**Refraction: An Adventure Game Exploring Fragmented Identity and Perception Through Memory Reconstruction.**

A Thesis Presented to the Course Specialists

of the College of Information and Communication Technology

of Taguig City University

In Partial fulfillment of the requirement for the Degree

of Bachelor of Science in Computer Science

By

Bacon, Jhon Lorenz G.

Dungao, Justine Zyrie

Enriquez, Timothy Jhon

Laure, Shane

June 2025

**CHAPTER I**

**INTRODUCTION**

**Introduction**

In recent years, narrative-driven indie games have gained increasing recognition for their ability to tell emotionally resonant stories through unique mechanics and atmospheric world-building. These games go beyond entertainment—they provide a space for players to explore complex themes such as grief, memory, identity, and perception. Titles like Mad Father, Ib, and To the Moon have captivated audiences not through high-end graphics or intense action, but through deeply personal narratives and immersive experiences.

Refraction, the proposed project, positions itself within this emerging genre of reflective, story-centric adventure games. It draws inspiration from games that invite the player to interact not just with the world, but with the story itself—making choices, uncovering hidden fragments, and shaping the outcome. As digital storytelling continues to evolve, this project seeks to contribute to the landscape by offering a game that blends mystery, exploration, and introspection into a single, compelling experience.

**Project Context**

The development of Refraction stems from a desire to create a narrative experience that explores how fragmented memories and distorted perceptions can influence a person's identity. In this game, players are placed in the shoes of a protagonist with no recollection of who they are or what happened to the world around them. Set in a strange and hauntingly beautiful environment, the player is guided by a mysterious companion as they collect memory fragments and piece together their past.

Unlike conventional linear storytelling, Refraction uses non-linear progression, player-driven choices, and symbolic visuals to let the narrative unfold organically. The surreal, dreamlike setting reflects the protagonist’s internal struggle and the uncertainty of memory. By incorporating mechanics such as memory reconstruction and branching paths influenced by player decisions, the game offers a personalized narrative journey that mirrors the complexity of human introspection.

This project is not just about gameplay—it's about evoking a sense of emotional resonance. Through exploration, decision-making, and interactive storytelling, Refraction aims to provide an experience that speaks to the very core of what it means to remember, to question reality, and to seek truth within oneself.

**Purpose and Description**

**Purpose of the Study**

The purpose of this study is to design and develop Refraction, a story-driven adventure game that explores themes of identity, memory, and perception. This project aims to examine how interactive storytelling and non-linear gameplay can be used to evoke emotional engagement and narrative immersion. By integrating decision-based mechanics, memory reconstruction, and symbolic visuals, Refraction intends to offer players a meaningful experience that reflects the complexity of self-discovery and personal truth.

Additionally, the study seeks to contribute to the growing field of narrative-based game development by demonstrating how game design can be used as a medium for introspective storytelling. It aims to bridge the gap between technical implementation and emotional depth, offering insights into both creative and functional aspects of building immersive, story-centric games.

**Description of the Study**

This study focuses on the development of Refraction, a 2D top-down adventure game built using the Godot Engine. The game follows a protagonist who awakens in a surreal, fragmented world with no memory of their identity. Guided by a mysterious companion, the player must explore different environments, collect scattered memory fragments, and slowly piece together the truth about their past.

The gameplay combines puzzle-solving, exploration, and dialogue-based choices that affect the narrative’s progression. A core feature of the game is its use of a dynamic narrative system—initially implemented through a Finite State Machine (FSM) but open to exploration of more adaptive storytelling algorithms. Player decisions and discovered memories shape the unfolding story, offering multiple narrative outcomes.

The study involves conceptualizing the narrative structure, designing the gameplay mechanics, developing the visual and sound elements, and implementing technical systems that support player progression and data saving. It also includes testing and refining the game based on user feedback to ensure emotional impact, gameplay clarity, and story cohesion.

**Objectives of the Study**

**Main Objective**

To develop an emotionally driven, story-rich adventure game titled Refraction that explores themes of fragmented identity and perception through memory-based gameplay, while showcasing the effective use of interactive storytelling and adaptive game design techniques.

**Specific Objectives**

Specifically, the researchers outlined the following objectives to guide the development of the project and ensure that the study’s purpose is met. These goals provided direction and focus, ensuring that each key aspect of the game—technical and narrative—was thoughtfully addressed.

The specific objectives are as follows:

1. To implement adaptive storytelling mechanics that personalize the player’s experience through a dynamic narrative system, using an advanced decision-based algorithm such as Behavior Trees or Dialogue Graphs.
2. To design and develop a highly immersive game environment that visually and aurally supports the themes of memory, identity, and emotional fragmentation.
3. To provide a user-friendly and intuitive interface that enables seamless player interaction, exploration, and decision-making throughout the game.
4. To integrate a reliable data management system for tracking player choices, progress, and narrative branches, ensuring replayability and data persistence using tools like SQLite or Firebase.
5. To ensure game quality through ISO 25010 evaluation standards, focusing on the following attributes:

5.1 Functionality

5.2 Usability

5.3 Efficiency

5.4 Security (for save data integrity)

5.5 Compatibility (across target platforms)

5.6 Maintainability

5.7 Portability

5.8 Reliability

Feedback will be gathered through surveys, user testing, and documented suggestions to identify areas for refinement and guide future development efforts.

**Conceptual Paradigm**

The scope of this study focused on the development of Refraction, an adventure game exploring fragmented identity and memory reconstruction. The game involves creating a dynamic narrative where player choices influence the outcome, using the Godot Engine and GDScript programming language for development. The gameplay includes interactive decision-making, puzzle-solving, and memory fragment collection. Additionally, the game features a user-friendly interface designed to enhance player experience, ensuring smooth navigation throughout the immersive environment. The study evaluates the game’s performance, including its functionality, usability, efficiency, and security.

