PENERAPAN ALGORITMA FINITE STATE MACHINE DALAM SISTEM KECERDASAN MUSUH PADA GAME AKSI TIGA DIMENSI BERBASIS UNITY

Dimas Satria Pamenang, Danang Wahyu Widodo, Risky Aswi Ramadhani
Teknik Informatika, Universitas Nusantara PGRI Kediri
Jl. KH Achmad Dahlan 76, Kota Kediri, Jawa Timur, Indonesia
Pamenang.dimas@gmail.com

ABSTRAK

Perkembangan industri game aksi tiga dimensi menuntut adanya kecerdasan buatan (AI) musuh yang adaptif dan responsif terhadap situasi permainan untuk menjaga tantangan dan keterlibatan pemain. Namun, banyak game masih menerapkan AI musuh yang monoton dan mudah diprediksi, sehingga mengurangi dinamika permainan. Penelitian ini bertujuan untuk merancang dan mengimplementasikan sistem AI musuh berbasis algoritma Finite State Machine (FSM) pada game Code Crusader: Anti-Virus Assault. FSM digunakan untuk mengatur perilaku musuh melalui lima state utama, yaitu Idle, Patrol, Chase, Attack, dan Dead. Implementasi dilakukan menggunakan bahasa pemrograman C# dalam lingkungan pengembangan game Unity. Pengujian sistem dilakukan melalui black-box testing terhadap seluruh transisi state serta user playtesting oleh 10 responden menggunakan kuesioner skala Likert. Hasil pengujian menunjukkan seluruh transisi perilaku AI berjalan sesuai rancangan tanpa kendala, dengan rata-rata skor uji pengguna mencapai 90% dalam kategori "sangat baik". Hasil ini membuktikan bahwa algoritma FSM efektif digunakan untuk menghasilkan AI musuh yang adaptif, responsif, dan mampu meningkatkan kualitas tantangan serta pengalaman bermain dalam game aksi tiga dimensi.

Kata kunci : *Game 3D, Unity, Artificial Intelligence, Finite State Machine, NPC, Game Development*

1. PENDAHULUAN

Dalam pengembangan game aksi tiga dimensi, kecerdasan buatan (AI) memegang peran penting dalam menentukan kualitas interaksi antara pemain dan karakter non-pemain (NPC). AI yang adaptif dan responsif terhadap situasi permainan terbukti mampu meningkatkan tantangan. AI dinamis membuat musuh lebih sulit dikalahkan dan memotivasi pemain untuk terus bermain [1]. Sayangnya, penerapan AI dalam banyak game masih belum maksimal, sehingga musuh sering kali menunjukkan perilaku yang seragam dan mudah diprediksi. Tanpa logika perilaku yang terstruktur, musuh akan bereaksi secara kaku dan kurang adaptif terhadap perubahan situasi permainan [2].

Untuk mengatasi permasalahan tersebut, salah satu pendekatan yang umum digunakan adalah Finite State Machine (FSM), yang memungkinkan pengaturan perilaku NPC secara sistematis melalui transisi antar state. Penggunaan FSM mampu meningkatkan responsivitas NPC sehingga musuh dapat mengejar pemain dari berbagai posisi dan memberikan interaksi yang lebih hidup [3]. Hal ini sejalan dengan pandangan bahwa FSM diperlukan untuk membangun logika perilaku musuh yang terstruktur dan fleksibel dalam menghadapi dinamika permainan [4].

Berdasarkan latar belakang tersebut, penelitian ini berfokus pada pengembangan sistem AI musuh berbasis FSM dalam game Code Crusader: Anti-Virus Assault, sebuah game aksi tiga dimensi berbasis Unity. Sistem ini dirancang dengan lima state utama, yaitu Idle, Patrol, Chase, Attack, dan Dead. Transisi antar state diatur berdasarkan kejadian dalam permainan, seperti deteksi pemain melalui radius dan sudut pandang, jarak serang, serta kondisi *health* musuh. Implementasi dilakukan menggunakan bahasa pemrograman C# pada Unity dengan bantuan NavMeshAgent untuk navigasi. Melalui pendekatan ini, penelitian ini menawarkan solusi terhadap keterbatasan AI musuh dalam game aksi tiga dimensi. FSM yang dirancang tidak hanya mengatur perilaku musuh secara sistematis, tetapi juga memungkinkan perpindahan antar state yang lancar dan logis sesuai aksi pemain.

