Review

Not peer-reviewed version

An Analysis of Finite State Machine Based Enemy Artificial Intelligence in Kirby: Nightmare in Dream Land

Vathanak Thyrun *

Posted Date: 20 January 2026

doi: 10.20944/preprints202601.1473.v1

Keywords: game artificial intelligence; finite state machine; enemy behavior; platform games; difficulty balancing

Review

An Analysis of Finite State Machine Based Enemy Artificial Intelligence in Kirby: Nightmare in Dream Land

Vathanak Thyrun

Independent Researcher, Cambodia; vathanak85thyrun@gmail.com

Abstract

The type of AI used to design video game enemies greatly affects the gameplay experience of speed, difficulty, and enjoyment. In most cases, the majority of developers who create 2D platforming games will choose to implement a simple but efficient AI design over an advanced AI model that learns based on experience. One of these simpler AI models that is frequently utilized by 2D platforming game developers is the Finite State Machine (FSM) model. The FSM model creates an organization of the enemy's actions into a limited number of well-defined behaviours, while also indicating how these behaviours relate to one another. We look at how AI uses the FSM method in the 2D platform game "Kirby: Nightmare in Dreamland," which first came out on the Game Boy Advance. The analysis of FSM models enemy AI behaviors and how those behaviors change and when they do so affects how hard the game is. Simulated experiments were conducted on how state time is spread out and how to make things harder by changing the attack cooldown. The results show that FSM-based AI is easy to control, doesn't need a lot of processing power, and has behavior that can be predicted. This makes it a good choice for platform games that are easy to get into. The results show that FSMs are still important in AI research and game design today.

Keywords: game artificial intelligence; finite state machine; enemy behavior; platform games; difficulty balancing

1. Introduction

AI in video games is mostly used to make the game more fun for the player. In a lot of classic and 2D platform games, the enemy AI has to find a balance between being hard, easy to guess, and easy to get to. Developers often use simple but effective ways to make enemies in 2D platformer games behave in a way that is consistent and reliable [1]. One of the best ways for game developers to plan how enemies will act is to use finite state machines. This is because they let you set up a small number of states that an enemy can move between and clearly define all of the enemy's possible actions. FSMs are also computationally sound and easy to use, and they give the designer a lot of control over how hard or easy they want the game to be [6].

The main goal of this study is to show how finite state machine based Artificial Intelligence (AI) can help make platform games more balanced and why it can still be a good way to control enemy behavior in 2D platforms over time. The design of Kirby: Nightmare in Dream Land (simple, beginner-friendly, and based on patterns) makes it a good game to use FSM technology to learn about its benefits. The AI for both the game's enemies and bosses is very predictable and reacts to what the player does [5]. This makes it possible for players to learn how to spot patterns in their opponents' attacks and practice to get better at dealing with those attacks. This paper will talk about how the FSM algorithm used to control enemy behavior in Kirby: Nightmare in Dream Land is related to state transitions, the amount of time each FSM state spends in each state, and the different levels of difficulty that can be created by changing FSM parameters.

 doi:10.20944/preprints202601.1473.v1

2. Background and Related Work

Over the past decade, Game AI has come a long way from simple Rule-based AI to the more advanced Machine Learning and Neural Network Models we have today. Even with these improvements, Finite State Machines (FSM) are still one of the most common AI methods used in the commercial game industry [2]. This is especially true for games like platformers, fighting games, and action-adventure games. There are many benefits to FSMs. They are easy to set up and debug, their behavior is predictable and clear, which makes them efficient from a computational point of view, and their built-in efficiency makes them perfect for making video games for low-end hardware where performance and reliability are the most important design factors [4].

Prior research and industry literature indicate that numerous methods for employing FSMs to simulate opponent behavior, when the designer aims for fairness and predictability, resemble "easy" games for beginners [3]. The Kirby series has the same design idea: it is for a lot of different types of people, including younger and more casual players. You can change the speed and difficulty of a game more easily with FSMs than with other types of AI, like behavior trees and reinforcement learning. Some advanced AI systems can make characters act differently than they normally do, which makes it hard to guess how the player will be treated. This makes the game less fun for players. This shows that FSMs are still important for making games today.

3. Finite State Machine Model

3.1. Overview of Finite State Machines

A Finite State Machine (FSM) is a well-established method of representing a range of behaviours within a logical framework [4]. It is a type of computer model that has a limited number of states, rules for when transitions happen, and a set of transitions between those states. The system is always in one state. Events or conditions, like a player getting close to another player or taking damage, can cause transitions.

