets design objectives and player expectations.

6.1 Evaluation

Evaluation in game development evaluates various aspects of the game to ensure it meets the design goals and player expectations. Through testing and gathering feedback from players and stakeholders, evaluation helps developers identify strengths and weaknesses, iterate on design choices, and ultimately create a more enjoyable and engaging game for the target audience.

- **User Testing:** Developers conduct playtesting sessions with target audiences to observe how players interact with the game, gather feedback on gameplay mechanics, identify usability issues, and assess overall enjoyment.
- **Critical Reviews:** Game journalists and reviewers evaluate the game's elements, such as gameplay, graphics, sound design, narrative, and innovation, providing insights and critiques to inform potential players and developers.
- **Metrics Analysis:** Developers analyze in-game metrics, such as player retention rates, session lengths, and player behavior patterns, to evaluate the game's performance and identify areas for improvement in engagement and monetization strategies.

6.2 Verification

Verification in game development focuses on ensuring that the game functions correctly and meets specified requirements. Here are three examples:

- **Quality Assurance Testing:** QA testers systematically test the game to identify and report bugs, glitches, and other issues that may affect gameplay, performance, or stability. This includes functional testing, compatibility testing, and regression testing to ensure that fixes do not introduce new problems.
- **Compliance Testing:** Verification also involves checking whether the game complies with industry standards, platform guidelines, and legal requirements. This includes testing for age ratings, accessibility features, and platform-specific regulations to ensure the game can be released and distributed successfully.
- **Performance Testing:** Developers conduct performance testing to assess how the game performs under various conditions, such as different hardware configurations, network environments, and player loads. This helps optimize the game's performance, including frame rates, loading times, and server responsiveness, to provide a smooth and enjoyable experience for players.
- **Unit Testing:** Verification involves conducting unit tests on individual components or units of the game's code to ensure that each unit functions correctly in isolation. Developers write test cases to validate the behavior of functions, classes, and modules, helping to identify and fix errors early in the development process and ensure the stability and reliability of the codebase.

- Here are a few unit tests we're going to use:

Test	Component	Test case description	Expected Result
1	Login	Correct Username & Correct Password and press sign in.	Go to next screen
2	Login	Incorrect Password/Incorrect Username and press 'sign in'.	Error message
3	Login	Press sign up	Go to Registration screen
4	Register	Incorrect field type and press 'create account' button.	Error message
5	Register	Correct field type and press 'create account'	Go to "practice prompt"
6	Player Movement	Test if player can move left when left arrow key pressed	Player moves left as expected
7	Player Movement	Test if player can move right when right arrow key pressed	Player moves right as expected
8	Take a puzzle	A player answer the question	Feedback message
9	Puzzle Interaction	Check if the puzzle opens when the player interacts with it	Puzzle opens as expected
10	Hint System	Test if hint system provides helpful hints to the player	Hints are relevant and aid in puzzle-solving
11	Level Progression	Test if completing one puzzle unlocks the next level	Next level unlocks upon puzzle completion
12	Adaptation to the player's behavior.	Test if puzzles change based on player actions	Puzzle elements or layout change based on player's progress

7. References

1. https://www.researchgate.net/profile/Jeffrey-Craighead/publication/265284198_Using_the_Unity_Game_Engine_to_Develop_SARGE_A_Case_Study/links/55d33fdf08aec1b0429f32f4/Using-the-Unity-Game-Engine-to-Develop-SARGE-A-Case-Study.pdf
2. https://www.theseus.fi/bitstream/handle/10024/496604/Breugelmans_Arno.pdf?sequence=2&isAllowed=y
3. <https://isprs-annals.copernicus.org/articles/IV-4-W4/161/2017/isprs-annals-IV-4-W4-161-2017.pdf>
4. https://books.google.co.il/books?hl=ar&lr=&id=OMNiDwAAQBAJ&oi=fnd&pg=PP1&dq=Reinforcement+Learning+Algorithms+with+unity+game&ots=CkBO_Ny-A92&sig=OHdCTANtyNth_N5-Auc_sdUXZq8&redir_esc=y#v=onepage&q=Reinforcement%20Learning%20Algorithms%20with%20unity%20game&f=false