

## Implementasi *Hybrid Finite State Machine* dan *Fuzzy Logic Tsukamoto* pada Perilaku Adaptif *Non-Player Character Game*

Dhavid Bhertus<sup>1\*</sup>

<sup>1</sup>Institut Teknologi & Bisnis Sabda Setia, Indonesia

<sup>1</sup>[dhavid.bhertus@itbss.ac.id](mailto:dhavid.bhertus@itbss.ac.id)



### Histori Artikel:

Diajukan: 7 Juli 2026

Disetujui: 25 September 2026

Dipublikasi: 30 September 2026

### Kata Kunci:

*Fuzzy Logic, Finite State Machine, Game, Non-Player Character, Perilaku Adaptif*

### Digital Transformation

*Technology (Digitech) is an Creative Commons License This work is licensed under a Creative Commons Attribution-NonCommercial 4.0 International (CC BY-NC 4.0).*

### Abstrak

Perilaku *Non-Player Character* (NPC) yang monoton akibat penggunaan aturan transisi yang kaku pada metode *Finite State Machine* (FSM) dapat mengurangi pengalaman imersif pemain dalam permainan digital. Integrasi logika *fuzzy* berpotensi meningkatkan fleksibilitas pengambilan keputusan NPC, namun kompleksitas komputasi yang ditimbulkan dapat memengaruhi performa permainan. Penelitian ini bertujuan menganalisis pengaruh integrasi *Fuzzy Logic Tsukamoto* ke dalam FSM terhadap perilaku NPC serta membandingkan dampak implementasi *fuzzy* secara parsial dan keseluruhan terhadap stabilitas performa game. Metode pengembangan yang digunakan adalah *Rapid Application Development* (RAD) dengan implementasi sistem pada *Game Engine Unity* menggunakan bahasa pemrograman C#. Pengujian dilakukan melalui *Black-box Testing* untuk memverifikasi fungsionalitas sistem dan pengukuran performa menggunakan parameter *Frame Per Second* (FPS) serta waktu *scripting* melalui *Unity Profiler*. Hasil pengujian menunjukkan bahwa seluruh skenario fungsional berhasil dijalankan sesuai kebutuhan sistem. Integrasi *Fuzzy Logic Tsukamoto* menghasilkan perilaku NPC yang lebih adaptif dibandingkan FSM murni karena keputusan dipengaruhi oleh kombinasi variabel jarak dan kecepatan pemain. Pengujian performa menunjukkan implementasi *Full Fuzzy* memperoleh rata-rata 78,11 FPS dengan waktu *scripting* 2,374 ms, sedangkan implementasi *Fuzzy Parsial* memperoleh 79,58 FPS dengan waktu *scripting* 2,334 ms. Perbedaan performa yang relatif kecil menunjukkan bahwa integrasi logika *fuzzy* tidak memberikan dampak signifikan terhadap stabilitas permainan. Dengan demikian, kombinasi FSM dan *Fuzzy Logic Tsukamoto* efektif meningkatkan fleksibilitas perilaku NPC tanpa mengorbankan performa game secara berarti.

## PENDAHULUAN

Perkembangan industri *video game* akhir-akhir ini mengalami perkembangan yang sangat pesat dikarenakan perannya sebagai salah satu sarana hiburan bagi generasi muda. Berdasarkan data yang diambil dari Pranselga et al (2021), tercatat bahwa jumlah *gamer* di Indonesia tergolong cukup besar, yakni antara 60 hingga 100 juta orang pada tahun 2020 dengan rata-rata mereka menghabiskan waktu mereka sekitar 25% per harinya. Namun, seiring dengan semakin melesatnya industri *game*, begitu juga tantangan yang perlu dihadapi oleh bidang ini. Pemain merasa jenuh dan bosan karena mekanik permainan yang monoton dan perilaku musuh yang berulang dan nantinya hal ini akan berdampak pada kurangnya pengalaman imersif pemain (Putera & Murti, 2018, dalam Al Kautsar et al., 2024).

Mekanik permainan merupakan salah satu elemen penting yang akan dinilai terlebih dahulu saat pemain memainkan *game* kita. Mekanik permainan adalah sebuah pilar dari permainan yang terdiri dari banyak hal, seperti pergerakan, interaksi, bahkan animasi karakter itu sendiri. Hal-hal seperti ini cenderung sering diabaikan oleh pengembang *game* pemula padahal meskipun terkesan seperti masalah yang kecil, unsur-unsur ini mampu menambahkan pengalaman imersif pemain untuk membuat karakter dalam *game* terkesan lebih natural dan tidak terkesan kaku maupun monoton (Al Kautsar et al., 2024). Oleh karena itu, sebuah metode berbasis kecerdasan buatan yang dinamakan sebagai *Finite State Machine* (FSM) dikembangkan guna menangani masalah seperti yang telah disebutkan sebelumnya. Metode FSM merupakan sebuah metode yang digunakan untuk mengatur animasi pergerakan karakter secara dinamis berdasarkan kondisi-kondisi tertentu yang telah dirancang oleh pengembang (Abdurrohman et al., 2023).

Namun, metode FSM tidaklah cukup untuk memberikan kesan natural pada karakter dan masih jauh dari kondisi manusia sesungguhnya yang sangat adaptif dan reaktif jika dihadapkan pada kondisi-kondisi tertentu. Metode FSM sendiri pada hakikatnya menggunakan angka-angka numerik sebagai sebuah batasan pada karakter

untuk berpindah dari satu kondisi (*state*) ke kondisi lainnya, sebagai contoh karakter akan mulai mengejar pemain jika jarak antar karakter kurang dari 7 *units* (Setiawan et al., 2021). Hal ini tentunya dapat berakibat fatal karena pada kenyataannya manusia tidak bereaksi terhadap suatu hal berdasarkan aturan kaku. Kondisi ini yang merupakan batasan dari metode FSM karena metode terkait tidak dapat menangani masalah dalam rentang seperti ini (Setiawan et al., 2021).

