**NUANSA INFORMATIKA**
Volume 19 Number 2 July 2025

***p-ISSN :1858-3911* , *e-ISSN : 2614-5405***
https://journal.fkom.uniku.ac.id/ilkom

# Fuzzy-Driven Adaptive NPC Behavior in a Meme-Based Platformer Game for Android Mobile

**Encep Sayid Amrulloh [1], Rio Andriyat Krisdiawan*[2], Iwan Lesmana[3], Luti Rohmawati[4]**
[1,2,3] Department of Computer Science, Faculty of Computer Science, Kuningan University
[4] Institute Pangeran Dharma Kusuma, Indramayu
E-mail: [1]20200810105@uniku.ac.id, *[2]**rioandriyat@uniku.ac.id,** [3]iwanlesmana@uniku.ac.id, [4]lutfirahmawati40@gmail.com

***Abstract***

*Game-based applications are increasingly used beyond entertainment to deliver adaptive, engaging user experiences. Yet, many mobile platformer games still rely on static enemy behaviors, leading to repetitive gameplay. This study introduces Pepe the Ponderland Warrior, a 2D platformer for Android that incorporates culturally relevant meme characters and dynamic NPC behavior using fuzzy logic. Developed with the Game Development Life Cycle (GDLC), the game uses the Fuzzy Sugeno inference system to adapt NPC responses based on player distance, health, and damage received. UML modeling guided the system design, while testing included black-box, white-box, and User Acceptance Testing (UAT). The fuzzy-based system enabled real-time, context-aware NPC decisions, creating more varied and challenging gameplay. The game passed functional and logical testing, with UAT from 30 users producing a high feasibility score of 81.2%, reflecting satisfaction in design, gameplay, and difficulty balance. By integrating fuzzy logic with meme-inspired content, this study offers a novel and efficient AI approach for mobile games, highlighting potential for expansion across platforms and with more adaptive inputs.*

***Keywords***— *fuzzy Sugeno, adaptive NPC, Android game,2D-platformer, pepe the frog, GDLC.*

*Submission: June 19, 2025* *Accepted: July 05, 2025* *Published: July 10, 2025*
*DOI Article: https://doi.org/10.25134/ilkom.v19i2.440*

## 1. INTRODUCTION

Games are interactive systems that allow players to engage in resolving conflicts or challenges through predefined rules, resulting in achievements such as scores, levels, or other objectives [1]. One of the most enduring and popular game genres is the platformer, characterized by player-controlled characters that jump, climb, and navigate obstacles in a two-dimensional (2D) environment with varying terrain [2], [3], [4], [5]. 2D platformer games offer several advantages, including relatively low production costs compared to 3D games, while still providing challenging and addictive gameplay, particularly on mobile platforms [2], [3].

In modern game marketing practices, the use of characters drawn from popular culture, including internet memes, has proven effective in attracting player attention and fostering emotional engagement. One such widely recognized meme character is Pepe the Frog, an iconic figure originating from Matt Furie's Boy's Club comic, which has proliferated across various digital platforms since 2008. Survey data collected from respondents indicate that Pepe possesses strong visual and emotional appeal, making it a promising candidate for use as a main character in a game. However, a search of the Google Play Store reveals that the utilization of this character in digital games remains minimal and has not been optimally developed, especially within the platformer genre. This gap presents a research opportunity to develop a game featuring Pepe that incorporates a more adaptive and dynamic gameplay approach.

With the rapid advancement of artificial intelligence (AI), integrating AI technologies into the behavior design of non-player characters (NPCs) in games has become increasingly common, aiming to enhance player experience. One of the most frequently applied AI methods is fuzzy logic, particularly the Sugeno fuzzy inference system, due to its ability to manage uncertainty in variables such as object distance, health level, and damage received [6], [7], [8], [9], [10]. In a study conducted by Paraschos and Koulouriotis (2025), fuzzy logic was used in a Dynamic Difficulty Adjustment (DDA) system in an AI-based shooter game to maintain a balance between challenge and player comfort across different play styles [11]. Similarly, Kevin et al. (2025), in the Journal of AI and Engineering Applications, implemented 27 fuzzy rules to generate NPC behaviors such as "evade", "defend", and "wait", based on player and environmental conditions in a 2D adventure

***p-ISSN :1858-3911* , *e-ISSN : 2614-5405***
https://journal.fkom.uniku.ac.id/ilkom

game [12]. Another example can be found in the work of Krisdiawan et al. (2023), who applied the Sugeno fuzzy algorithm in a mobile-based game for dyslexia rehabilitation, adjusting difficulty levels according to individual learning progress [7].

Nonetheless, several limitations remain in these existing studies. First, most implementations of fuzzy logic in games focus only on enemy behavior in single scenarios or levels, without considering the complex and dynamic environmental conditions typically found in platformer games. Second, there is a lack of research that integrates popular cultural elements, such as meme characters like Pepe the Frog, as a central gameplay component to enhance players' emotional engagement. Initial surveys conducted by the authors indicate that this character holds significant visual and cultural appeal among teenagers and young adults.