Dengan demikian, penerapan FSM secara tepat diharapkan mampu menciptakan AI musuh yang adaptif, responsif, dan menantang. Peningkatan kualitas perilaku musuh ini juga diharapkan dapat meningkatkan pengalaman bermain yang lebih dinamis pada game Code Crusader: Anti-Virus Assault.

2. TINJAUAN PUSTAKA

2.1. Penelitian Terdahulu

Beberapa penelitian telah membahas penerapan Finite State Machine (FSM) dalam kecerdasan buatan untuk karakter musuh dalam game. Artikel "Sistem Perlawanan Musuh Dengan Artificial Intelligence (AI) Finite State Machine (FSM) Pada Game Petualangan" menjelaskan bahwa FSM dapat meningkatkan variasi perilaku musuh dan membuat pengalaman bermain menjadi lebih menantang [5]. Penelitian lain berjudul "Enhancing Gaming Engagement through the Integration of Game Design Document and Finite State Machine" membuktikan bahwa integrasi FSM dalam sistem desain karakter non-pemain mampu meningkatkan responsivitas dan keterlibatan pemain dalam game [3]. Sementara itu, "Penerapan Finite State Machine Dalam Sistem Pertarungan Role

Playing Game" menegaskan bahwa FSM sangat cocok digunakan untuk mengatur logika pertarungan musuh secara modular dan efisien [4].

Temuan-temuan tersebut mendukung penggunaan FSM sebagai pendekatan utama dalam pengembangan AI musuh untuk game aksi tiga dimensi yang menuntut respons cepat dan perilaku adaptif.

2.2. Game 3D

Game adalah sarana hiburan yang bersifat interaktif, di mana pemain terlibat secara aktif dalam berbagai aktivitas yang dirancang untuk menghadirkan kesenangan dan tantangan [6]. Game tiga dimensi (3D) menjadi salah satu jenis permainan digital yang menampilkan lingkungan serta objek dalam ruang tiga dimensi, sehingga mampu memberikan pengalaman visual yang lebih realistis dan mendalam bagi pemain. Selain itu, Game 3D tidak hanya berfungsi sebagai media hiburan, tetapi juga sebagai sarana edukasi karena menyajikan visualisasi ruang dan interaksi yang dapat meningkatkan minat serta pemahaman pengguna terhadap materi yang disampaikan [7]. Aspek visual dalam game 3D, baik melalui sudut pandang orang pertama maupun ketiga, mampu memperkuat keterlibatan emosional pemain, sehingga menciptakan pengalaman bermain yang terasa nyata [8].

2.3. Unity 3D

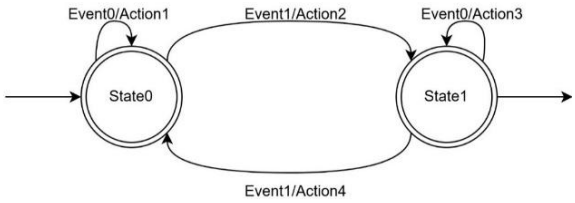
Unity 3D adalah engine game populer yang mendukung pengembangan game multiplatform dengan visual 3D serta scripting menggunakan C#. Unity 3D dapat mempercepat proses pembuatan game karena menyediakan platform yang mudah diakses, dokumentasi lengkap, dan komunitas besar, yang secara signifikan meningkatkan produktivitas pengembang hingga 30% [9].

Lebih lanjut, Unity 3D mendukung fitur real-time rendering, animasi, dan physics engine yang cukup robust, menjadikannya pilihan utama untuk game aksi 3D. Unity 3D memfasilitasi pengembangan sistem AI modular serta mempercepat siklus desain dan debugging, memungkinkan prototyping cepat dan lebih efisien [10].