Masalah serupa ditemukan pada penelitian terdahulu oleh Abdurrohman, Pranoto, dan Prasetya (2023) yang berfokus pada penerapan FSM ini dalam *game* 3D. Hal serupa juga diteliti oleh Pranselga, Setiawan dan Apriandari (2021) namun pada *game engine* yang berbeda dari sebelumnya meskipun masalah terkait tidak disebutkan secara eksplisit. Penelitian yang diangkat oleh Setiawan, Wibowo dan Pranoto (2021) dianggap sebagai terobosan baru karena mereka mencoba menggabungkan logika *fuzzy* dalam FSM agar mekanik permainan menjadi lebih alami. Namun berdasarkan hasil pengujiannya, pengintegrasian ini justru berdampak sebaliknya, yakni permainan yang telah dibuat hanya mendapat nilai cukup yang mendominasi sebesar 52% dari para responden (Setiawan et al., 2021). Terdapat celah penting di mana pergerakan alami dari para NPC justru mengurangi pengalaman imersif pemain, seperti penambahan beban pada permainan sehingga menciptakan *bug*, *glitch* ataupun *error* tertentu. Hal ini juga senada dengan yang dikemukakan oleh Fedotova et al. (2023) bahwa penambahan logika kompleks pada bagian *scripting* dapat menambah beban komputasi pada GPU maupun CPU.

Berdasarkan segala permasalahan yang telah dipaparkan sebelumnya, maka penelitian ini akan berfokus pada pengintegrasian metode FSM dengan logika *fuzzy* secara parsial. Hal ini dilakukan karena mempertimbangkan kemampuan dari *game engine* yang digunakan serta perangkat dari pemain yang ditujukan. Penerapan secara massal dapat menyebabkan penurunan performa *game* yang justru mengurangi pengalaman imersif pemain. Oleh sebab itu penerapan pada *state* tertentu diharapkan dapat menjadi kunci dalam memecahkan kedua masalah tersebut. Dengan demikian, penelitian ini bertujuan untuk mengembangkan sebuah metode baru dalam menjawab permasalahan mekanik permainan yang monoton dengan mempertimbangkan performa *game* yang dibuat.

## STUDI LITERATUR

Perkembangan industri *game* menjadikan pengalaman pemain, khususnya pengalaman imersif, sebagai salah satu indikator utama keberhasilan suatu *game*. Pengalaman imersif didefinisikan sebagai kondisi ketika pemain merasa terlibat secara mendalam dalam lingkungan *virtual* sehingga mampu mengalihkan perhatian dari dunia nyata (Gao et al., 2022; Cutting et al., 2025). Salah satu faktor yang memengaruhi pengalaman tersebut adalah perilaku *Non-Player Character* (NPC), karena animasi maupun respons yang kaku dapat membuat permainan terasa monoton dan mudah ditebak (Setiawan et al., 2021; Al Kautsar et al., 2024). Untuk mengatasi permasalahan tersebut, banyak penelitian menerapkan *Finite State Machine* (FSM) sebagai metode pengendalian perilaku NPC. FSM bekerja melalui tiga komponen utama, yaitu *state*, *event*, dan *action*, sehingga karakter dapat berpindah perilaku sesuai kondisi tertentu (Herlambang, 2019, dalam Setiawan et al., 2021; Rosuli et al., 2025). Penelitian Pranselga et al. (2021) dan Abdurrohman et al. (2023) menunjukkan bahwa FSM efektif mengendalikan perilaku NPC melalui *state* seperti *idle*, *chasing*, *attacking*, dan *death*. Meskipun demikian, penggunaan batas numerik yang bersifat tegas menyebabkan respons NPC kurang adaptif dan cenderung berulang sehingga berpotensi mengurangi pengalaman imersif pemain (Setiawan et al., 2021).

Untuk mengatasi keterbatasan tersebut, berbagai penelitian mengintegrasikan logika *fuzzy* ke dalam sistem pengambilan keputusan NPC. Berbeda dengan logika klasik yang hanya mengenal kondisi benar atau salah, logika *fuzzy* memungkinkan suatu nilai memiliki derajat keanggotaan antara 0 hingga 1 melalui proses *fuzzifikasi*, evaluasi aturan, implikasi, agregasi, dan *defuzzifikasi* (Sutojo et al., 2011, dalam Setiawan et al., 2021; Naba dalam Setiawan et al., 2021). Pendekatan ini memungkinkan NPC menghasilkan respons yang lebih fleksibel berdasarkan kombinasi beberapa variabel, seperti jarak dan kecepatan pemain (Hakiki dalam Setiawan et al., 2021). Dalam penerapannya, Rizal et al. (2022) memanfaatkan metode Sugeno untuk menentukan tingkat kesulitan permainan secara adaptif, sedangkan Setiawan et al. (2021) mengintegrasikan *Fuzzy Tsukamoto* dengan FSM untuk menghasilkan perilaku NPC yang lebih alami. Penelitian lainnya milik Amrulloh et al. (2025) juga menerapkan logika *fuzzy* metode Sugeno dalam perilaku NPC berdasarkan kondisi permainan, seperti jarak pemain, Kesehatan NPC, dan *damage* yang diterima. Dibandingkan metode Sugeno, Tsukamoto dinilai lebih sesuai dalam memodelkan perilaku NPC karena menggunakan fungsi keanggotaan monoton yang menghasilkan keputusan secara bertahap (Tundo & Mahardika, 2023). Temuan tersebut diperkuat oleh Fuad et al. (2023) yang menyatakan bahwa metode Tsukamoto memiliki fleksibilitas tinggi dalam menangani kondisi yang mengandung ketidakpastian.