This study aims to address the above gaps by proposing a novel design and development of Pepe the Ponderland Warrior, a 2D platformer game for Android devices. This game uniquely combines the use of the culturally iconic meme character Pepe the Frog with the adaptive capabilities of the Sugeno fuzzy algorithm to regulate NPC behavior. NPC actions are adapted based on parameters such as distance, health, damage, and movement speed; all processed through a Sugeno fuzzy inference system. The game is structured into four progressively challenging thematic levels Cave, Snow, Jungle, and Swamp allowing for the evaluation of AI adaptability under different gameplay scenarios. It is optimized for Android devices with a minimum SDK version of 6 and designed as an offline, single-player game to ensure broad accessibility.

The objectives of this research are: (1) to design and develop the Pepe the Ponderland Warrior 2D platformer game for Android, featuring a popular meme character; and (2) to implement the Sugeno fuzzy algorithm for adaptive and efficient NPC behavior regulation. This study is expected to contribute to the development of AI-based games that not only adapt to individual player styles but also incorporate internet cultural elements as a novel means of enhancing player engagement.

## 2. RESEARCH METHOD

### 1. Data Collection Methods

This research adopts a qualitative-descriptive approach through a software engineering perspective using a system development study. To obtain valid and relevant data for designing Pepe The Ponderland Warrior game, the researcher applied the following data collection techniques:

a. Literature Review

A comprehensive literature review was conducted to understand and build the theoretical foundation supporting system development. Sources included scholarly publications related to interactive game design, non-player character (NPC) behavior modeling in 2D games, and the application of fuzzy logic—particularly the Sugeno model. References were gathered from accredited national journals, international indexed journals (IEEE, Scopus), proceedings, and previous research.

b. Questionnaire

A questionnaire was administered via Google Form to assess user interest in Pepe the Frog as a main game character. Respondents were primarily casual and mobile game players. The goal was to evaluate perceptions on character appeal, gameplay preferences, and adaptive game mechanics expectations.

c. Documentation and System Artifacts

Supporting data included interface mockups, initial storyboards, Game Design Document (GDD), and prototype testing results. Visual documentation served as validation of design decisions and provided evidence during testing and refinement phases of the system.

### 2. Game Development Method

This research employed the **Game Development Life Cycle (GDLC)** model, which provides an iterative and structured framework for game development. GDLC consists of six primary phases [13] :

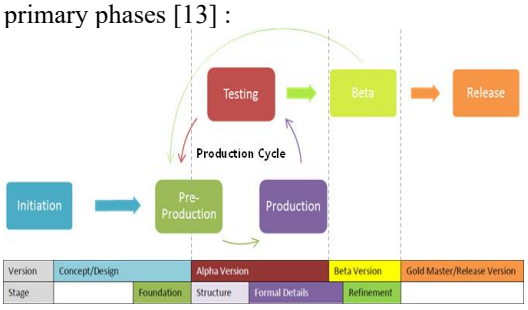



Figure 1. Game Development Life Cycle (GDLC) [3], [13].

a. **Initiation** – Gathering user requirements and ideas through literature review and questionnaire analysis.
b. **Pre-production** – Designing the *Game Design Document (GDD)*, initial interface sketches, and modeling system logic using

***p-ISSN :1858-3911*** **,** ***e-ISSN : 2614-5405***
https://journal.fkom.uniku.ac.id/ilkom

Unified Modeling Language (UML), including use case, activity, sequence, and class diagrams.

c. **Production** – Developing the game in Unity with C# programming, integrating the **Fuzzy Sugeno algorithm** to control NPC behavior.
d. **Testing** – Conducting **black-box testing** for system functionality and **white-box testing** for validating the internal logic of the fuzzy inference system, including cyclomatic complexity analysis.
e. **Beta** – External user testing using **User Acceptance Testing (UAT)** to gather feedback on gameplay and adaptive behavior.
f. **Release** – Final deployment of the game on Android platform (minimum SDK 6) as an offline single-player game.

The GDLC model was chosen for its capacity to support dynamic, user-centered development of interactive and intelligent game systems [13].

## 3. Problem Solving Method (Fuzzy Sugeno Implementation)

To address the challenge of static and non-adaptive NPC behavior, the **Fuzzy Sugeno algorithm** was applied as a decision-making system based on fuzzy inference logic.

### a. Justification for Sugeno Model

Fuzzy Sugeno was selected over alternative models (e.g., Mamdani or machine learning approaches) for the following reasons [11], [12]:

- **Computational efficiency**: Sugeno produces crisp outputs in the form of real-valued linear functions, eliminating the need for complex defuzzification—ideal for resource-constrained Android devices.
- **Handling of uncertain input**: The system efficiently processes uncertain variables such as distance, life points, and damage using fuzzy linguistic sets.
- **Precision in decision-making**: Unlike Mamdani's linguistic outputs, Sugeno supports precise numeric computations, suitable for real-time action control.