However, the study has some limitations. The game is developed for PC platforms only, with no immediate plans for mobile or console versions. The narrative depth is also limited by the available resources, focusing on a core set of choices rather than an expansive branching storyline. While the game offers a meaningful, interactive experience, its overall length is limited to approximately 10 hours of gameplay. The game’s technical performance is restricted by the capabilities of the Godot Engine and targeted hardware, which may impact accessibility on lower-end devices. Additionally, the game is designed solely as a single-player experience, with no multiplayer features planned. The testing phase also involved a limited user group, and the results may not fully capture a diverse range of user experiences.

**Figure 1. Conceptual Paradigm**

**Significance of the Study**

The significance of developing Refraction, an adventure game exploring fragmented identity and memory reconstruction, lies in its potential to provide an engaging and thought-provoking experience. The game focuses on themes of memory, identity, and perception, offering a narrative-driven journey that challenges players to navigate a surreal, dystopian world.

This study’s importance is rooted in its contribution to the gaming industry, specifically in the genre of adventure games with psychological themes. By blending interactive storytelling with environmental puzzles, Refraction offers a unique approach to exploring memory reconstruction and the boundary between reality and perception. The game’s ability to address complex topics, such as trauma and self-discovery, provides an opportunity for players to engage in meaningful reflection, encouraging emotional growth and empathy.

The deployment of Refraction could benefit multiple groups.

**End users**, specifically players, will have access to a game that promotes critical thinking, emotional engagement, and introspection. The game offers an interactive platform where users can explore the depths of the protagonist's psyche while navigating the consequences of their choices.

Additionally, **future researchers** and game developers can use this study as a reference for developing similar narrative-driven experiences, particularly those exploring mental health, memory, and identity. The findings of this study can offer valuable insights into creating games that challenge perceptions, push boundaries, and provide innovative approaches to interactive storytelling.

**Operational Definition of Terms/ Variables**

**Adventure Game**

An interactive, story-driven game where players explore environments, solve puzzles, and make decisions that influence the narrative. Refraction falls under this genre, featuring memory reconstruction and identity exploration as central themes.

**Memory Reconstruction**

The process of collecting fragmented memories in the game. Players interact with various environmental puzzles and dialogues to retrieve pieces of the protagonist's past, gradually piecing together their identity and understanding of the world.

**Fragmented Identity**

The protagonist’s disjointed sense of self, which is explored throughout the game. The protagonist must piece together their memories to uncover their true identity, reflecting the theme of psychological and emotional recovery.

**Perception**

The way the protagonist perceives their world and themselves, influenced by their memories and environment. Perception plays a significant role in Refraction, as the protagonist’s view of reality becomes distorted over time due to the psychological trauma they experience.

**Finite State Machine (FSM)**

A computational model used for managing the game’s decision-making system. In Refraction, FSM handles player choices and branching storylines, ensuring the game’s narrative flows logically without redundant paths. It supports the branching mechanics for different story outcomes based on decisions made by the player.

**Puzzle Mechanics**

Interactive elements that require the player to solve challenges within the game world. In Refraction, these puzzles are tied to the collection of memory fragments and unlocking further layers of the story. They contribute to the exploration of the protagonist's mind.

**Player Choice**

Decisions made by the player that directly impact the story and game world. In Refraction, the player's choices determine the unfolding of the plot, leading to multiple possible endings. These choices reflect the themes of identity, memory, and perception.

**Narrative-Driven**

A game design approach where the primary focus is on delivering an engaging story. Refraction uses this approach, with the protagonist's journey and psychological growth serving as the backbone of the gameplay.

**Emotional Engagement**

The game’s ability to evoke emotional responses from the player through its story, characters, and settings. Refraction aims to connect with players on an emotional level, particularly regarding themes of trauma, memory, and self-discovery.

**Surreal Dystopian Setting**

The fictional world in which the game takes place, marked by an otherworldly, dream-like atmosphere. The setting in Refraction represents the protagonist’s fragmented mind, offering an immersive environment that is both familiar and alien.

**Identity Recovery**

The protagonist’s journey to rediscover their true self by overcoming confusion and uncovering repressed memories. This process drives the narrative of Refraction, where the player aids in piecing together the protagonist’s identity through exploration and puzzle-solving.

**Multiple Endings**

A game feature where different choices lead to different conclusions. In Refraction, the player’s decisions throughout the game lead to various endings, with one secret ending unlocked by specific actions.

**Dynamic Exploration**

A gameplay element where the environment and interactions change based on player input. In Refraction, dynamic exploration allows the player to explore a surreal world that reflects the protagonist’s changing psychological state.

**Environmental Puzzles**

Challenges that involve interacting with the game’s world to uncover hidden elements or solve problems. These puzzles play a crucial role in progressing through the game and are essential for uncovering memory fragments.

**CHAPTER II**

**REVIEW OF RELATED LITERATURE**

**CHAPTER III**

**RESEARCH METHODOLOGY**

This section describes the research method, participants, population and sample, research instruments, data gathering procedures, and statistical analysis of the data. This is used to gather detailed information about user satisfaction, gameplay experience, and overall usability.

**Research Method Used**

The researchers used a developmental research approach focused on the design and development of Refraction, a story-driven adventure game that explores themes of memory and identity. The game was built using the Godot Engine and integrates multiple algorithms to enhance gameplay and narrative flow—specifically, the Finite State Machine (FSM) for managing character behavior, a Narrative Progression Algorithm to handle branching storylines, and a Tree-based Algorithm for structured decision-making and dialogue paths.

**Participants**

The participants in this study were a group of individuals who have experience playing adventure games. They were selected based on their familiarity with video games and their ability to provide feedback on gameplay, story, and user experience. The participants included both game testers and regular players who helped evaluate the game’s mechanics, narrative progression, and overall engagement.