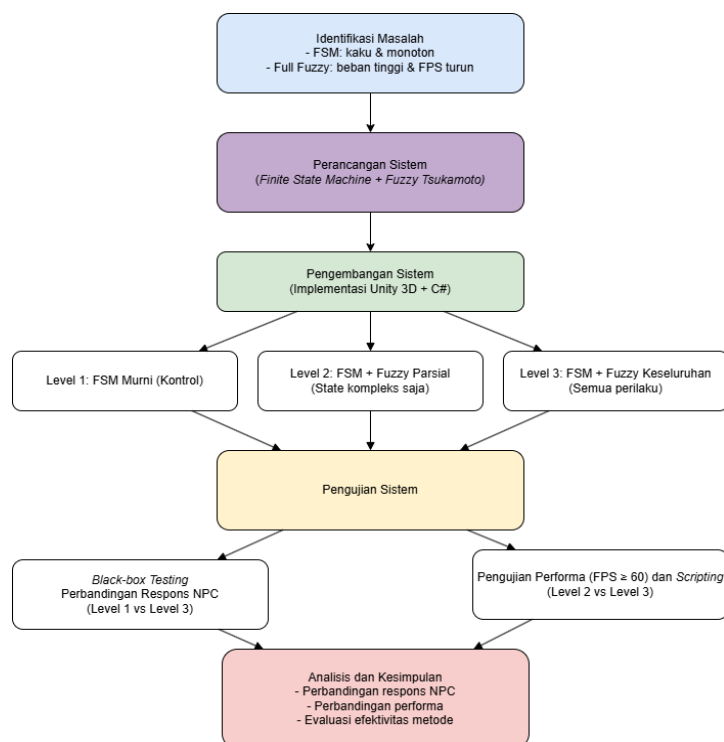
Di sisi lain, peningkatan kompleksitas logika juga berpotensi memengaruhi performa permainan. Riwinoto dan Tan (2021) menyatakan bahwa proses *rendering* dan *scripting* merupakan penyebab utama penurunan performa *game*, sedangkan penambahan logika *fuzzy* dapat meningkatkan beban komputasi sehingga memengaruhi stabilitas sistem. Fedotova et al. (2023) juga menjelaskan bahwa kompleksitas algoritma berkaitan langsung dengan kebutuhan optimasi CPU maupun GPU. Oleh karena itu, optimasi *scripting* menjadi aspek

penting dalam pengembangan *game* modern (Koulaxidis & Xinogalos, 2022). Salah satu indikator yang umum digunakan untuk mengevaluasi performa tersebut adalah *Frame Per Second* (FPS) karena nilai FPS merepresentasikan kelancaran permainan yang dirasakan pemain (Fedotova et al., 2023). Dengan demikian, pengembangan perilaku NPC tidak hanya perlu mempertimbangkan kualitas pengambilan keputusan, tetapi juga efisiensi komputasi agar pengalaman bermain tetap optimal.

Berdasarkan berbagai penelitian terdahulu, FSM merupakan metode yang efektif untuk mengendalikan struktur perilaku NPC, sedangkan logika *fuzzy* mampu meningkatkan fleksibilitas pengambilan keputusan karakter melalui mekanisme inferensi yang adaptif. Penelitian Setiawan et al. (2021) telah menunjukkan bahwa FSM dan *Fuzzy* Tsukamoto dapat diintegrasikan untuk menghasilkan perilaku NPC yang lebih alami, sementara Amrulloh et al. (2025) menunjukkan penerapan *Fuzzy* Sugeno untuk menghasilkan respons NPC berdasarkan beberapa kondisi permainan. Namun, penelitian-penelitian tersebut belum secara khusus membahas pembagian fungsi antara FSM sebagai pengelola *state* perilaku dan *Fuzzy* Tsukamoto sebagai mekanisme pengambilan keputusan adaptif yang diterapkan secara selektif pada *state* tertentu. Selain itu, penelitian terkait performa *game* lebih banyak membahas optimasi *rendering* dan algoritma secara umum (Riwinoto & Tan, 2021; Fedotova et al., 2023; Koulaxidis & Xinogalos, 2022). Oleh karena itu, penelitian ini mengusulkan integrasi *Fuzzy* Tsukamoto secara parsial pada *state* yang memerlukan pengambilan keputusan adaptif, dengan FSM tetap digunakan untuk mengelola struktur perilaku NPC, sehingga dapat dikaji keseimbangan antara adaptivitas perilaku NPC dan performa permainan.

## METODE

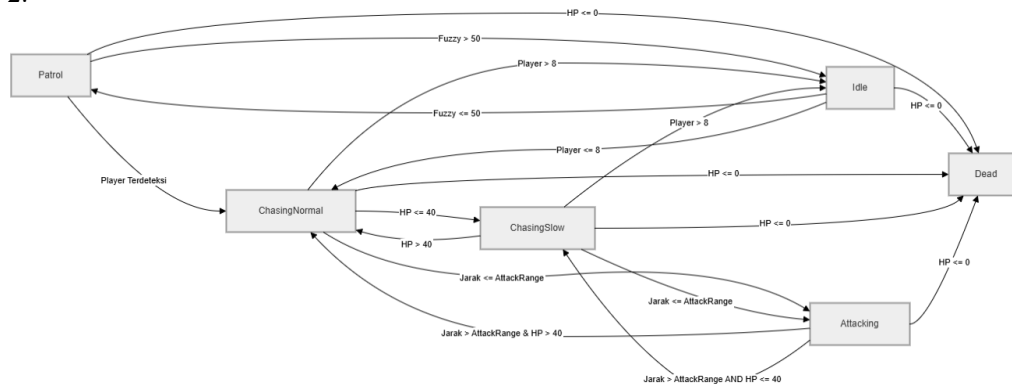
Penelitian ini menggunakan model *Rapid Application Development* (RAD) karena menekankan siklus pengembangan yang singkat melalui pendekatan iteratif dan pembuatan prototipe sehingga sesuai untuk eksperimen teknis dalam mengembangkan perilaku NPC (Pratama et al., 2023). Metode ini memungkinkan pengoptimalan algoritma tanpa harus menyelesaikan keseluruhan sistem permainan terlebih dahulu serta mendukung evaluasi tiga tingkat implementasi NPC yang dikembangkan. Tahapan RAD yang diterapkan meliputi identifikasi kebutuhan pengguna, perancangan sistem, pengembangan sistem, dan implementasi produk (Virtalioka et al., 2024). Pada tahap identifikasi kebutuhan, dilakukan analisis terhadap fenomena kejenuhan pemain akibat perilaku NPC yang monoton sekaligus mengidentifikasi keterbatasan integrasi logika yang kompleks terhadap performa permainan (Riwinoto & Tan, 2021). Selanjutnya, sistem dirancang dan dikembangkan secara iteratif untuk menghasilkan perilaku NPC yang lebih adaptif melalui integrasi *Finite State Machine* dan *Fuzzy* Tsukamoto, kemudian diimplementasikan serta diuji berdasarkan aspek fungsionalitas dan performa. Perancangan sistem juga mempertimbangkan efisiensi *scripting* agar tidak membebani CPU maupun GPU secara berlebihan sehingga stabilitas permainan tetap terjaga di atas standar 60 FPS (Koulaxidis & Xinogalos, 2022).