### b. Mathematical Formulation of the Sugeno Model

The fuzzy rules are formulated as:

*If* $x_1$ *is* $A_1$ *and* $x_2$ *is* $A_2$ *and* $x_3$ *is* $A_3$*, then* $y = a_1x_1 + a_2x_2 + a_3x_3 + c$ *(1)*

Where:

- $x_1$, $x_2$, $x_3$ are input variables (distance, life, damage)
- $A_1$, $A_2$, $A_3$ are fuzzy linguistic sets (e.g., near, medium, far; low, medium, high)
- y is the output (NPC action), represented by a linear function

The following is a flowchart of the Sugeno fuzzy algorithm:

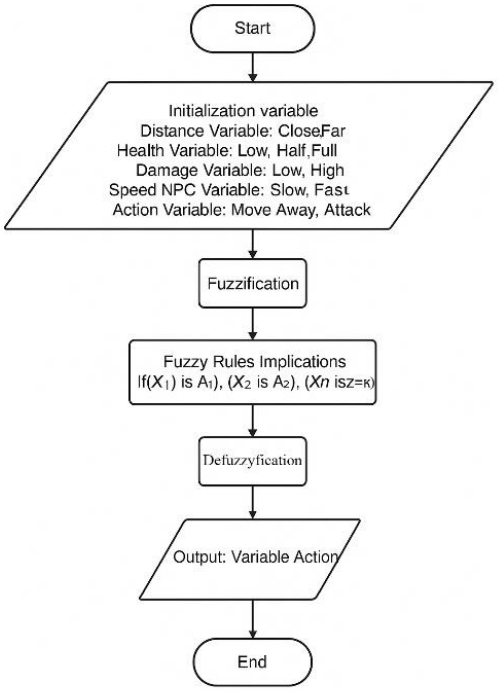



Figure 2. Flowchart of the Sugeno Fuzzy Algorithm

**Inference steps include**:

1. **Fuzzification** – Converting crisp input values into membership degrees.
2. **Rule Evaluation** – Calculating the firing strength using min or product operators.
3. **Output Calculation** – Computing the output for each rule based on its linear function.
4. **Aggregation (Defuzzification)** – Weighted average method:

$$y\ Output = \frac{\sum(wi \cdot yi)}{\sum wi} \qquad (2)$$

Where:

- $wi$ is the **firing strength** of the *i-th* rule (typically calculated using fuzzy logic operators such as minimum or product of membership values),
- $yi$ is the **output** of the *i-th* rule, which in Sugeno models is a **constant or linear function** of the input variables (e.g., y=a1x1+a2x2+).

***p-ISSN :1858-3911*** **,** ***e-ISSN : 2614-5405***
https://journal.fkom.uniku.ac.id/ilkom

## 3. RESEARCH RESULTS

### 3.1 Game Design Document (GDD)

**1. Game Layout Chart**

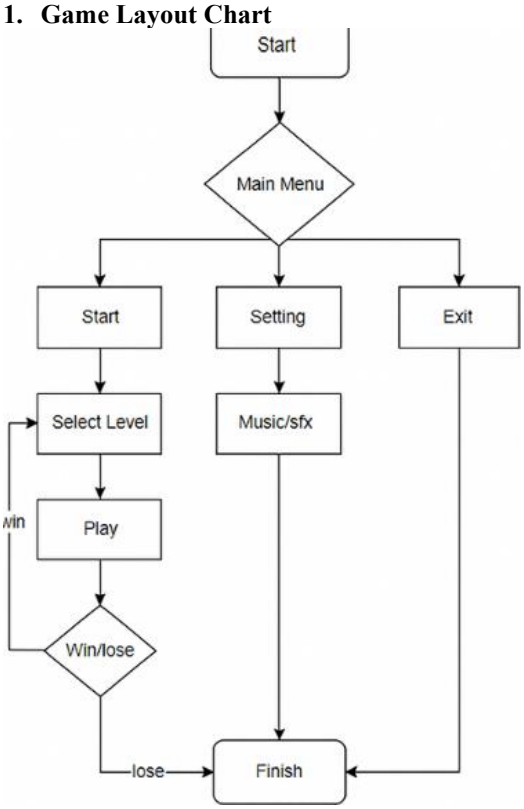



Figure 3. Game layout chart

The sequence of activities in Figure 3 can be described as follows:

1) The game starts from the **Start** node.
2) The **Main Menu** provides three options: **Start**, **Settings**, and **Exit**.
3) Selecting **Start** will lead to the **Level Selection** screen.
4) After choosing a level, the game proceeds to the **Play** stage.
5) The game then evaluates the outcome:
   a. If the player **wins**, they proceed to the next level.
   b. If the player **loses**, the game ends and returns to the **Finish** state.
6) Selecting **Settings** opens the **Music/SFX configuration screen**, where audio settings can be adjusted.
7) Selecting **Exit** will close the application and return to the device's home screen.

**2. StoryBoard Game**

The storyboard game Pepe the Ponderland Warrior is presented in tables 1 to table 3.

Table 1. Story Board Scene 1

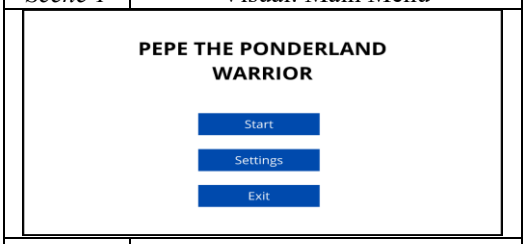
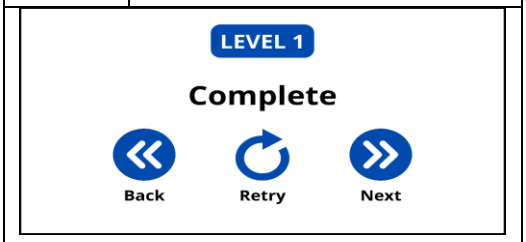
| *Scene 1* | Visual: Main Menu |
|---|---|
| |   |
| Audio | **Backsound**: Soft accompaniment music.<br>**Sound FX**: Sound of click buttons |
| Storyline | In the main menu, players will be given action buttons to select menus and display the game title. |