**Population**

**Sample**

**Research Instruments**

The main research instruments used in this study were surveys, user feedback, and gameplay data. These tools were used to gather information on user experience, narrative engagement, and game functionality. The surveys included questions about gameplay satisfaction, story immersion, and usability of the game interface. Additionally, gameplay data (such as completion rates and choice outcomes) was collected to analyze the effectiveness of the game's design and decision-making mechanics.

**Data Analysis and Procedure**

The data collected from surveys and gameplay was analyzed using both qualitative and quantitative methods. Qualitative data from open-ended survey responses were analyzed thematically to identify common user sentiments and suggestions for improvement. Quantitative data, including gameplay statistics, were analyzed to evaluate player progress and the impact of different choices within the game. The results were used to assess the game's success in achieving its objectives and to identify areas for further refinement.

**Percentage**

In order to make comparison of the responses possible, this statistical tool was employed since it reflects which items were favored most or favored least.

The formula is:

**% = f/n x 100**

Where:

% = percentage

f = frequency

n = total number of responses

**Weighted Mean**

The Arithmetic Mean is the summation of the scores divided by the number of respondents. A weight Arithmetic Mean is used which variables involved in this study, disaster, preparedness, response capability, quality of training, etc., are considered abstract and continuous and cannot be counted individually.

It is obtained by applying the formula:

Xw = ∑fw/ ∑f

Where:

Xw = sum of all the products of f and w is the frequency of each weight and w is the weight as

4,3,2,1.

∑f = sum of all the responses of the sample size

**Hardware Requirements**

Table 2. Hardware Requirements

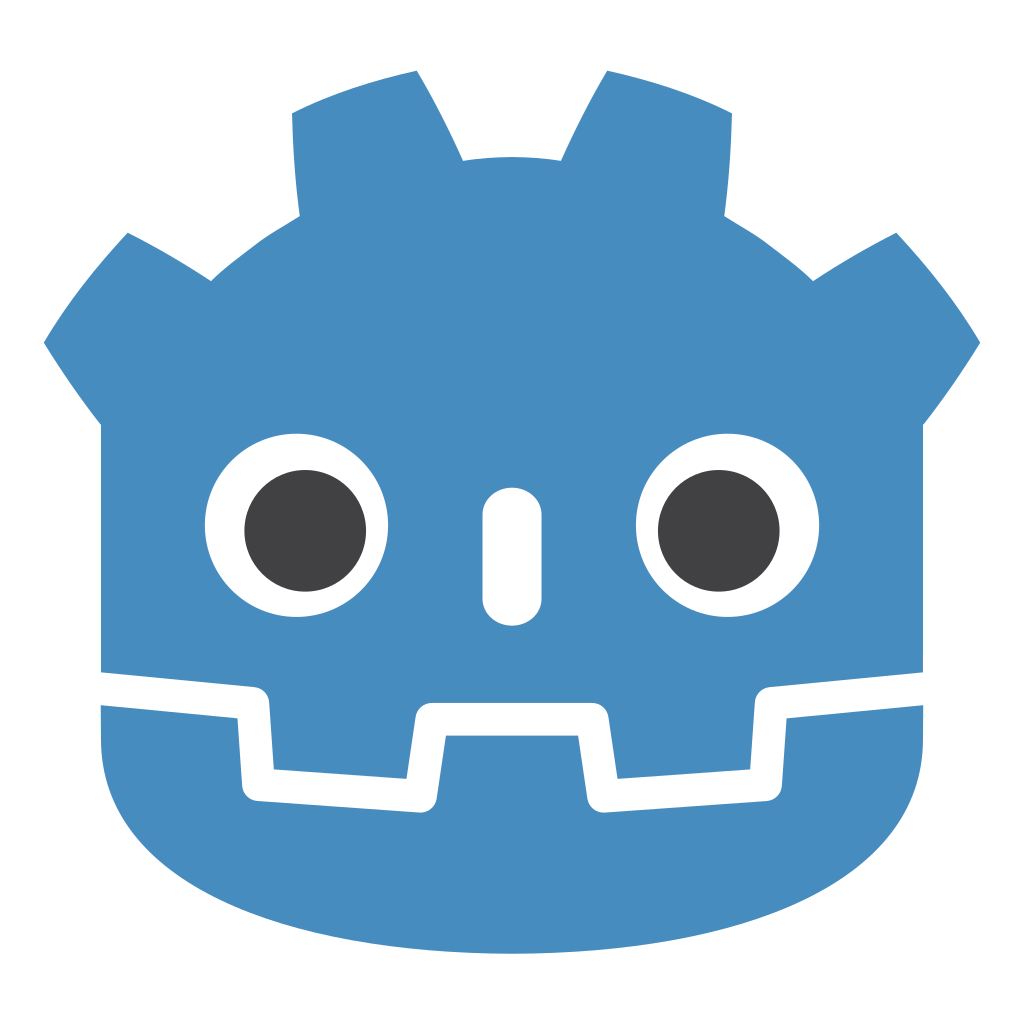
|  |  |  |
| --- | --- | --- |
| **Component** | **Minimum Requirement** | **Recommended Specification** |
| **Operating System** | **Windows 10** | **Windows 10** |
| **Processor** | **Intel I5-5500** | **Intel I5-7500** |
| **Graphics** | **AMD RX 550** | **AMD RX580** |
| **Memory** | **4GB RAM** | **4GB RAM or higher** |
| **Storage** | **5 GB available space** | **5 GB available space** |

Table 2 outlines the hardware requirements necessary for running the proposed system effectively. The minimum requirements include a Windows 10 operating system, an Intel i5-5500 processor, an AMD RX 550 graphics card, 4GB of RAM, and at least 5GB of available storage. These specifications are sufficient for the basic operation of the application, including messaging, file sharing, and encrypted communication.