Gambar 1 Bagan Alur Penelitian

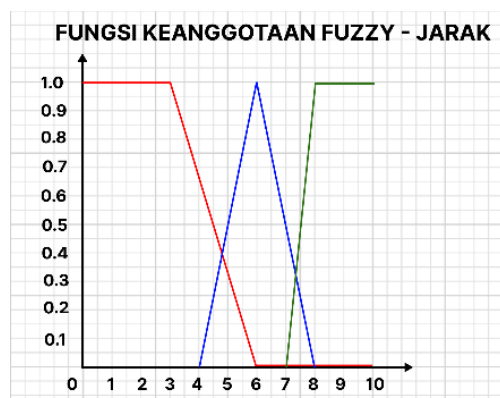
Gambar 1 menunjukkan alur penelitian yang terdiri atas tahap identifikasi masalah, perancangan sistem, pengembangan sistem, pengujian, serta analisis dan penarikan kesimpulan. Pada tahap pengembangan, sistem dibagi menjadi tiga tingkat implementasi untuk membandingkan FSM murni, integrasi FSM dengan *Fuzzy Tsukamoto* secara keseluruhan, dan integrasi *Fuzzy Tsukamoto* secara parsial. Selanjutnya, pengujian dilakukan melalui *Black-box Testing* untuk mengevaluasi respons NPC serta pengujian performa menggunakan FPS dan waktu *scripting* untuk membandingkan implementasi *Full Fuzzy* dan *Fuzzy Parsial*.

Selanjutnya, perancangan NPC dimulai dengan pembuatan diagram FSM untuk mendefinisikan *state* dasar NPC seperti *chase*, *battle*, dan *death* berdasarkan aturan numerik FSM. Aturan-aturan tersebut dibuat berdasarkan sumber-sumber yang telah digunakan, kemudian disesuaikan kembali dengan kebutuhan yang diperlukan. Adapun diagram dibuat untuk memudahkan pada tahap pengimplementasian di *Unity* nantinya dan dapat dilihat pada Gambar 2.



Gambar 2 Diagram *Finite State Machine*.

Seperti yang telah tertera dalam diagram tersebut, terdapat dua *state* yang difokuskan untuk diatur oleh *fuzzy logic*, yakni *patrol* dan *standby/idle*. Adapun *fuzzy logic* dalam sistem yang dirancang dipengaruhi oleh 2 buah variabel, yakni jarak dan kecepatan. Kedua nilai masukan ini nantinya akan dinormalisasi berdasarkan aturan *fuzzy* yang telah dirancang. Adapun fungsi-fungsi keanggotaan tersebut dapat dilihat sebagai berikut.



Fungsi Jarak

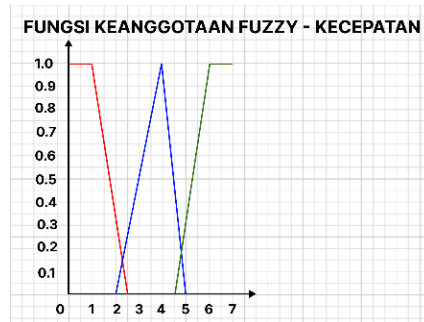
Gambar 3 Fungsi Keanggotaan Jarak

Fungsi jarak adalah fungsi yang digunakan untuk menentukan aksi yang dilakukan oleh musuh. Fungsi ini sendiri dibedakan menjadi 3 jenis, yakni dekat, sedang dan jauh. Adapun berdasarkan diagram yang telah dipaparkan, maka dapat disimpulkan himpunan linguistiknya adalah pada Tabel 1.

Tabel 1 Himpunan Linguistik Fungsi Jarak.

| Fungsi | Domain | Parameter   |
|--------|--------|-------------|
| Dekat  | 0-6    | [0;0;3;6]   |
| Sedang | 4-8    | [4;6;8]     |
| Jauh   | 7-10   | [7;8;10;10] |

### Fungsi Kecepatan



Gambar 4 Fungsi Keanggotaan Kecepatan

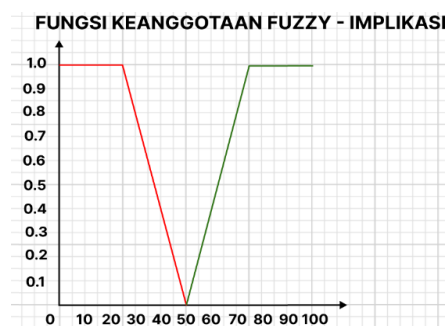
Fungsi kecepatan adalah fungsi yang digunakan untuk menentukan kecepatan aksi yang dilakukan oleh musuh. Fungsi ini sendiri dibedakan menjadi 3 jenis, yakni lambat, sedang, dan cepat. Adapun berdasarkan diagram yang telah dipaparkan, maka dapat disimpulkan himpunan linguistiknya adalah pada Tabel 2.

Tabel 2 Himpunan Linguistik Fungsi Kecepatan

| Fungsi | Domain | Parameter   |
|--------|--------|-------------|
| Lambat | 0-2,5  | [0;0;1;2,5] |
| Sedang | 2-5    | [2;4;5]     |
| Cepat  | 4,5-7  | [4,5;6;7;7] |

### Fungsi Implikasi

Berdasarkan logika *fuzzy* yang digunakan, maka didapat *rule based* ataupun aturan utama yang akan digunakan dari 2 parameter tersebut dengan dibagi menjadi 2 bagian, yakni *patrol* dan *idle/standby*. Kemudian untuk fungsi implikasi yang digunakan ialah fungsi AND (fungsi MIN). Adapun derajat keanggotaan dari masing-masing parameter tersebut nantinya akan dihitung, kemudian diproses melalui proses *defuzzifikasi* menggunakan metode Tsukamoto dengan diagram berikut.