Table 2. Story Board Scene 2

| *Scene 2* | *Visual: Play Game* |
|---|---|
| |  |
| Audio | **Backsound**: Cheerful accompaniment music.<br>**Sound FX**: Sound of click buttons |
| Storyline | The game page displays the player's health status as lives. Then the player walks to the finish line by pressing the left, right, and jump buttons while passing obstacles and enemies. |

Table 3. Story Board Scene 3

| *Scene 3* | *Visual: Mission Complete* |
|---|---|
| |   |
| Audio | **Backsound**: Happy and cheerful music.<br>**Sound FX**: Sound of click buttons |
| Storyline | After the player has completed the game, there will be a menu to return, restart the game level, or proceed to the next level. |

**3. Game Object**

***p-ISSN :1858-3911*** **,** ***e-ISSN : 2614-5405***
https://journal.fkom.uniku.ac.id/ilkom

Game object yang terdapat pada pepe the ponderland warrior dapat dilihat pada table 4.

Table 4. Game Object

| Game Object | Description | Action |
|---|---|---|
|  | Player pepe | Move forward, backward, up, down. |
|  | Bat NPC (Enemies) | Fly forward, backward, up, down. |
|  | Slime NPC (Enemies) | Move forward, backward, up, down. |
|  | Obstacle | Spinning |
|  | Flag finish game | Object finish game |

## 4. Implementation Fuzzy in Game

A decision is made when the distance is 8, the number of health is 70, and the damage caused is 10.

**Formation of Fuzzy sets**

1. Distance Variable
   a. Close: 0-10
   b. Far: 5-15

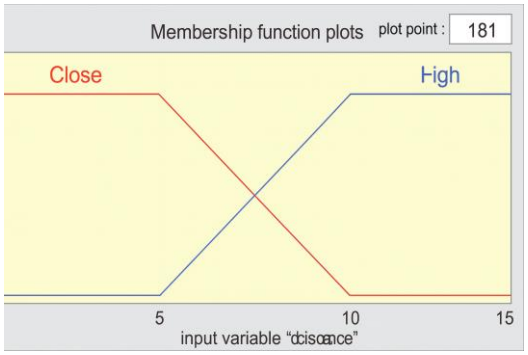



Figure 4. Degree of Membership Distance

$$\mu \text{ Distance close}[5] = \text{distance} > 10 \quad (3)$$
$$\mu \text{ Distance far}[5] = \text{distance} < 5$$

2. Health Variable
   a. Low: 0-50
   b. Half: 25-75
   c. Full: 50-100

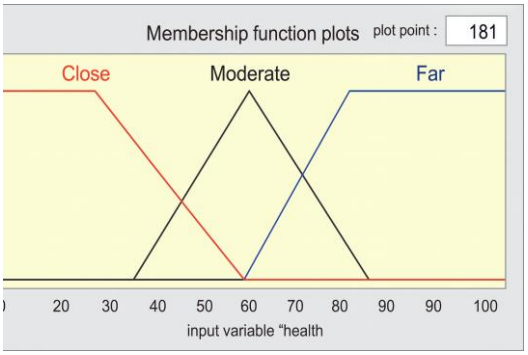



Figure 5. Degree of Membership of the Health Variable

$$\mu \text{ Health low}[45] = \text{health} < 25$$
$$\mu \text{ Health half}[45] = \text{health} \leq 25;\ \text{health} \geq 75 \quad (4)$$
$$\mu \text{ Health full}[45] = \text{health} > 75$$

3. Damage Variable
   a. *low*: 0-12
   b. *high*: 6-20

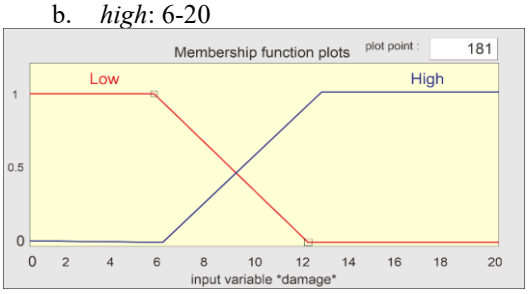



Figure 6. Degree of Membership of the Damage Variable

$$\mu\ damage\ low[15] = damage < 6 \quad (5)$$
$$\mu\ damage\ high[15] = damage > 12$$

**Fuzzy Rules**

Table 5. Fuzzy Rules

| R | J | NM | DM | A |
|---|---|---|---|---|
| R1 | Close | Low | *low* | Move away |
| R2 | Close | Low | *high* | Move away |
| R3 | Close | Low | *low* | Move away |
| R4 | Close | Low | *high* | Move away |
| R5 | Close | Half | *low* | Move away |
| R6 | Close | Half | *high* | Attack |
| R7 | Close | Half | *low* | Move away |
| R8 | Close | Half | *high* | Attack |
| R9 | Close | Full | *low* | Attack |
| R10 | Close | Full | *high* | Attack |
| R11 | Close | Full | *low* | Attack |
| R12 | Close | Full | *high* | Attack |