On the other hand, the recommended specifications suggest using a higher-tier Intel i5-7500 processor and an AMD RX 580 graphics card, along with increased memory, to ensure optimal performance—especially for more resource-intensive features like real-time audio/video calls and dynamic animations. Meeting or exceeding the recommended setup enhances system stability, ensures smoother user interactions, and delivers an overall better experience.

**Software Requirements**

The following figures represent the software requirements used in the development of the proposed offline game, which features a combination of 2D and 3D elements. These components provided the necessary tools, environments, and systems to implement key functionalities such as rendering, logic processing, state management, and data persistence.



**Figure 2. Godot Logo**

Godot is the primary game engine used in the development of the offline game. It offers built-in support for both 2D and 3D game environments, making it ideal for the hybrid visual style of the project. The engine's scene system and animation tools facilitated the development of dynamic interactions and immersive gameplay. Godot also supports GDScript and C#, allowing for a flexible and efficient programming workflow.



**Figure 3. GDScript**

GDScript is a high-level scripting language built specifically for the Godot Engine. It features a Python-like syntax, making it beginner-friendly and efficient for rapid development. In this project, GDScript is used to handle core gameplay mechanics, character interactions, dialogue sequences, and environmental events. Its tight integration with the Godot engine allows for seamless real-time updates and streamlined control over the game’s narrative flow, animations, and logic, making it ideal for creating immersive and responsive adventure game experiences.



**Figure 4. C#**

C# is a modern, object-oriented programming language developed by Microsoft that is widely used for building robust and scalable applications. In this project, C# is utilized within the Godot engine to implement advanced gameplay mechanics, system functions, and UI responsiveness. Its strong typing, efficient memory management, and rich library support make it suitable for managing complex logic such as player interactions, memory reconstruction sequences, and secure data handling. C# enhances the overall performance and maintainability of the application, allowing for faster development and more reliable execution across different platforms.



**Figure 5. SQLite Logo**

SQLite was integrated into the game to handle local data storage, including player progress and in-game choices. As an embedded database, it offers fast performance and reliability without requiring an internet connection, aligning with the offline nature of the game. The simplicity and portability of SQLite made it an ideal solution for persistent game state management across different play sessions.

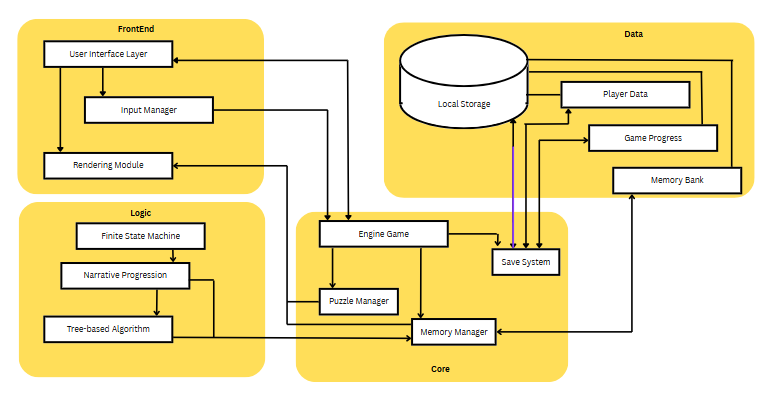
**Project Design**

The design of the "Refraction" adventure game is centered around creating an engaging and immersive experience that combines narrative-driven exploration with puzzle-solving mechanics. The game will utilize a well-defined structure, with an emphasis on dynamic environments and branching storylines that are shaped by player decisions. The game world will be designed as a surreal, dystopian city, featuring overgrown vegetation and mysterious creatures that players can explore as they piece together their fragmented memories.

The user interface (UI) will be intuitive and minimalistic, allowing players to easily navigate through the game and interact with the various elements in the environment. Key features like the dialogue choices and puzzle interactions will be accessible through a streamlined menu that is designed to enhance the player experience without overwhelming them. The gameplay will focus heavily on the player's choices and their ability to solve environmental puzzles, each leading to new discoveries that reveal more about the protagonist's past and the world around them.

The game will be driven by a combination of algorithms, including a finite state machine (FSM), a narrative progression algorithm, and a tree-based algorithm. These algorithms will govern how the narrative progresses based on the player's choices, creating a dynamic and engaging experience that adapts to the decisions made throughout the game. This approach allows for multiple paths, outcomes, and endings, ensuring that each playthrough offers a unique experience. Through thoughtful design and a focus on narrative depth, "Refraction" will offer players an unforgettable journey of self-discovery and survival in a world where the boundaries between dream and reality blur.