Gambar 5 Fungsi Keanggotaan Implikasi

Berdasarkan Gambar 5, maka dapat dirumuskan *rule based* yang akan dipakai untuk diterapkan dalam pengujian algoritma ini, yakni dapat dilihat pada Tabel 3.

Tabel 3 Rule Based Fuzzy Logic

| Syarat   |        |          |           | Hasil    |          |
|----------|--------|----------|-----------|----------|----------|
| Operator | Jarak  | Operator | Kecepatan | Operator | Tindakan |
| IF       | Dekat  | AND      | Pelan     | THEN     | Standby  |
|          | Dekat  |          | Sedang    |          | Standby  |
|          | Dekat  |          | Cepat     |          | Standby  |
|          | Sedang |          | Pelan     |          | Patrol   |
|          | Sedang |          | Sedang    |          | Patrol   |
|          | Sedang |          | Cepat     |          | Standby  |
|          | Jauh   |          | Pelan     |          | Patrol   |
|          | Jauh   |          | Sedang    |          | Patrol   |
|          | Jauh   |          | Cepat     |          | Patrol   |

Implementasi logika yang telah dipaparkan sebelumnya akan dikembangkan menggunakan *Game Engine Unity 3D* dengan bahasa pemrograman C#. Pengujian fungsionalitas sistem sekaligus membandingkan variasi respons dalam kondisi tertentu untuk NPC yang hanya diterapkan model FSM dan *fuzzy* secara parsial akan diuji melalui *Black-box Testing*. Adapun kondisi-kondisi yang akan diuji juga diambil berdasarkan sumber-sumber yang telah digunakan dan disesuaikan kembali dengan kebutuhan penelitian. Selain uji fungsionalitas, dilakukan juga pemantauan teknis terhadap stabilitas *Frame Rate Per Second* (FPS) dan waktu eksekusi *scripting* selama 60 detik pada NPC yang diterapkan *fuzzy logic* pada 3 state (*patrol*, *standby* dan *battle*) dan *fuzzy logic* pada 2 state saja atau yang disebut sebagai *fuzzy* parsial (*patrol* dan *standby*). Aspek ini diuji guna mengukur beban komputasi perangkat ketika menjalankan tahap *scripting* yang cukup kompleks tersebut.

## HASIL

Bagian ini berfokus pada pemaparan hasil dari *game* yang telah dirancang, pengujian fungsionalitas serta pengujian stabilitas performa menggunakan parameter *Frame per Second* (FPS). *Game* dirancang secara sederhana karena penelitian ini lebih difokuskan pada pengujian logika dan model yang diterapkan dibandingkan aspek visual. Adapun tampilan *game* tersebut dapat dilihat pada Gambar 6.



Gambar 6 Tampilan Antarmuka *Game*

Gambar 6 menunjukkan lingkungan permainan yang digunakan sebagai media implementasi dan pengujian perilaku NPC. Pada lingkungan tersebut, NPC ditempatkan sebagai karakter yang dapat merespons keberadaan pemain melalui perubahan perilaku. Implementasi menghasilkan tiga kondisi pengujian, yaitu NPC dengan FSM tanpa campur tangan *fuzzy* pada level 1, NPC dengan penerapan *fuzzy* pada seluruh states pada level 2, dan NPC dengan penerapan *fuzzy* pada state *patrol* dan *standby* pada level 3. Perbedaan implementasi tersebut digunakan untuk mengamati perubahan respons NPC sekaligus membandingkan beban pemroses yang dihasilkan.

Sebagaimana yang telah dipaparkan pada bagian sebelumnya bahwa *game* ini terdiri dari 3 tahap dengan perlakuan yang berbeda pada masing-masing tahapan. Oleh karena itu dilakukan uji fungsionalitas pada tahap 1 dan tahap 3 dari *game* yang telah dirancang untuk melihat perbedaan respons NPC ketika diberikan perlakuan tertentu. Tabel 4 menunjukkan hasil pengujian *Black-box* pada tahap 1, yaitu ketika NPC hanya menerapkan metode FSM tanpa integrasi *fuzzy logic*.

Tabel 4 *Black-box Testing Level 1*

| No. | Kondisi Input (Aksi Pemain)   | Output yang Diharapkan (Respon)              | Hasil                                                                                                                                                                                                          |
|-----|-------------------------------|----------------------------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 1.  | Jarak <i>player</i> > 8 unit  | NPC tetap diam karena di luar radius deteksi | [10:50:37] [TEST 1] NPC tetap diam/patrol. Jarak player = 9.40<br>UnityEngine.Debug.Log (object)                                                                                                               |
| 2.  | Jarak <i>player</i> <= 8 unit | NPC mulai mengejar pemain secara konstan     | [10:50:47] [TEST 2] Player terdeteksi. NPC mulai mengejar. Jarak = 7.97<br>UnityEngine.Debug.Log (object)<br>[10:50:47] [TEST 2] State berubah menjadi CHASING. HP NPC = 120<br>UnityEngine.Debug.Log (object) |
| 3.  | Jarak <= Jangkauan serang     | NPC melakukan aksi serangan                  | [10:50:51] [TEST 3] NPC masuk state ATTACKING. Jarak = 1.99<br>UnityEngine.Debug.Log (object)<br>[10:51:14] [TEST 3] NPC menyerang player sebesar 10 damage<br>UnityEngine.Debug.Log (object)                  |
| 4.  | Hp NPC <= 0                   | NPC masuk ke state <i>Death</i>              | [10:51:14] [TEST 4] HP NPC <= 0. NPC masuk state DEAD<br>UnityEngine.Debug.Log (object)                                                                                                                        |

Pada level 1, NPC menggunakan FSM murni sehingga perubahan perilaku ditentukan oleh kondisi yang telah ditetapkan. Ketika jarak pemain berada di luar jangkauan deteksi, NPC tetap berada pada kondisi diam.