***p-ISSN :1858-3911*** **,** ***e-ISSN : 2614-5405***
https://journal.fkom.uniku.ac.id/ilkom

| | | | | |
|---|---|---|---|---|
| R13 | Far | Low | *low* | Move away |
| R14 | Far | Low | *high* | Move away |
| R15 | Far | Low | *low* | Move away |
| R16 | Far | Low | *high* | Move away |
| R17 | Far | Half | *low* | Move away |
| R18 | Far | Half | *high* | Attack |
| R19 | Far | Half | *low* | Move away |
| R20 | Far | Half | *high* | Attack |
| R21 | Far | Full | *low* | Attack |
| R22 | Far | Full | *high* | Attack |
| R23 | Far | Full | *low* | Attack |
| R24 | Far | Full | *high* | Attack |

Table description:
1) R = *Rules*
2) J = Distance
3) NM = Health
4) DM = *Damage*
5) A = *Action*

***Deffuzification***

Determine the crisp values of distance, number of health, and damage produced into fuzzy sets and determine their membership degrees. Distance 10, number of Health 80, and damage produced 10.

a. Distance

$$\alpha Close = \mu Close(8) = \frac{10-8}{8-5} = 0{,}66 \quad (6)$$

$$\alpha Far = \mu Far(8) = \frac{8-5}{10-5} = 0{,}6$$

b. Health

$$\alpha Half = \mu Half(70) = \frac{75-70}{70-50} = 0{,}25 \quad (7)$$

$$\alpha Full = \mu Full(70) = \frac{70-50}{75-50} = 0{,}8$$

c. Damage

$$\alpha Low = \mu Low10) = \frac{12-10}{10-6} = 0{,}5 \quad (8)$$

$$\alpha High = \mu High(10) = \frac{10-6}{12-6} = 0{,}66$$

**Implications**

$\alpha Close \cap \alpha Half \cap \alpha Low = 0{,}25$ x 50
$\alpha Close \cap \alpha Half \cap \alpha High = 0{,}25$ x 50
$\alpha Close \cap \alpha Full \cap \alpha Low = 0{,}5$ x 100
$\alpha Close \cap \alpha Full \cap \alpha High = 0{,}66$ x 100
$\alpha Far \cap \alpha Half \cap \alpha Low = 0{,}25$ x 50
$\alpha Far \cap \alpha Half \cap \alpha High = 0{,}25$ x 50
$\alpha Far \cap \alpha Full \cap \alpha Low = 0{,}5$ x 100
$\alpha Far \cap \alpha Full \cap \alpha High = 0{,}6$ x 100

***Deffuzzyfikasi***

Calculate defuzzification using the average formula.

$$Decision = \frac{\Sigma \alpha_i z_i}{\Sigma \alpha_i} \quad (9)$$

$$z = \frac{0{,}25(50)+0{,}25(50)+0{,}5(100)+0{,}66(100)+0{,}25(50)+0{,}25(50)+0{,}5(100)+0{,}6(100)}{0{,}25+0{,}25+0{,}5+0{,}66+0{,}25+0{,}25+0{,}5+0{,}6} \quad (10)$$

$$z = \frac{276}{3{,}26} = 84{,}66$$

*Crisp decision index* = 84,66
*Fuzzy decision index* = 80% Attack, 20% Move away

## 5. Use Case Diagram System Game

a) Use Case Design

A use case diagram is a UML model used to briefly describe who uses the system and what can be done with the system. The use cases for the application to be built [11] are illustrated in Figure 7.

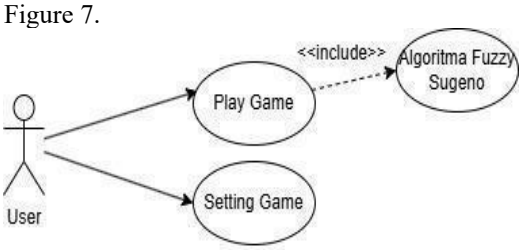



Figure 7 *Usecase diagram*

A case that illustrates the interaction between users (identified as "Users") and the main functionality of a game system. Players, as the main actors, can perform two different actions or use cases: "Play Game" (Start a New Game) and "Game Settings." This diagram provides a high-level overview of the features available to players in this game.

b) *Class Diagram*

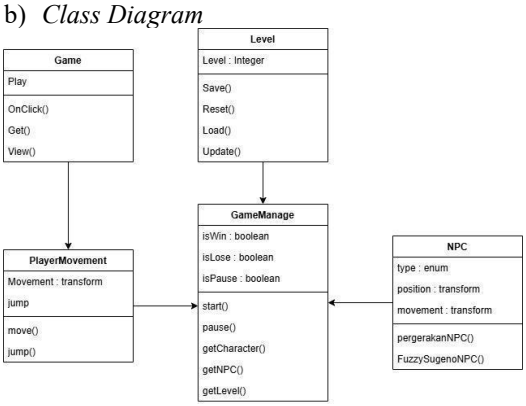



Figure 8. *Class Diagram*.

## 6. Interface Game

a) Main menu Interface

Figure 9 displays the Main Menu Interface, which serves as the initial entry point of the game. It contains three primary options: Start, Settings, and Exit. This interface provides intuitive navigation, allowing players to begin

***p-ISSN :1858-3911* , *e-ISSN : 2614-5405***
https://journal.fkom.uniku.ac.id/ilkom

gameplay, adjust audio preferences, or exit the application.


Figure 9. Main menu interface

b) *Play* Interface

Figure 10 illustrates the Gameplay Interface, where players control the main character using directional buttons: left, right, and up. The interface also features a health bar to display the player's current life status and a pause button that allows the game to be temporarily halted. This layout is designed for accessibility and optimized interaction on touchscreen devices.