2.4. Finite State Machine

Finite State Machine (FSM) adalah model komputasi yang merepresentasikan sistem melalui sejumlah kondisi terbatas (states) dengan transisi yang terjadi berdasarkan input atau kondisi tertentu. Model ini digunakan secara luas dalam pengembangan sistem kecerdasan buatan (AI) pada game, khususnya untuk mengatur perilaku karakter non-pemain (NPC) secara sistematis. FSM merupakan salah satu teknik tertua dan paling fundamental dalam desain AI game, yang telah digunakan sejak era Pac-Man hingga Tomb Raider, karena kesederhanaannya dan kemampuannya untuk menghasilkan perilaku NPC yang terstruktur namun terlihat cerdas [3]. Selain itu, FSM memberikan cara yang efisien dan terkontrol dalam mendefinisikan dan mengelola logika perilaku NPC, seperti merespons serangan, menjelajah area, atau menghindari musuh, melalui struktur kondisi yang saling berhubungan [11].

Dalam konteks game, FSM memungkinkan non-player character (NPC) berpindah antar kondisi seperti "patrol", "chase", dan "attack" sesuai kondisi lingkungan dan interaksi pemain. Berikut diagram sederhana FSM yang umum digunakan dalam game AI:



Gambar 1. Diagram finite state machine (fsm) pada perilaku npc musuh

Diagram ini menunjukkan transisi antar status NPC musuh dalam game berdasarkan kondisi lingkungan.

FSM dimodelkan sebagai 5-tuple:

$$FSM = (\mathrm{Q}, \mathrm{E}, \delta, \mathrm{q0}, \mathrm{F}) \quad (1)$$

a. Q: {Idle, Patrol, Chase, Attack, Dead}
b. E: {PlayerDetected, PlayerInRange, PlayerOutOfRange, NoWaypoint, HPZero, AnimationEnd}
c. δ\deltaδ: fungsi transisi Q×E→Q′Q \times E \to Q'Q×E→Q′
d. q0: Idle
e. FFF: {Dead}

$$\delta(\mathrm{Q}, \mathrm{E}) = \mathrm{Q}') \quad (2)$$

3. METODE PENELITIAN

3.1. Studi Literatur

Tahap awal dilakukan kajian literatur dari jurnal, artikel, dan dokumentasi teknis terkait penerapan algoritma Finite State Machine (FSM) dalam pengembangan AI musuh di game tiga dimensi. Literatur yang dikaji juga mencakup teknik navigasi AI menggunakan NavMeshAgent, serta metode deteksi pemain berbasis jarak dan sudut pandang dalam Unity.

3.2. Perancangan Sistem

AI musuh dirancang menggunakan algoritma FSM dengan lima state utama, yaitu Idle, Patrolling, Chasing, Attacking, dan Dead. Transisi antar state ditentukan berdasarkan kondisi permainan seperti deteksi pemain dalam radius tertentu, sudut pandang, jarak serang, serta status kesehatan musuh. Hubungan antar state divisualisasikan dalam diagram rancangan FSM yang ditunjukkan pada Gambar 1.

3.3. Implementasi Teknis di Unity

Logika FSM diimplementasikan menggunakan bahasa pemrograman C# di Unity Engine. Navigasi musuh dikendalikan menggunakan NavMeshAgent untuk bergerak di area permainan. Deteksi pemain dilakukan menggunakan perhitungan Vector3. Distance untuk radius jarak, dan Vector3.Angle untuk sudut pandang musuh terhadap pemain. FSM dijalankan dalam coroutine IEnumerator agar perilaku AI berjalan dinamis mengikuti kondisi lingkungan permainan.

3.4. Black-box Testing

Menguji seluruh skenario transisi antar state AI musuh untuk memastikan setiap kondisi berjalan sesuai dengan rancangan FSM tanpa memperhatikan isi kode program. Setiap kombinasi event dan kondisi diuji, seperti transisi dari Patrolling ke Chasing saat pemain terdeteksi, atau dari Attacking ke Dead saat health habis.