FSMs are often used in video games to control how enemies act. Each state is linked to a certain action or behavior, and transitions decide how enemies react to what the player does.

3.2. FSM States in Kirby

Based on observed gameplay behavior, enemy AI in Kirby: Nightmare in Dream Land can be modeled using the following states:

State	Description
Idle	Enemy remains inactive or stationary
Patrol	Enemy moves in a predefined pattern
Attack	Enemy performs an offensive action
Hit	Enemy reacts to receiving damage
Recover	Temporary invulnerability or delay
Defeated	Enemy is removed from the game

3.3. State Transition Logic

Enemy behavior transitions are governed by simple rule-based logic. A simplified version of the FSM algorithm is shown below:

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 20 January 2026 doi:10.20944/preprints202601.1473.v1


```
state = IDLE

while enemy_alive:

   if state == IDLE and player_detected:
      state = ATTACK

   elif state == ATTACK and enemy_hit:
      state = HIT

   elif state == HIT:
      state = RECOVER

   elif state == RECOVER and timer_expired:
      state = IDLE
```

This structure ensures that enemy behavior remains consistent and predictable while still responding to player interaction [4].

4. Methodology

4.1. Experimental Design

Due to the game engine's source code not open to the public, tests were done using simulated observations based on playing the game over and over. The experiments were done with only one kind of enemy in a controlled setting.

The following parameters were fixed:

- Enemy health points
- Player ability
- Environment layout

FSM timing parameters, particularly attack cooldown duration, were varied to analyze difficulty scaling.

4.2. State Probability Calculation

To quantitatively analyze enemy behavior, the probability of each FSM state was computed based on the proportion of time the enemy spent in that state during an encounter. The state probability is defined as:

$$P(s) = \frac{T_s}{T_{total}}$$

where Ts represents the time spent in state s, and Ttotal is the total encounter duration. This calculation allows direct comparison of how dominant each behavior state is during gameplay [6].

4.3. Difficulty Metric

Gameplay difficulty was quantified using the following metric [3]:

$$D = \frac{\text{Damage to player}}{\text{Encounter duration}}$$

 doi:10.20944/preprints202601.1473.v1

This formula reflects the intensity of the encounter by combining both player damage and encounter length.

5. Results

5.1. FSM State Time Distribution

The average time spent in each FSM state during an enemy encounter was measured across multiple trials.

FSM State	Avg Time (s)	Percentage (%)
Idle	10.2	25.5
Patrol	6.8	17.0
Attack	15.6	39.0
Hit	4.1	10.3
Recover	3.3	8.2

The FSM analysis of enemy behavior was performed over 20 runs against an enemy under conditions where the player had control over enemy encounters, with the duration spent in each of the FSM states Idle, Patrol, Attack, Hit, and Recover being recorded from the beginning of the encounter to the point where the enemy was defeated. All trials were conducted using identical enemy types, player abilities, and layouts in order to provide maximum consistency and minimise external variance.

The FSM state times were calculated by averaging them over all 20 runs to provide a representative sample. The average state duration was calculated by taking the mean of the recorded durations of each individual state, and the total encounter duration was calculated by summing these average durations of all states. The percentages of each state were then calculated by dividing the average duration of each state by the total duration of all states, thereby reducing the effect of outliers and providing a stable estimation of the distribution of enemy behavior for the FSM model. The results show that the Attack state dominates enemy behavior, while Idle and Patrol states provide recovery opportunities for the player.

5.2. Difficulty Scaling Results

Adjusting the attack cooldown significantly influenced difficulty.

Attack Cooldown (s)	Avg Player Damage	Fight Duration (s)	Difficulty Score
2.0	8	55	0.15
1.5	12	45	0.27
1.0	18	38	0.47
0.5	25	30	0.83

To assess the influence of FSM tuning parameters on the difficulty level associated with gameplay, the second experiment focused on manipulating the attack cooldown feature within the enemy FSM. 20 independent runs were completed for each attack cooldown variation (2s, 1.5s, 1s and 0.5s). All other factors (enemy health and player power) remained the same during the experiment so that differences seen would be a result only of differences created by alterations in the timing of attack cool-downs.

 doi:10.20944/preprints202601.1473.v1

We kept track of two main numbers for each run: the total damage the player took and the total time it took to beat the enemy, from the start of the fight to the end. The numbers in Table show the average damage done by players and the average length of fights for each cooldown setting over the 20 runs. A difficulty score was calculated using a time normalized damage metric, which is the average player damage divided by the average encounter duration. This score was used to measure how hard the encounter was overall. This metric shows how intense an encounter is by measuring how hard the enemy hits and how quickly they put pressure on the player.