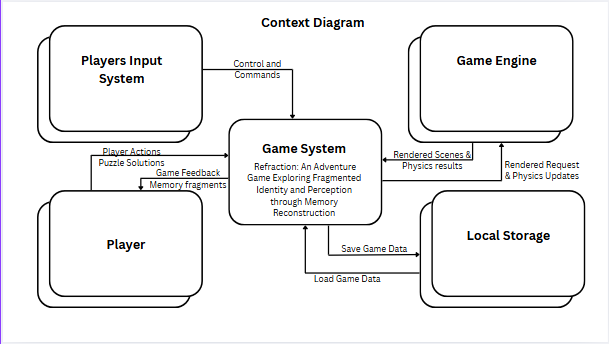
**System Architecture**

****

**Figure 6.** System Architecture

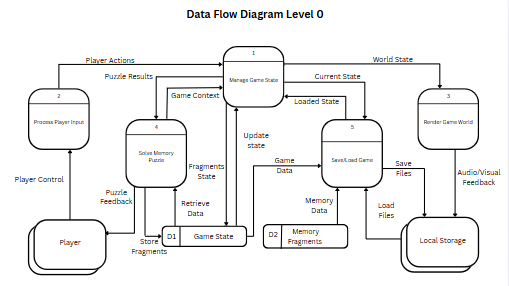
Figure 6 depicts the System Architecture of the Refraction: An Adventure Game Exploring Fragmented Identity and Perception Through Memory Reconstruction.. The architecture consists of four interconnected layers designed to provide an immersive identity exploration experience. The Frontend Layer handles all player interactions through a User Interface component that manages visual elements, a Rendering Module that generates the game environment, and an Input Manager that processes player commands. The Core Layer functions as the central processing unit, where the Game Engine coordinates all game mechanics while the Puzzle Manager generates memory-based challenges, the Memory Manager tracks recovered fragments, and the Save System preserves player progress locally. All persistent data is maintained in the Data Layer, which stores the Memory Bank containing all discovered memory fragments, Player Data for profile information, and Game Progress tracking completion status. The Logic Layer implements the game's algorithmic foundation through a Finite State Machine driving the Narrative Logic for consistent storytelling progression, a Tree-based Algorithm powering the Perception Logic to create branching perceptual changes based on recovered memories, and a dynamic Narrative Progression system within the Adaptation Logic that adjusts the experience based on player performance. This integrated architecture ensures that as players solve puzzles and recover memory fragments, the game world transforms in response, creating a personalized journey of fragmented identity reconstruction with each gameplay session.

**Data Flow Diagram**



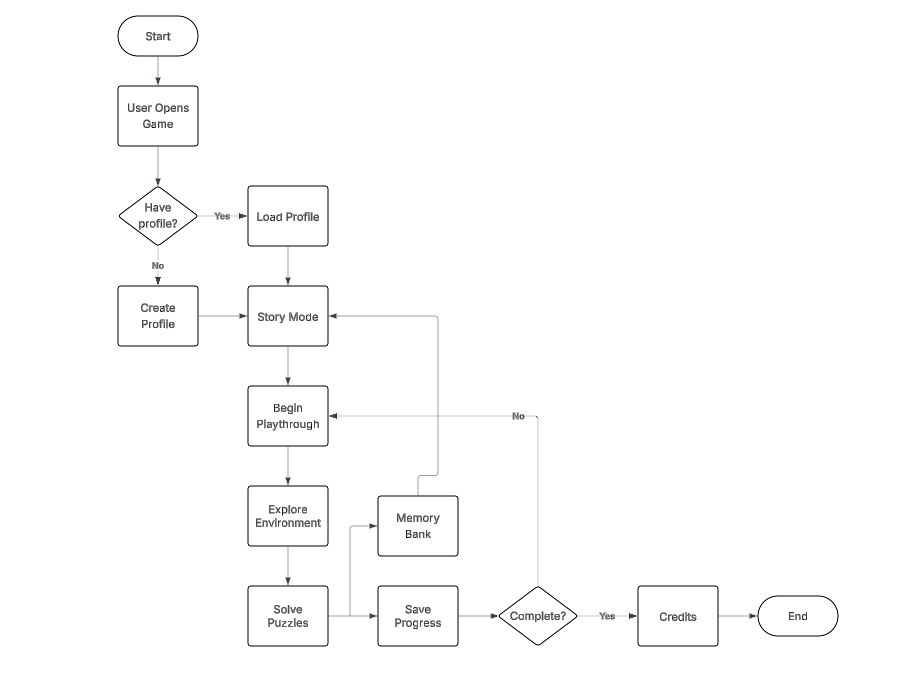
**Figure 7. Data Flow Diagram Level 0**

The Figure 7 illustrates a Data Flow Diagram (DFD) Level 0, also known as a Context Diagram, for the Refraction game system. At the center is the “Game System” labeled as “Refraction: An Adventure Game Exploring Fragmented Identity and Perception through Memory Reconstruction.” This central process interacts with four external entities: Players Input System, Player, Game Engine, and Local Storage. The Players Input System sends “Control and Commands” to the Game System. The Player exchanges multiple data flows Feedback” and “Memory fragments.” The Game Engine provides “Rendered Scenes & Physics results” to the Game System while receiving “Rendered Request & Physics Updates.” Local Storage maintains game persistence through bidirectional data flows, with the Game System “Save Game Data” flowing to storage and “Load Game Data” flowing back to the system when needed. This context diagram effectively captures the high-level data exchanges between the Refraction game system and its external entities, showing how player interactions, rendering processes, and data persistence are managed within the game’s architecture.



**Figure 8. Data Flow Diagram Level 1**

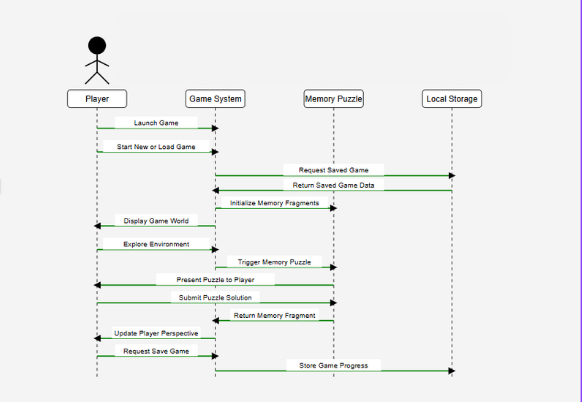
The figure 8 illustrates a Level 1 Data Flow Diagram (DFD) for a memory-based puzzle game system, detailing the internal processes and their interactions with external entities and data stores. At the core is the "Manage Game State" process, which coordinates game progression by integrating inputs from various modules. The "Player" provides control inputs that are processed by "Process Player Input" and used in the "Solve Memory Puzzle" module to update the game context. Puzzle solutions and memory fragments are retrieved from two data stores—Game State and Memory Fragments—supporting dynamic gameplay. The "Render Game World" process uses the current game state to generate audiovisual feedback for the player, while the "Save/Load Game" module interacts with Local Storage to persist or restore game progress. This diagram effectively outlines how player input, memory puzzles, game state management, rendering, and data persistence work together within the system’s architecture.