3.5. Uji Pengguna (User Playtesting)

Selain pengujian teknis, dilakukan uji coba gameplay oleh 10 responden untuk menilai kualitas perilaku AI musuh dalam permainan. Responden diminta mengisi kuesioner berbasis skala Likert 1–5 terhadap beberapa aspek, meliputi responsivitas musuh, variasi perilaku, kelancaran animasi, serta tingkat tantangan yang dirasakan. Perhitungan persentase skor tiap pertanyaan dilakukan dengan rumus:

$$Presentase\ skor = \frac{Total\ skor\ diperoleh}{Skor\ maksimum}\ x\ 100\%$$

Hasil playtesting digunakan untuk mengevaluasi pengalaman bermain dari perspektif pemain dan akan dibahas pada bagian Hasil dan Pembahasan.

3.6. Diagram Alur Metode Penelitian

Tahapan metode penelitian divisualisasikan dalam diagram alur berikut:

Gambar 2. Diagram alur metode penelitian

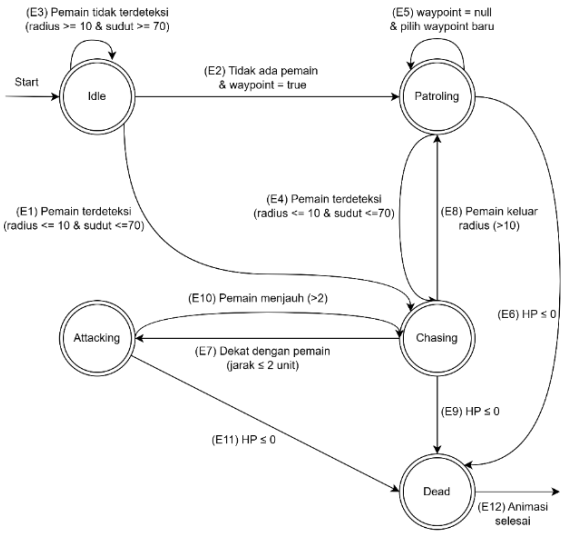
4. HASIL DAN PEMBAHASAN

4.1. Diagram FSM

Penelitian ini menghasilkan sistem AI musuh berbasis algoritma Finite State Machine (FSM) pada game Code Crusader: Anti-Virus Assault. AI musuh yang dikembangkan terdiri atas lima state utama, yaitu Idle, Patrol, Chase, Attack, dan Dead, yang diimplementasikan menggunakan Unity Engine dengan bahasa pemrograman C#.

Proses implementasi dimulai dari perancangan struktur FSM yang terdiri dari state-state tersebut, kemudian dilanjutkan dengan pembuatan logika transisi antar state berdasarkan kondisi di dalam permainan. AI musuh dapat melakukan patroli secara acak di area permainan menggunakan NavMeshAgent, mendeteksi keberadaan pemain dalam radius tertentu, melakukan pengejaran, menyerang saat jarak cukup dekat, dan kembali ke idle atau patrol ketika pemain menjauh atau saat musuh mati.

Berikut hasil implementasi diagram FSM untuk musuh dalam game:



Gambar 3. Diagram fsm musuh

Selain itu, sistem AI berhasil diintegrasikan dengan animasi karakter musuh serta sistem navigasi menggunakan NavMeshAgent, sehingga AI dapat bergerak mengikuti jalur sesuai NavMesh, menghindari rintangan, dan menyesuaikan kondisi permainan secara dinamis.

4.2. Tabel FSM

Berikut disajikan tabel transisi FSM untuk memudahkan pengembangan dan verifikasi logic FSM:

Tabel 1. Tabel fsm

State Saat Ini(Q)	Event(E)	Aksi(A)	State Berikutnya(Q′)
Idle	(E1) Pemain terdeteksi (radius ≤ 10 & sudut ≤ 70)	Mengejar pemain	Chasing
Idle	(E2) Tidak ada pemain & waypoint = true	Mulai patroli	Patrolling
Idle	(E3) Pemain tidak terdeteksi (radius >= 10 & sudut >= 70)	Tidak ada aksi	Idle
Patrolling	(E4) Pemain terdeteksi (radius <= 10 & sudut <=70)	Mengejar pemain	Chasing
Patrolling	(E5) waypoint = null & pilih waypoint baru	Pilih waypoint baru	Patrolling
Chasing	(E7) Dekat dengan pemain (jarak <= 2)	Serang pemain	Attacking
Chasing	(E8) Pemain keluar radius (>10)	Kembali ke patrol	Patrolling
Chasing	(E9) HP ≤ 0	Musuh mati	Dead
Attacking	(E10) Pemain menjauh (>2)	Kejar pemain	Chasing
Attacking	(E11) HP ≤ 0	Musuh mati	Dead
Dead	(E12) Animasi selesai	Hapus musuh	[*] (Exit)

4.3. Transisi FSM

Transisi yang ditentukan dalam tabel di atas berperan penting dalam memverifikasi implementasi logika FSM di dalam kode program, setiap kombinasi keadaan (Q) dan peristiwa (E) dipetakan menjadi state berikutnya (Q′) dengan notasi:

a. Dari State Idle

$$\begin{aligned}\delta(\text{Idle}, \text{E1}) &= \text{Chasing}\\ \delta(\text{Idle}, \text{E2}) &= \text{Patrolling}\\ \delta(\text{Idle}, \text{E3}) &= \text{Idle}\end{aligned} \quad (3)$$

b. Dari State Patrolling

$$\begin{aligned}&\delta(\text{Patrolling, E4}) = \text{Chasing}\\ &\delta(\text{Patrolling, E5}) = \text{Patrolling}\\ &\delta(\text{Patrolling, E6}) = \text{Dead}\end{aligned} \quad (4)$$

c. Dari State Chasing

$$\begin{aligned}&\delta(\text{Chasing, E7}) = \text{Attacking}\\ &\delta(\text{Chasing, E8}) = \text{Patrolling}\\ &\delta(\text{Chasing, E9}) = \text{Dead}\end{aligned} \quad (5)$$

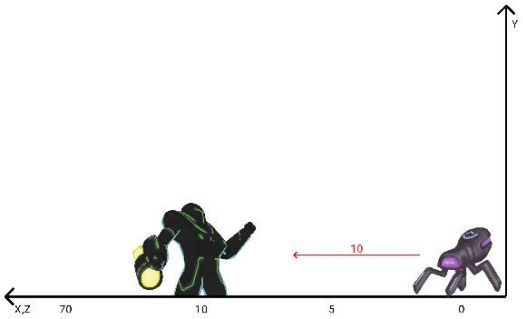
d. Dari State Attacking

$$\begin{aligned}&\delta(\text{Attacking, E10}) = \text{Chasing}\\ &\delta(\text{Attacking, E11}) = \text{Dead}\end{aligned} \quad (6)$$

e. Dari State Dead

$$\text{Dead, E12}) = [*] \text{ (exit, musuh dihapus)} \quad (7)$$

4.4. Posisi Player dan Musuh



Gambar 4. Jarak player dan musuh

Gambar 3 memperlihatkan radius pandangan (view radius) dari musuh dan pemain dalam lingkungan 3D, yang menunjukkan seberapa jauh musuh dapat mendeteksi keberadaan pemain berdasarkan koordinat (x, y, z).

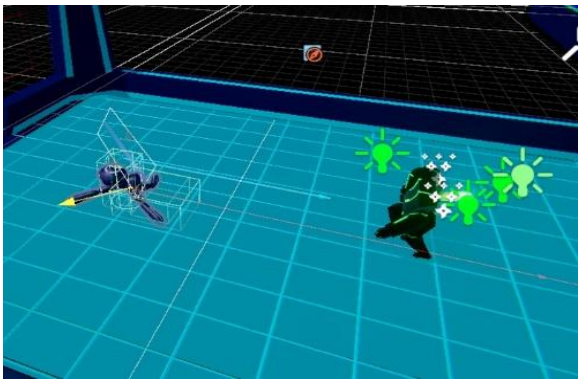
4.5. Implementasi Kode FSM

```
1.  private IEnumerator
2.  FSMStateLoop()
3.  {
4.     while (!isDead)
5.     {
6.        switch
7.  (nonBossEnemyState)
8.        {
9.           case
10. NonBossEnemyState.Idle:
11.              // AI diam,
12. menunggu waktu idle selesai
13.              break;
14.           case
15. NonBossEnemyState.Patrolling:
16.              // AI bergerak
17. menuju waypoint acak
18.              break;
19.           case
20. NonBossEnemyState.Chasing:
21.              // AI mengejar
22. pemain jika terdeteksi
23.              break;
24.           case
25. NonBossEnemyState.Attacking:
26.              // AI menyerang
27. pemain jika dalam jarak
28.              break;
29.           case
30. NonBossEnemyState.Dead:
31.              // AI mati dan
32. dihapus dari arena
33.              break;
34.        }
35.        yield return new
36. WaitForSeconds(1f);
37.     }
38. }
```