Ketika pemain memasuki jangkauan deteksi, NPC berpindah menuju kondisi mengejar, kemudian melakukan serangan ketika pemain berada dalam jangkauan serang. Hasil ini menunjukkan bahwa FSM mampu menghasilkan alur perilaku yang terstruktur sesuai dengan kondisi yang telah dirancang. Selanjutnya, pengujian fungsionalitas pada tahap 3 dalam *game* yang telah dirancang untuk melihat perbedaan respons ketika sudah diintegrasikan *fuzzy logic* khususnya pada *state patrol* dan *standby*. Adapun hasil dari *Black-box testing* yang telah dilakukan dapat dilihat pada Tabel 5.

Tabel 5 *Black-box Testing Level 3*

| No | Variabel Input |           | Aturan yang Diwakili | Output yang Diharapkan | Hasil Uji                                                                                                                            |
|----|----------------|-----------|----------------------|------------------------|--------------------------------------------------------------------------------------------------------------------------------------|
| .  | Jarak          | Kecepatan |                      |                        |                                                                                                                                      |
| 1. | Dekat          | Pelan     | 1                    | <i>Standby</i>         | [11:31:08] [FUZZY TEST Rule 1   Jarak = 2.13   Kecepatan = 0.00   Output = 75.00   State = STANDBY<br>UnityEngine.Debug.Log (object) |
| 2. | Dekat          | Sedang    | 2                    | <i>Standby</i>         | [11:31:08] [FUZZY TEST Rule 2   Jarak = 1.68   Kecepatan = 4.23   Output = 75.00   State = STANDBY<br>UnityEngine.Debug.Log (object) |
| 3. | Dekat          | Cepat     | 3                    | <i>Standby</i>         | [11:31:08] [FUZZY TEST Rule 3   Jarak = 1.48   Kecepatan = 6.06   Output = 75.00   State = STANDBY<br>UnityEngine.Debug.Log (object) |
| 4. | Sedang         | Pelan     | 4                    | <i>Patrol</i>          | [11:12:33] [FUZZY TEST Rule 4   Jarak = 7.23   Kecepatan = 0.00   Output = 25.00   State = PATROL<br>UnityEngine.Debug.Log (object)  |
| 5. | Sedang         | Sedang    | 5                    | <i>Patrol</i>          | [11:12:22] [FUZZY TEST Rule 5   Jarak = 7.28   Kecepatan = 4.03   Output = 25.00   State = PATROL<br>UnityEngine.Debug.Log (object)  |
| 6. | Sedang         | Cepat     | 6                    | <i>Standby</i>         | [11:11:50] [FUZZY TEST Rule 6   Jarak = 7.28   Kecepatan = 5.64   Output = 52.97   State = STANDBY<br>UnityEngine.Debug.Log (object) |
| 7. | Jauh           | Pelan     | 7                    | <i>Patrol</i>          | [11:11:38] [FUZZY TEST Rule 7   Jarak = 9.40   Kecepatan = 0.20   Output = 25.00   State = PATROL<br>UnityEngine.Debug.Log (object)  |
| 8. | Jauh           | Sedang    | 8                    | <i>Patrol</i>          | [11:11:39] [FUZZY TEST Rule 8   Jarak = 9.52   Kecepatan = 2.91   Output = 25.00   State = PATROL<br>UnityEngine.Debug.Log (object)  |
| 9. | Jauh           | Cepat     | 9                    | <i>Patrol</i>          | [11:11:50] [FUZZY TEST Rule 9   Jarak = 8.90   Kecepatan = 5.63   Output = 25.00   State = PATROL<br>UnityEngine.Debug.Log (object)  |

Berbeda dengan *level 1*, hasil pengujian pada *level 3* menunjukkan bahwa keputusan NPC mempertimbangkan kombinasi jarak dan kecepatan pemain. Hal tersebut terlihat pada kondisi jarak sedang, ketika kecepatan pemain pelan dan sedang menghasilkan kondisi *patrol*, sedangkan ketika kecepatan pemain meningkat menjadi cepat, NPC berpindah ke kondisi *standby*. Perubahan keluaran tersebut menunjukkan bahwa kombinasi variabel masukan dapat menghasilkan respons NPC yang berbeda meskipun salah satu variabel berada pada kategori yang sama. Hasil tersebut juga menunjukkan karakteristik penerapan *Fuzzy Logic* Tsukamoto dalam penelitian ini, yaitu keputusan NPC diperoleh melalui kombinasi beberapa kondisi masukan. Dengan demikian, *Fuzzy Parsial* memberikan mekanisme pengambil keputusan yang mempertimbangkan hubungan antara jarak dan kecepatan pemain pada *state* yang telah ditentukan.

Selanjutnya, pengujian stabilitas performa pada tahap 2 dan tahap 3 guna membandingkan dan mengevaluasi efektivitas penerapan *fuzzy logic* baik secara *full* maupun parsial dalam *game* yang telah dirancang. Parameter utama yang digunakan dalam pengujian ini adalah FPS yang diperoleh melalui salah satu fitur pada *Game Engine Unity* itu sendiri, yakni *Unity Game Profiler*. Berikut merupakan ringkasan performa pada kedua tahapan yang telah diuji tersebut.

Tabel 6 FPS Rata-rata dan Waktu Eksekusi *Scripting*.

| Tingkatan Implementasi  | FPS Rata-rata | Waktu Scripting | Hasil Uji                                                                                                                                                                                                                                                   |
|-------------------------|---------------|-----------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Level 2 (Full Fuzzy)    | 78,11 FPS     | 2,374 ms        | <p>[11:58:26] [FPS TEST] Durasi = 60.01s   Total Frame = 4687   FPS Rata-rata = 78.11<br/>UnityEngine.Debug.Log (object)</p> <p>Systems impact across selection (mean time)</p> <p>Others (5.260ms)</p> <p>Rendering (4.730ms)</p> <p>Scripts (2.374ms)</p> |
| Level 3 (Fuzzy Parsial) | 79,58 FPS     | 2,334 ms        | <p>[11:54:30] [FPS TEST] Durasi = 60.01s   Total Frame = 4775   FPS Rata-rata = 79.58<br/>UnityEngine.Debug.Log (object)</p> <p>Systems impact across selection (mean time)</p> <p>Others (7.855ms)</p> <p>Rendering (5.228ms)</p> <p>Scripts (2.334ms)</p> |

Berdasarkan Tabel 6, implementasi *Fuzzy Parsial* menghasilkan rata-raya 79,58 FPS, sedangkan *Full Fuzzy* menghasilkan 78,11 FPS. Selisih sebesar 1,47 FPS menunjukkan bahwa penerapan *fuzzy* hanya pada dua *states* menghasilkan *frame rate* yang sedikit lebih tinggi dibandingkan penerapan pada seluruh *states*. Pola yang sama terlihat pada waktu eksekusi *scripting*, yaitu 2,334 ms pada *Fuzzy Parsial* dan 2,374 ms pada *Full Fuzzy*, dengan selisih 0,040 ms.