**FlowChart **

**Figure 9. Proposed FlowChart**

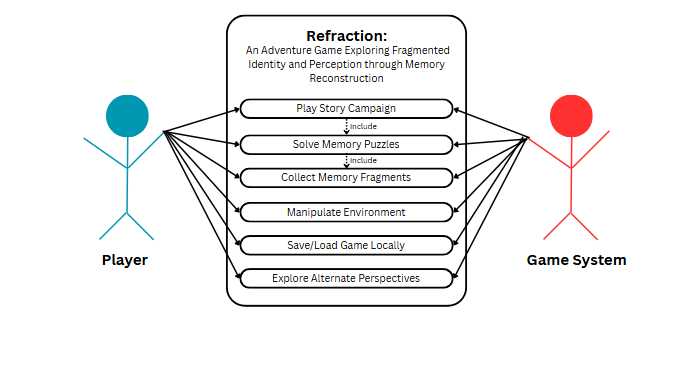
**The figure illustrates a flowchart representing the gameplay logic of the Refraction game system**, outlining the sequence of activities a player follows from launching the game to completing it. The flow begins when the user opens the game, prompting a decision on whether an existing player profile is available. If a profile exists, it is loaded; otherwise, a new profile is created. In both cases, the player proceeds into Story Mode and begins the playthrough. During the game, the player explores the environment and engages in solving puzzles. As gameplay unfolds, players interact with the Memory Bank and are given the option to save their progress. The system then checks if the game is complete. If the game is not yet completed, the loop returns the player to Story Mode for continued play. If the game is complete, the system transitions to displaying the game credits, concluding the experience. This flowchart effectively captures the player's journey and decision-making process throughout the game, highlighting how exploration, puzzle-solving, memory interaction, and progress tracking are integrated within the Refraction game structure.

**Unified Model Language**

****

**Figure 10. Sequence Diagram**

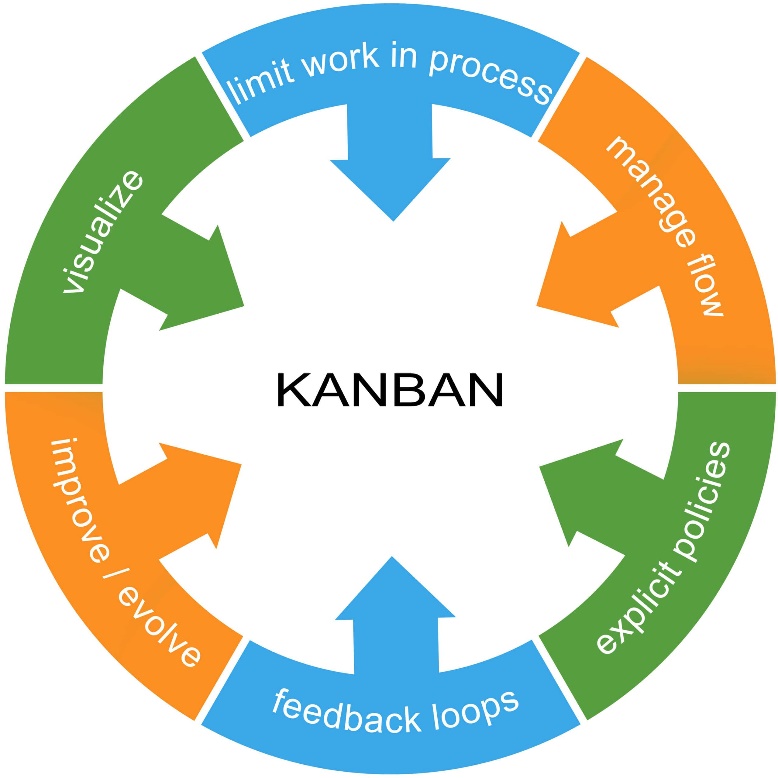
**The figure 10 illustrates a UML Sequence Diagram for the Refraction game system**, detailing the dynamic interactions between the Player, Game System, Memory Puzzle module, and Local Storage during gameplay. The interaction begins when the Player initiates the game by launching it and choosing to start a new game or load a saved one. In response, the Game System sends a request to Local Storage to retrieve saved game data, which is returned and used to resume the session. The Game System then displays the game world, allowing the Player to explore the environment. As exploration progresses, the system initializes memory fragments and triggers a Memory Puzzle, which is presented to the Player. Upon submitting a solution, the Memory Puzzle module returns the corresponding memory fragment to the Game System. The system then updates the Player’s in-game perspective based on this input. Finally, when the Player requests to save the game, the Game System sends the updated state to Local Storage for persistence. This sequence diagram effectively captures the real-time interaction and flow of data between core components, highlighting how memory reconstruction, player interaction, and data management are orchestrated within the game architecture.

****

**Figure 11. Use Case**

The Figure 11 illustrates the use case diagram for Refraction, depicting the interaction between two primary actors: the Player and the Game System. At the center is the game "Refraction: An Adventure Game Exploring Fragmented Identity and Perception through Memory Reconstruction." The diagram shows six key use cases that represent the core gameplay interactions: "Play Story Campaign," "Solve Memory Puzzles," "Collect Memory Fragments," "Manipulate Environment," "Save/Load Game Locally," and "Explore Alternate Perspectives." Each use case is connected to both the Player and Game System with directional arrows, indicating that these functions involve both actors. The "Play Story Campaign" and "Solve Memory Puzzles" use cases include an "include" relationship, suggesting these are essential components of the gameplay experience that incorporate other functionalities. This diagram effectively captures how players interact with the game system to reconstruct memories, solve puzzles, and explore different perspectives, all central to the game's theme of fragmented identity and perception.