4.6. Implementasi dalam Gameplay

Gambar 5. Gameplay

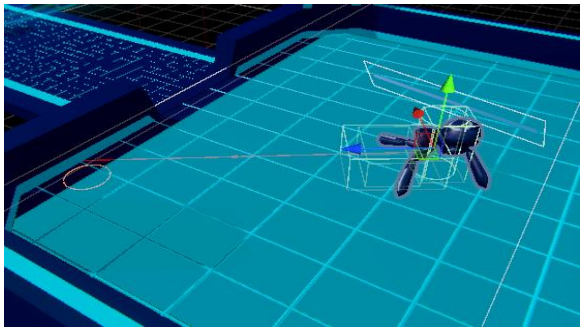
Gambar 5 menunjukkan tampilan saat pemain sedang berhadapan langsung dengan karakter musuh dalam permainan. Pada tahap ini, interaksi antara pemain dan musuh menjadi inti dari gameplay.



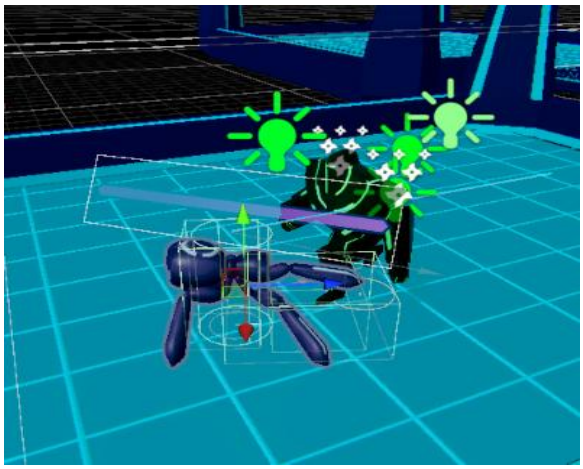
Gambar 6. Radius view musuh

Gambar 6 memperlihatkan bagaimana musuh memiliki kemampuan untuk mendeteksi kehadiran pemain dalam jarak tertentu yang disebut view radius. Ketika pemain memasuki area jangkauan ini, musuh akan langsung beralih ke mode pengejaran (chasing) dan mulai mengikuti arah pergerakan pemain. Fitur ini memberikan dinamika dalam permainan, di mana pemain harus tetap waspada terhadap posisi musuh di sekitar.

Gambar 7 ditunjukkan bahwa musuh tidak hanya diam di satu tempat, tetapi juga melakukan patroli secara acak menggunakan sistem waypoint. Gerakan acak ini membuat jalur musuh menjadi tidak dapat diprediksi, sehingga menambah elemen kejutan dan menuntut pemain untuk lebih cermat dalam menyusun strategi pergerakan. Perilaku ini menciptakan kesan bahwa musuh hidup dan terus beraktivitas di dalam dunia permainan.



Gambar 7. Patrol waypoint musuh



Gambar 8. Collider musuh

Gambar 8 menggambarkan area collider yang digunakan oleh musuh untuk mendeteksi tabrakan atau interaksi dengan pemain dalam rangka melakukan serangan. Ketika pemain masuk ke dalam area jangkauan serangan ini, musuh akan secara otomatis meluncurkan serangan. Mekanisme ini berperan penting dalam sistem pertarungan, memastikan bahwa interaksi antara pemain dan musuh berjalan secara real-time dan responsif.