Perbedaan tersebut menunjukkan bahwa pengurangan cakupan penerapan *fuzzy* berkaitan dengan berkurangnya waktu eksekusi *scripting* pada kondisi pengujian. Namun, perbedaannya relative kecil dan kedua implementasi menghasilkan *frame rate* di atas 60 FPS. Dengan demikian, hasil pengujian performa menunjukkan adanya perbedaan beban pemrosesan antara *Full Fuzzy* dan *Fuzzy Parsial*, meskipun keduanya tetap dapat berjalan pada kondisi pengujian yang dilakukan.

## PEMBAHASAN

Berdasarkan hasil pengujian *Black-box* yang telah dilakukan, seluruh skenario pengujian berhasil menghasilkan keluaran sesuai dengan spesifikasi kebutuhan sistem. Pengujian fungsionalitas menggunakan metode *Black-box* mengonfirmasi bahwa seluruh sistem yang telah dirancang, khususnya fitur pengambilan keputusan NPC, telah berjalan sesuai dengan spesifikasi kebutuhan awal. Pada implementasi FSM, NPC mampu melakukan transisi state dari *Idle*, *Patrol*, *Chasing*, *Attacking*, hingga *Death* sesuai kondisi yang diberikan. Sementara itu, pada implementasi Logika *Fuzzy* Tsukamoto, seluruh aturan (*rule base*) yang dirancang berhasil menghasilkan keluaran yang sesuai dengan kondisi jarak dan kecepatan pemain.

Dukungan empiris ini tidak hanya mengonfirmasi bahwa teori *Finite State Machine* yang menyatakan bahwa perilaku NPC dikendalikan melalui perpindahan *state* berdasarkan kondisi tertentu, tetapi juga memperkuat temuan penelitian terdahulu yang menyebutkan bahwa FSM efektif digunakan untuk mengendalikan perilaku NPC dalam *game*. Konsisten dengan literatur terdahulu, fenomena ini membuktikan postulat bahwa FSM mampu menghasilkan perilaku yang terstruktur dan mudah dikendalikan melalui aturan transisi yang jelas.

Namun demikian, hasil pengujian juga menunjukkan bahwa penerapan logika *fuzzy* memberikan fleksibilitas tambahan dalam proses pengambilan keputusan NPC. Dukungan empiris ini juga mengonfirmasi bahwa teori Logika *Fuzzy* Tsukamoto yang menyatakan bahwa suatu keputusan dapat dipengaruhi oleh beberapa variabel sekaligus melalui proses *fuzzifikasi*, inferensi, dan *defuzzifikasi*. Jika ditelaah lebih jauh, korelasi antara variabel jarak dan kecepatan pemain terhadap keputusan NPC terjadi karena mekanisme inferensi *fuzzy* memungkinkan suatu kondisi memiliki derajat keanggotaan pada lebih dari satu himpunan *fuzzy* secara bersamaan. Rasionalisasi di balik temuan ini adalah bahwa mekanisme tersebut memungkinkan keputusan NPC mempertimbangkan beberapa kondisi masukan secara bersamaan, sehingga menghasilkan variasi respons yang lebih beragam dibandingkan kondisi berbasis batas tunggal pada FSM.

Selain pengujian fungsionalitas, dilakukan pula pengujian performa untuk mengetahui pengaruh implementasi *fuzzy* terhadap kinerja permainan. Hasil pengujian selama 60 detik menunjukkan bahwa implementasi *Full Fuzzy* menghasilkan rata-rata *frame rate* sebesar 78,11 FPS dengan waktu *scripting* sebesar 2,374 ms, sedangkan implementasi *Fuzzy Parsial* menghasilkan rata-rata *frame rate* sebesar 79,58 FPS dengan waktu *scripting* sebesar 2,334 ms. Analisis perbandingan kinerja menunjukkan bahwa penerapan *Full Fuzzy* hanya meningkatkan beban *scripting* sebesar 0,040 ms dibandingkan *Fuzzy Parsial*. Selisih tersebut relatif kecil dan tidak menunjukkan perbedaan performa yang besar pada kondisi pengujian karena kedua metode masih mampu mempertahankan nilai FPS di atas standar 60 FPS.

Temuan ini mengonfirmasi teori bahwa penambahan proses komputasi pada logika *fuzzy* akan meningkatkan kebutuhan pemrosesan sistem. Akan tetapi, peningkatan tersebut berada pada tingkat yang sangat rendah sehingga tidak menyebabkan perubahan performa yang besar, terutama pada tingkat penerapan logika *fuzzy* pada beberapa *state* saja. Jika ditelaah lebih jauh, perbedaan performa yang kecil tersebut terjadi karena proses *fuzzy* yang diterapkan hanya terdiri atas *fuzzifikasi*, evaluasi sembilan aturan, dan *defuzzifikasi* sederhana. Dengan demikian, beban komputasi yang ditambahkan masih jauh lebih kecil dibandingkan proses lain seperti *rendering*, animasi, dan navigasi NPC yang juga berjalan selama permainan berlangsung.