**System Development**



**Figure 12. Agile Kanban Development Life Cycle**

In the development of the "Refraction" adventure game, the Agile Kanban Software Development Life Cycle (SDLC) model will be utilized to ensure a structured yet flexible approach to the project’s execution. Kanban, a visual workflow management method, is characterized by its emphasis on continuous delivery and improvement. The development process will be organized using a Kanban board, where tasks are visually represented as cards in different columns such as “To Do,” “In Progress,” and “Done.” This system allows for seamless task prioritization, real-time progress tracking, and transparency in development, enabling the team to manage resources efficiently and respond promptly to any challenges.

The pull-based nature of Kanban means that tasks will only move to the next stage once the previous one is completed, ensuring no bottlenecks in the workflow and maintaining a steady pace of progress. By incorporating this methodology, the team will be able to focus on iterative development, delivering incremental improvements that align with user feedback and project needs, thus fostering adaptability, reducing risks, and enhancing collaboration throughout the project.

**Planning**

In this phase, the team defines the project goals, scope, and high-level requirements. Agile planning is iterative, meaning that it is adjusted regularly throughout the development process based on feedback and progress. The main focus during this phase is on identifying core features and user stories, with an initial roadmap established to guide the project. Tasks are organized on a Kanban board, giving the team a visual overview of the work that needs to be done.

**Analysis**

The analysis phase involves a deeper dive into the game’s requirements. This includes a detailed study of the gameplay mechanics, story, character design, and technical specifications. The team reviews the current needs of the project, identifies potential risks, and clarifies any uncertainties. Feedback from initial playtesting and stakeholder meetings is gathered, and user stories are refined to ensure that the project’s scope is achievable and aligned with user expectations.

**Design**

Once the project’s requirements and scope are defined, the design phase begins. In this phase, the overall architecture of the game is developed, including the user interface (UI) design, graphics, character design, and sound. The technical design also takes place, including the choice of game engine (Godot in this case), the selection of algorithms (FSM, Narrative Progression, and Tree-based), and data storage solutions (Firebase or SQLite). The design phase in Agile Kanban is highly collaborative and allows for feedback loops that ensure the design aligns with the evolving vision of the project.

**Implementation**

During implementation, the team begins actual development, coding, and integrating the game’s features. Agile allows for flexibility here, where the team can prioritize tasks based on immediate needs and user feedback. The Kanban board allows for tracking progress on each task, from coding individual game mechanics to implementing the encryption algorithms for secure communication within the game. This phase is focused on delivering small, functional parts of the game that can be tested and improved over time.

**Testing & Integration**

In this phase, the team focuses on quality assurance (QA) and integration. Testing is carried out iteratively throughout development, ensuring that the implemented features work as expected and meet the project’s requirements. The game’s various features, including its algorithms, are thoroughly tested for functionality, performance, and security. Integration testing is also done to ensure that different components of the game (UI, algorithms, and gameplay mechanics) work together seamlessly. Any issues or bugs identified are addressed through feedback loops within the Kanban system.

**Maintenance**

Once the game has been released, the maintenance phase begins. This phase focuses on bug fixes, updates, and responding to player feedback. Agile Kanban allows for continuous iteration during this phase, where the team can release incremental updates and fixes. The Kanban board remains in use to track ongoing work, and the team monitors the game’s performance, making improvements as necessary based on user feedback and emerging requirements. This ensures that the game evolves over time, maintaining its quality and addressing any post-launch issues.

**ALGORITHM DISCUSSION**

**Comparison of Algorithm**

In the proposed game "Refraction," the narrative progression and decision-making system rely on specific algorithms designed to handle branching paths and player choices. The researchers decided to compare and evaluate different algorithms, focusing on the Finite State Machine (FSM), Narrative Progression Algorithm, and Tree-based Algorithm. This comparison aims to assess their effectiveness in driving the game's story, managing player decisions, and ensuring dynamic gameplay.

By analyzing factors such as state transitions, branching logic, and the complexity of narrative paths, the researchers aim to determine the most suitable algorithm for structuring the game's progression. This comparison also highlights the trade-offs between the different approaches, balancing the depth of narrative exploration with system performance and player engagement.

The following table outlines the strengths and weaknesses of the three key algorithms considered for the game:

**Finite State Machine (FSM)**

| **Strengths** | **Weaknesses** |
| --- | --- |
| The **Finite State Machine (FSM)** is simple, deterministic, and easy to implement. It works well in systems where the state transitions are clearly defined, making it ideal for situations that require well-structured, predictable outcomes. FSM allows for easy debugging, as it is easy to trace the state of the system at any given point. Additionally, it is relatively lightweight in terms of processing, ensuring a smooth experience even in resource-constrained environments. FSM is highly suitable for games or systems where the logic is relatively straightforward, such as a series of fixed outcomes based on user actions. | However, FSM becomes limited when handling complex branching narratives or non-linear progression. It can struggle with more dynamic or emergent storylines, as the number of states and transitions required can quickly increase, leading to a cumbersome and hard-to-manage state machine. It is also less adaptable for systems that need more flexibility or variation, as each new scenario requires adding more states and transitions, potentially making the logic more difficult to maintain. |

**Narrative Progression Algorithm**

| **Strengths** | **Weaknesses** |
| --- | --- |
| The **Narrative Progression Algorithm** provides a high degree of flexibility, making it well-suited for games or applications with complex, branching storylines. It allows for dynamic progression where the narrative can unfold differently based on player choices, and it is capable of supporting a wide variety of story outcomes and character interactions. The algorithm can track multiple threads of the narrative simultaneously, offering a more immersive and varied storytelling experience. This flexibility makes it a great choice for games or interactive experiences that prioritize the player’s agency in the narrative. | On the downside, the **Narrative Progression Algorithm** can be resource-intensive, especially in large-scale projects with multiple branching paths and complex narratives. As more branches are added, the system may require significant computational power to track and process the various narrative states. The algorithm’s complexity can also make it harder to scale and maintain over time, requiring careful management to avoid performance bottlenecks or loss of coherence in the narrative. Additionally, designing and implementing a well-balanced narrative progression system may require more development time and a deeper understanding of storytelling techniques. |