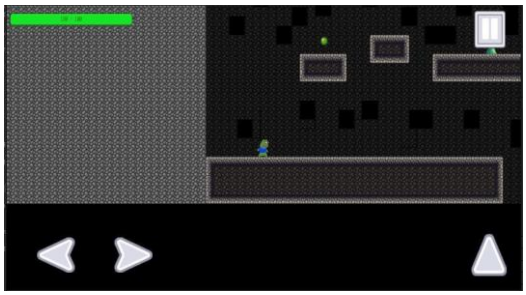


Figure 10. Play Interface

c) Algorithms in game

Figure 11 shows the implementation of the Fuzzy Sugeno algorithm in determining the behavior of non-player characters (NPCs). NPC actions—such as attacking, retreating, or remaining idle—are dynamically adapted based on input variables including distance to the player, health level, and damage received. This intelligent behavior mechanism aligns with the study's objective of providing an adaptive and engaging player experience.

Figure 11. Algorithms in games

d) *Complete* Interface

Figure 12 shows the **Completion Interface**, which appears upon the successful completion of a level. It includes three options: **Back**, **Retry**, and **Next Level**, allowing players to return to the main menu, replay the current level, or proceed to the next stage. This interface supports replayability and user control over game progression.


Figure 14. Complete Interface

## 4. DISCUSSION

### 4.1 Alpha Testing

**1. Blackbox Testing**

Black box testing was conducted to evaluate the functionality of the game application from the user's perspective, without examining the internal code structure. This method focuses on ensuring that the system responds correctly to various inputs and user interactions, in alignment with the functional requirements.

Table 6 summarizes the black box test results for each main feature of the game interface:

Table 6. Black box test results

| No | Test Scenario | Test Case | Expected Result | Actual Result | Conclusion |
|---|---|---|---|---|---|
| **1** | Start Game | Press Start button | Game starts and plays level | Game starts as expected | Passed |
| **2** | Complete Game | Finish a level | Show complete screen | Complete screen displayed | Passed |
| **3** | Pause Game | Press Pause button | Show pause menu | Pause menu displayed | Passed |
| **4** | Lose Game | Player defeated | Show fail screen | Fail screen displayed | Passed |
| **5** | Access Settings | Press Settings button | Display music/ SFX options | Settings displayed properly | Passed |

***p-ISSN :1858-3911*** **,** ***e-ISSN : 2614-5405***
https://journal.fkom.uniku.ac.id/ilkom

All tested features functioned according to the intended design, confirming that the interface elements and core gameplay mechanics were implemented correctly.

**2. Whitebox Testing**

White box testing was used to examine the internal logic of the game, particularly the implementation of the **Fuzzy Sugeno algorithm** in the NPC decision-making system. Cyclomatic complexity analysis was applied to measure the logical paths within the fuzzy evaluation method.

Table 7. White Box Testing of Fuzzy Sugeno Algorithm

| No. | Code program |
|---|---|
| 1 | *public class FuzzySugenoSystem : MonoBehaviour*<br>*{public List<FuzzyVariable> inputVariables = new List<FuzzyVariable>();*<br>*public List<SugenoRule> rules = new List<SugenoRule>();* |
| 2 | *public float Evaluate(params float[] inputValues)*<br>*{List<float> firingStrengths = new List<float>();*<br>*List<float> ruleOutputs = new List<float>();* |
| 3 | *foreach (var rule in rules)*<br>*{float firingStrength = 1f;* |
| 4 | *foreach (var condition in rule.conditions){* |
| 5 | *foreach (var variable in inputVariables){* |
| 6 | *if (variable.name == condition.variableName)*<br>*{int inputIndex = inputVariables.IndexOf(variable);* |
| 7 | *if (inputIndex < inputValues.Length)*<br>*{float membership = variable.GetMembership*<br>*(condition.fuzzySetName, inputValues[inputIndex]);*<br>*firingStrength = Mathf.Min(firingStrength, membership);*<br>*}}}} firingStrengths.Add(firingStrength);*<br>*float output = rule.CalculatedOutput(inputValues);*<br>*ruleOutputs.Add(output);*<br>*}*<br>*float sumWeightOutput = 0f;*<br>*float sumWeight = 0f;* |
| 8 | *for (int i = 0; i < rules.Count; i++)*<br>*{*<br>*Debug.Log("rule " + rules[i].name +" : " + firingStrengths[i] + " " + ruleOutputs[i]);*<br>*sumWeightOutput += firingStrengths[i] * ruleOutputs[i];*<br>*sumWeight += firingStrengths[i];*<br>*}* |
| 9 | *return sumWeight > 0 ? sumWeightOutput / sumWeight : 0f;*<br>*}*<br>*}* |

Based on Table 7, Flowgraph Cyclomatic Complexity was obtained to conduct path-based testing by calculating cyclomatic complexity. The following is an image of Flowgraph Cyclomatic Complexity.