Berangkat dari analisis di atas, desain sistem yang menggabungkan FSM dan Logika *Fuzzy* Tsukamoto menawarkan pendekatan yang mampu menghasilkan perilaku NPC yang lebih adaptif tanpa mengorbankan performa permainan secara signifikan. Secara teoritis, penelitian ini memperkaya literatur pada bidang pemodelan perilaku NPC untuk *game* dengan membuktikan secara empiris bahwa integrasi *fuzzy* ke dalam mekanisme FSM dapat meningkatkan fleksibilitas pengambilan keputusan NPC dengan tambahan beban komputasi yang relatif rendah.

## KESIMPULAN

Berdasarkan hasil penelitian, penerapan metode yang dikembangkan berhasil menghasilkan perilaku NPC yang lebih adaptif dibandingkan penggunaan FSM murni melalui pengambilan keputusan yang mempertimbangkan kombinasi jarak dan kecepatan pemain. Pengujian *Black-box* menunjukkan bahwa seluruh fitur dan aturan yang dirancang telah berjalan sesuai dengan kebutuhan sistem, sedangkan pengujian performa

menunjukkan bahwa penerapan *Fuzzy* secara parsial maupun *Full Fuzzy* menghasilkan perbedaan beban komputasi yang relatif kecil dan tetap mempertahankan performa permainan di atas standar minimum. Hasil tersebut menunjukkan bahwa penerapan *Fuzzy* pada *state* tertentu dapat meningkatkan fleksibilitas perilaku NPC tanpa memberikan dampak signifikan terhadap performa permainan. Namun, penelitian ini masih terbatas pada penggunaan dua variabel masukan dan aturan *fuzzy* yang relatif sederhana. Penelitian selanjutnya dapat menambahkan variabel masukan, meningkatkan kompleksitas lingkungan dan jumlah NPC, serta membandingkan pendekatan yang digunakan dengan metode lain untuk memperoleh hasil yang lebih komprehensif.

#### REFERENSI

- Abdurrohman, D., Pranoto, Y. A., & Prasetya, R. P. (2023). Perancangan game Lost Forest 3D menggunakan metode finite state machine berbasis desktop. *JATI (Jurnal Mahasiswa Teknik Informatika)*, 7(4), 2280–2287.
- Al Kautsar, M. F., Haryanto, H., Astuti, E. Z., Mulyanto, E., Rosyidah, U., & Kardianawati, A. (2024). Penerapan finite state machine dalam perancangan perilaku musuh pada game 2D platformer bertema sejarah. *TeknoIS: Jurnal Ilmiah Teknologi Informasi dan Sains*, 14(2), 236–243.
- Amrulloh, E. S., Krisdiawan, R. A., Lesmana, I., & Rohmawati, L. (2025). Fuzzy-driven adaptive NPC behavior in a meme-based platformer game for Android mobile: Perilaku NPC adaptif berbasis fuzzy dalam game meme platformer berbasis ponsel Android. *Nuansa Informatika*, 19(2), 123–132. <https://doi.org/10.25134/ilkom.v19i2.440>
- Cutting, J., Klassen, R. M., Saiger, M., Thompson-Lee, S., & Snell, R. J. S. (2025). Exploring player experience factors for designing persuasive recruitment games. *Proceedings of the ACM on Human-Computer Interaction*, 9(GAMES022), 1–31. <https://doi.org/10.1145/3748617>
- Fedotova, N., Protsenko, M., Baranova, I., Vashchenko, S., & Dehtiarenko, Y. (2023). Research on calculation optimization methods used in computer games development. *Informatyka, Automatyka, Pomiar w Gospodarce i Ochronie Środowiska*, 13(3), 37–42. <https://doi.org/10.35784/iapgos.3828>
- Fuad, M., Wattimena, F. Y., Rizani, A., & Yuswardi. (2023). Investment decision making in digital business using Tsukamoto fuzzy logic. *International Journal Software Engineering and Computer Science (IJSECS)*, 3(2), 144–150. <https://doi.org/10.35870/ijsecs.v3i2.1525>
- Gao, Y., Wu, D., & Zhou, L. (2024). How to improve immersive experience? *IEEE Transactions on Multimedia*, 26, 8691–8703. <https://doi.org/10.1109/TMM.2022.3199666>
- Koulaxidis, G., & Xinogalos, S. (2022). Improving mobile game performance with basic optimization techniques in Unity. *Modelling*, 3(2), 201–223. <https://doi.org/10.3390/modelling3020014>
- Pranselga, A., Setiawan, I. R., & Apriandari, W. (2021). Implementasi finite state machine pada karakter NPC musuh dalam game adventure in Java. *JUTISI (Jurnal Ilmiah Teknik Informatika dan Sistem Informasi)*, 10(3), 391–404.
- Pratama, E. A., Krisgianti, S., & Paramita, H. M. (2023). Penerapan metode rapid application development (RAD) pada pengembangan sistem informasi inventaris aset Desa Rempoah. *Infotek: Jurnal Informatika dan Teknologi*, 6(1), 49–59.
- Rizal, A., Astuti, Y. P., & Setiawan, A. (2022). Tingkat kesulitan adaptif pada Android game bertema flora fauna endemik Indonesia dengan fuzzy logic. *Jurnal Buana Informatika*, 13(1), 1–10.
- Riwinoto, & Tan, W. (2021). Analisis performa prototype game pada platform Android. *JAMN (Journal of Applied Multimedia and Networking)*, 5(2), 102–109.
- Setiawan, D. A., Wibowo, S. A., & Pranoto, Y. A. (2021). Penerapan metode finite state machine dan fuzzy pada game "Black Warrior". *JATI (Jurnal Mahasiswa Teknik Informatika)*, 5(2), 708–719.
- Tundo, T., & Mahardika, F. (2023). Fuzzy inference system Tsukamoto–Decision Tree C4.5 in predicting the amount of roof tile production in Kebumen. *Jurnal Teori dan Aplikasi Matematika*, 7(2), 13034. <https://doi.org/10.31764/jtam.v7i2.13034>
- Virtalioka, S., Vernanda, D., & Nulpazri, A. R. (2024). Perancangan sistem informasi aplikasi rental mobil dengan metode rapid application development berbasis mobile. *Jurnal Komputer Bisnis Teknologi Informasi (JKBTI)*, 3(3), 90–97.