6. Discussion

According to this study, the Finite State Machine (FSM) model is a useful and efficient way to control how enemies act in Kirby: Nightmare In Dream Land. The distribution of FSM states shows that there is a conscious effort to balance aggression with passivity. The Attack state is the biggest (in terms of the time spent in that FSM state compared to other FSM states). Idle and Patrol both make time windows that show when the player is likely to recover. These different behaviors together make it easier for players to get to the game as a whole. The FSM model doesn't make players use random or adaptive AI behaviors. Instead, it teaches them to spot and use repetitive patterns to their advantage [5], which is how the Kirby series is played.

The experiments show that FSM AI is great at making hard tasks easier. By changing just one variable in the FSM (the time between attacks), the user can make the fight against the AI easier or harder. You don't need to change the damage, health, or ability to attack. The results suggest that players deal more damage and finish their opponents faster when they have less time between attacks than when they have more time. This is an example of how changing the time between attacks can help designers change the game's difficulty level. So, FSM AI is a great choice for video game designers who want to make sure that everyone has a good time, no matter how good they are at the game. FSMs are easy to design and use, but they can also make things harder or easier by changing the timing.

FSMs are very stable and don't need much computing power, which makes them great for platforms with limited hardware resources [4]. Also, because FSMs are easier to run, they will use less CPU than more complicated AI systems. So, when we need enemies to always respond the same way, FSMs are a great way to do it. Even though there may be some problems with making behaviors more complicated or emergent, these problems aren't as big of a deal in Kirby: Nightmare In Dream Land because the main goal is for an enemy to act in a predictable way. Based on this analysis, it looks like FSMs are still useful for 2D platformers because they still find the right balance between making the game hard enough to be fun and easy enough to use.

8. Conclusion

The research outlined in this paper focused on how Enemy AI systems are generally implemented using FSM models in the game Kirby: Nightmare in Dream Land. As described throughout the article, an analysis of FSM models was completed, resulting in conclusions regarding the application of their utility to create Predictable, Efficient and easily Tuned Enemy Behaviours for designers' control of difficulty and pacing of the Game's Play. By changing things like "Attack Timing" as an example, designers were able to create a different experience for players without creating additional complexity for System level development. Overall, FSMs are fundamental to Enemy Behaviour Modelling within the 2D Platform genre, while maintaining Accessibility, Balance, and Learning for players. From our perspective, FSMs continue to represent valid methodologies for Game AI designs today.

References

1. D. Jagdale, "Finite State Machine in Game Development," ResearchGate, 2021. https://www.researchgate.net/publication/355518086_Finite_State_Machine_in_Game_Development

 doi:10.20944/preprints202601.1473.v1

2. M. B. R. Alvian, "Implementation of Finite State Machine to Determine The Behaviour of Non-Playable Character in Leadership Simulation Game," Journal of Games, Game Art and Gamification, 2024. https://journal.binus.ac.id/index.php/jggag/article/view/10894
3. M. Weber et al., "Dynamic Difficulty Adjustment in Digital Games Using Genetic Algorithms," SBGames 2020. https://www.sbgames.org/proceedings2020/ComputacaoFull/209636.pdf
4. Newcastle University, "Artificial Intelligence 1: Finite State Machines," School of Computing, 2016. https://research.ncl.ac.uk/game/mastersdegree/gametechnologies/previousinformation/artificialintelligence1finitestatemachines/2016%20Tutorial%208%20-%20Finite%20State%20Machines.pdf
5. S. Chen, "Literature Review of Application of AI in Improving Gaming Experience: NPC Behavior," Master's thesis, LUT University, 2024. https://lutpub.lut.fi/bitstream/handle/10024/167834/mastersthesis_Chen_Shuao.pdf
6. R. Andrea, "Modeling of Enemy Behavior Using a Finite State Machine," International Journal of Modern Education and Computer Science (IJMECS), 2025. https://www.mecs-press.org/ijmecs/ijmecs-v13-n1/IJMECS-V13-N1-4.pdf
7. NesHacker, "How NES Games Use State Machines For Everything," YouTube, Apr. 7, 2023. https://www.youtube.com/watch?v=8lZ53Sx5oc0.