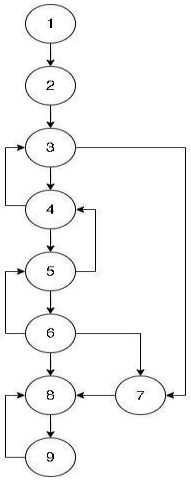



Figure 15. *Flowgraph Cyclomatic Complexity*

Calculation of cyclomatic complexity, using the formula:

$$V(G) = E - N + 2 \qquad (11)$$

From the previous flowgraph, the following values can be determined:

E = 14; N = 9;

After entering these values into the formula, the result is:

V(G) = E – N + 2

V(G) = 14 – 9 + 2 = 7

From the cyclomatic complexity calculation, there are 7 independent paths, namely:

1. Path 1: 1 → 2 → 3 → 4 → 5 → 6 → 8 → 9
2. Path 2: 1 → 2 → 3 → 4 → 5 → 6 → 7 → 8 → 9
3. Path 3: 1 → 2 → 3 → 4 → 5 → 3 → 4 → 5 → 6 → 8 → 9
4. Path 4: 1 → 2 → 3 → 4 → 3 → 4 → 5 → 6 → 8 → 9
5. Path 5: 1 → 2 → 3 → 4 → 5 → 6 → 8 → 9 → 5 → 6 → 8 → 9
6. Path 6: 1 → 2 → 3 → 4 → 5 → 6 → 7 → 3 → 4 → 5 → 6 → 8 → 9
7. Path 7: 1 → 2 → 3 → 4 → 5 → 6 → 8 → 9 → 3 → 4 → 5 → 6 → 7 → 8 → 9

The result indicates that the method contains 7 independent logical paths. These paths were traced and tested individually to ensure complete path coverage and logical correctness in decision output. The implementation was verified to generate consistent and context-sensitive NPC behavior during gameplay.

***p-ISSN :1858-3911*** **,** ***e-ISSN : 2614-5405***
https://journal.fkom.uniku.ac.id/ilkom

### 4.2 Beta Testing (UAT Testing)

The User Acceptance Test (UAT) was conducted with 30 respondents to evaluate the game's usability, interface clarity, character control, and the adaptiveness of NPC behavior. Participants assessed six key indicators using a Likert scale: Strongly Agree (5), Agree (4), Neutral (3), Disagree (2), and Strongly Disagree (1).

Table 8 presents the total weighted score for each question based on the assigned weights. The maximum possible score was 900, calculated from 30 respondents × 6 questions × 5 points.

Table 8. UAT Questionnaire Result

| Indicator | Score |
|---|---|
| Visual appeal of the main menu | 140 |
| Ease of understanding the game menu | 132 |
| Character Control | 142 |
| Relevance of Pepe the Frog as a character | 139 |
| Balance in gameplay difficulty | 137 |
| Adaptiveness of enemy behavior (challenge vs fairness) | 141 |
| **Total** | **731 / 900** |

The overall eligibility percentage is calculated as:

$$Eligibelity(\%) = \frac{731}{900} = 81.2\ \%$$

This result indicates that the game *Pepe The Ponderland Warrior* is considered **acceptable and well-received** by users. Most respondents reported that the game interface was intuitive, character control was accessible, and the implementation of NPC behavior using the Fuzzy Sugeno algorithm created a fair yet challenging gameplay experience. More detailed visualization can be seen in Figure 16.

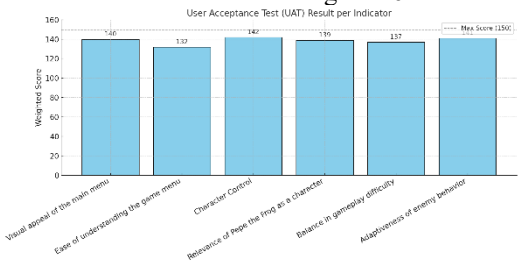



Figure 16. Visualizes User Acceptance Test (UAT) Result Per Indicator

## 5. CONCLUSION

This study has successfully designed and developed a 2D platformer game titled Pepe The Ponderland Warrior, targeting the Android platform, with a key focus on implementing the Fuzzy Sugeno algorithm to control adaptive NPC behavior. The development process followed the Game Development Life Cycle (GDLC) model, ensuring iterative design, prototyping, testing, and release.

The main contribution of this research lies in its novel combination of adaptive AI behavior with cultural engagement using Pepe the Frog, a popular internet meme, as the central character. This approach not only introduces unique visual appeal but also addresses the market gap—the absence of meme-based platformer games that integrate intelligent difficulty adjustment mechanisms.

From the AI perspective, the implementation of the Fuzzy Sugeno inference system enabled NPCs to respond dynamically to real-time game conditions such as distance to the player, health status, and damage received. Compared to traditional rule-based or static behavior models, this approach produced more responsive, context-aware NPCs, leading to a richer gameplay experience.

The system was validated through functional testing (black-box and white-box) and User Acceptance Testing (UAT) involving 30 respondents. The game achieved a feasibility score of 81.2%, indicating that the majority of users perceived the gameplay as engaging, balanced, and enjoyable.

In conclusion, this research demonstrates the effectiveness of combining fuzzy logic with platformer game mechanics to create personalized, adaptive, and immersive player experiences. It provides a strong foundation for the future development of AI-driven game systems and serves as a reference for incorporating fuzzy decision-making models in resource-constrained mobile environments..