**Tree-based Algorithm**

| **Strengths** | **Weaknesses** |
| --- | --- |
| The **Tree-based Algorithm** offers deep, dynamic branching paths, which is ideal for managing complex decision trees in games or applications. Each decision or action can lead to different outcomes, allowing for more personalized and immersive experiences. This structure supports a clear, visual hierarchy of choices and consequences, making it easier to design and visualize complex storylines. Tree-based algorithms also enable efficient backtracking and exploration of different narrative routes, making them useful for games that feature replayability or multiple endings based on user choices. | The primary weakness of the **Tree-based Algorithm** is that it can lead to performance issues when dealing with large decision trees. As the tree grows, particularly in large, open-world, or highly interactive games, the number of possible paths and outcomes increases exponentially, which can cause significant computational overhead. This can result in slower processing times, lag, and difficulty maintaining smooth performance. Additionally, managing and maintaining a vast tree structure can become complex, making it challenging to track all potential branches and ensure the narrative remains cohesive and well-integrated. |

**Features**

**Finite State Machine (FSM)**

* State-driven transitions: FSM handles the state changes based on the player’s actions and decisions. Each state represents a different narrative or gameplay condition.
* Clear structure: This algorithm helps manage the game's flow, ensuring it remains within predefined logical sequences, ideal for specific decisions or scenes with fixed outcomes.
* Predictable behavior: FSM allows for easy debugging and consistency, which is essential for maintaining immersion in the game's unfolding plot.

**Narrative Progression Algorithm**

* Dynamic story sequencing: This algorithm adapts the sequence of events based on player choices, making sure that the narrative evolves in real-time with each decision.
* Emotion-driven progression: It influences the pacing and emotional buildup of the story, reflecting the protagonist's mental state and growth as they uncover memories.
* Conditional branching: Different outcomes and story branches are triggered based on the player's actions, leading to a variety of potential conclusions.

**Tree-based Algorithm**

* Branching decisions: This algorithm creates a tree structure, where each choice leads to different story paths, and the player’s decisions affect later events.
* Scalable structure: It allows for easy expansion as more choices and events are added, which is crucial for larger, nonlinear games like Refraction.
* Memory integration: By organizing the narrative into branches, this algorithm helps manage memory fragments and their impact on story progression.

**Functions**

**Finite State Machine (FSM)**

* State management: It controls the different "states" of the game, such as exploration, dialogue, puzzles, and confrontations. This ensures smooth transitions between these elements.
* Game flow control: FSM ensures that the game doesn’t stray from the necessary narrative or gameplay paths, maintaining a logical progression.
* Simplifies complex decisions: It helps structure decisions that lead to predetermined results, offering clarity and consistency.

**Narrative Progression Algorithm**

* Dynamic decision-making: The algorithm adapts the narrative flow in response to player choices, making the story evolve uniquely for each player.
* Psychological impact reflection: As the protagonist recovers fragmented memories, this algorithm adjusts the narrative to reflect their emotional or cognitive changes.
* Pacing control: The algorithm helps manage the speed at which the story unravels, slowing down or speeding up based on the emotional intensity of the plot.

**Tree-based Algorithm**

* Branching story progression: It handles the branching structure of the game, determining which paths the player can take based on prior choices, ensuring a responsive and dynamic narrative.
* Tracking player choices: The algorithm tracks player decisions and how they influence future events, creating a web of consequences that affect the game's end.
* Choice consequence management: It ensures that decisions made in earlier stages resonate throughout the game, creating meaningful impacts.

**Uses**

**Finite State Machine (FSM)**

* Narrative flow control: Used to ensure that story events occur in a linear or semi-linear fashion, providing structure to the overall narrative experience.
* Puzzle mechanics: FSM is often used to manage puzzle states, tracking whether they are completed or not and altering the environment accordingly.
* Decision tracking: For key choices with predefined outcomes, FSM ensures that decisions are resolved logically.

**Narrative Progression Algorithm**

* Memory fragment collection: This algorithm is used to guide the protagonist’s memory reconstruction journey, dictating the sequence in which memories are recovered and how they influence future choices.
* Character development: It tracks the emotional growth of the protagonist as they experience revelations, with decisions affecting their psychological state and actions.
* Creating unique playthroughs: By controlling the branching and pacing of the narrative, this algorithm guarantees that no two playthroughs are exactly alike, enhancing replayability.

**Tree-based Algorithm**

* Branching storylines: Used to create a multi-path narrative, where each choice leads to different outcomes. It ensures the game adapts to the player's decisions and leads them down different paths.
* Tracking player impact: The algorithm helps track how a player's choices ripple through the story, influencing not just immediate consequences but also long-term outcomes.
* Customizable endings: The tree structure is used to create multiple endings, depending on the different branches and the player’s cumulative choices.

**Application Comparison**

In this section, the researchers compared several existing adventure games that utilize similar narrative and decision-making systems, focusing on the algorithms used, story progression features, and platform compatibility. This comparison highlights how the selected approach for "Refraction" stacks up against other well-known games in the genre. The comparison is organized as follows:

| **Game** | **Narrative Structure** | **Algorithm Used** | **Key Features** | **Platform Compatibility** |
| --- | --- | --- | --- | --- |
| **Mad Father** | Linear with branching endings | Finite State Machine (FSM) | Puzzle-solving, horror, light choice consequences | PC |
| **Alice: Madness Returns** | Linear progression with psychological themes | FSM + Event-driven scripting | Combat, environmental puzzles, symbolic storytelling, memory fragments | PC, PlayStation, Xbox |
| **Refraction** | Branching with memory-based progression | Narrative Progression + FSM | Memory fragments, emotional choices, puzzle-exploration blend | PC |