## 6. SUGGESTIONS

Based on the results and findings of this research, several suggestions can be offered to guide future development and research in related areas. First, the game Pepe The Ponderland Warrior can be further enhanced by expanding the number of levels, integrating more complex environmental elements (e.g., moving platforms, traps, or dynamic lighting), and introducing additional NPC behavior types beyond simple attack or retreat patterns. Second, the implementation of the Fuzzy Sugeno algorithm can be extended by incorporating more input variables such as player performance history or reaction time to achieve a finer level of difficulty personalization. Third, cross-platform compatibility (e.g., iOS or web-

***p-ISSN :1858-3911* , *e-ISSN : 2614-5405***
https://journal.fkom.uniku.ac.id/ilkom

based deployment) should be considered to reach broader user segments. Lastly, future studies are encouraged to explore comparative analysis between Fuzzy Sugeno and other adaptive AI methods such as reinforcement learning or hybrid neuro-fuzzy systems to evaluate performance, scalability, and user engagement in various game genres.

## REFERENCES

[1] K. S. , & Z. E. Tekinbas, *Rules of play: Game design fundamentals*. MIT press, 2003. Accessed: Jul. 04, 2025. [Online]. Available: https://books.google.co.id/books?hl=en&lr=&id=YrT4DwAAQBAJ&oi=fnd&pg=PR13&dq=related:AAzhZBxv6GkJ:scholar.google.com/&ots=fqUCah7omY&sig=_FV4eX9XQAZHhPQHcDZ0o5TPq-U&redir_esc=y#v=onepage&q&f=false

[2] Octodinata, S. B., and & H. I., "Rancang Bangun Game Edukasi Genre Platformer 2D Berbasis Android," *Jurnal RESTI*, vol. 2, no. 7, pp. 318–325, 2023.

[3] R. A. Krisdiawan and Darsanto, "PENERAPAN MODEL PENGEMBANGAN GAME GDLC (GAME DEVELOPMENT LIFE CYCLE )DALAM MEMBANGUN GAME PLATFORM BERBASIS MOBILE," *TEKNOKOM*, Mar. 2019, doi: https://doi.org/10.31943/teknokom.v2i1.33.

[4] A. F. Kusumah and R. Kurniawan, "Development of a Serious Game as an Educational Tool for Obesity Prevention," *Jurnal Sains, Nalar, dan Aplikasi Teknologi Informasi*, vol. 3, no. 2, pp. 58–63, Jan. 2024, doi: https://doi.org/10.20885/snati.v3.i2.32.

[5] F. A. Jellyanto, I. Kuswardayan, and N. Suciati, "Rancang Bangun Pembangkit World Dinamis menggunakan Algoritma Recursive Backtracking pada Game 2D Platformer 'Mine Meander,'" *Jurnal Teknik ITS*, vol. 5, no. 2, Oct. 2016, doi: https://doi.org/10.12962/j23373539.v5i2.19364.

[6] R. A. Krisdiawan, M. F. Rohmana, and A. Permana, "Pembuatan Game Runaway From Culik Dengan Algoritma Fuzzy Mamdani," *Buffer Informatika*, vol. 6, no. 1, pp. 33–40, Jun. 2020, doi: https://doi.org/10.25134/buffer.v6i1.2623.

[7] R. A. Krisdiawan, T. Sugiharto, and N. Nugraha, "Terapi Rehabilitasi Dyslexia Berbasis Mobile Game dengan Dynamic Difficulty Adjustment (DDA) Menggunakan Algoritma Fuzzy Sugeno," *Jurnal Teknologi Informatika dan Komputer*, vol. 9, no. 2, pp. 975–991, Sep. 2023, doi: https://doi.org/10.37012/jtik.v9i2.1837.

[8] R. D. Anggraeni, F. Nugroho, and J. N. Fadila, "Implementasi Algoritma Fuzzy Sugeno Pada Game Pious Boy," *Jurnal Informatika: Jurnal Pengembangan IT*, vol. 6, no. 1, pp. 26–28, Jan. 2021, doi: https://doi.org/10.30591/jpit.v6i1.2321.

[9] Z. Zhu and M. Zhu, "Evaluation Method of Performance of Cross-Border e-Commerce System Based on Fuzzy DEA Model," *Mobile Information Systems*, vol. 2022, 2022, doi: https://doi.org/10.1155/2022/1456584.

[10] U. Nurhasan, "Analisis Perilaku Non Playable Character (Npc) Pada Game Menggunakan Fuzzy Sugeno," *Techno.Com*, vol. 19, no. 3, pp. 308–320, Aug. 2020, doi: https://doi.org/10.33633/tc.v19i3.3477.

[11] P. D. Paraschos and D. E. Koulouriotis, "Fuzzy Logic-Based Dynamic Difficulty Adjustment for Adaptive Game Environments," *Electronics (Basel)*, vol. 14, no. 1, p. 146, Jan. 2025, doi: https://doi.org/10.3390/electronics14010146.

[12] Kevin, Octara Pribadi, and Hendri, "Implementation of Fuzzy Logic Algorithm to Improve NPC Decision-Making in 2D Adventure Games Using Unity," *Journal of Artificial Intelligence and Engineering Applications (JAIEA)*, vol. 4, no. 3, pp. 2376–2381, Jun. 2025, doi: https://doi.org/10.59934/jaiea.v4i3.1175.

[13] R. A. Krisdiawan, "Implementasi Model Pengembangan Sistem GDLC dan Algoritma Linear Congruential Generator Pada Game Puzzle," *Nuansa Informatika*, vol. 12, no. 2, pp. 1–9, Mar. 2018, doi: https://doi.org/10.25134/nuansa.v12i2.1634.