4.7. Hasil Pengujian Black-box

Tabel 2 Hasil pengujian blackbox

Skenario	State Awal	Input/Kondisi	State Akhir	Hasil
Inisialisasi game	-	Game dimulai	Idle	Diterima
Deteksi pemain dalam radius	Idle	Player ≤10 unit, sudut ≤70°	Chasing	Diterima
Memulai patroli	Idle	Tidak ada player & waypoint tersedia	Patrolling	Diterima
Tetap idle	Idle	Player >10 unit	Idle	Diterima
Patroli ke chase	Patrolling	Player ≤10 unit, sudut ≤70°	Chasing	Diterima
Ganti waypoint	Patrolling	Waypoint null	Patrolling	Diterima
Chase ke attack	Chasing	Player ≤2 unit	Attacking	Diterima
Kembali patroli	Chasing	Player >10 unit	Patrolling	Diterima
Musuh mati saat chase	Chasing	HP ≤ 0	Dead	Diterima
Attack ke chase	Attacking	Player >2 unit	Chasing	Diterima
Musuh mati saat attack	Attacking	HP ≤ 0	Dead	Diterima
Hapus musuh	Dead	Animasi selesai	Musuh dihapus	Diterima

4.8. Hasil Pengujian Pengguna

Tabel 3. Hasil pengujian pengguna

No	Pertanyaan	Skor					Persentase
		1	2	3	4	5	
1	AI aktif saat idle (berjalan/diam tanpa pemain)	0	0	1	5	4	86%
2	AI segera menyadari pemain di depan	0	0	0	4	6	92%
3	AI berhenti mengejar saat pemain kabur	0	0	1	5	4	86%
4	Gerakan AI wajar dan natural	0	0	0	4	6	92%
5	AI memiliki variasi pola perilaku	0	0	0	4	6	92%
6	Pertarungan terasa menantang	0	0	1	5	4	86%
7	Butuh strategi berbeda untuk tiap tipe musuh	0	0	0	4	6	92%
8	AI responsif dan adaptif terhadap situasi permainan	0	0	0	4	6	92%
9	AI mendorong keinginan pemain untuk terus bermain	0	0	0	5	5	90%
10	AI langsung menyerang saat jarak dekat	0	0	0	4	6	92%

Tabel 4. Interpretasi nilai

Interval Persentase (%)	Kategori Interpretasi
81% – 100%	Sangat Baik
61% – 80%	Baik
41% – 60%	Cukup
21% – 40%	Kurang
0% – 20%	Sangat Kurang

Berdasarkan Tabel 4, hasil pengujian menunjukkan bahwa game ini memiliki rata-rata persentase sebesar 90%, yang termasuk dalam kategori "Sangat Baik". Hal ini menunjukkan bahwa sistem AI musuh telah berhasil memberikan pengalaman yang adaptif, menantang, dan menarik bagi pemain.

4.9. Pembahasan

Berdasarkan hasil pengujian, sistem AI musuh berbasis FSM dalam game Code Crusader: Anti-Virus Assault mampu berjalan sesuai desain. AI responsif terhadap kehadiran pemain dan dapat berpindah antar state dengan lancar. Pengujian menunjukkan AI dapat melakukan patrol, mengejar, menyerang, dan mati sesuai ketentuan skenario game.

Keunggulan FSM terlihat dari kemampuannya mengelola transisi perilaku AI secara modular dan efisien. Dibandingkan model AI lain seperti Behavior Tree, FSM lebih ringan dalam penggunaan resource dan cocok untuk AI musuh dengan jumlah state terbatas namun interaktif.

Keterbatasan penelitian ini adalah AI belum memiliki variasi perilaku kompleks seperti penghindaran serangan atau kolaborasi antar musuh. Selain itu, transisi hanya berbasis proximity detection, tanpa parameter health musuh atau strategi AI dinamis.

5. KESIMPULAN DAN SARAN

Berdasarkan hasil penelitian yang telah dilakukan, dapat disimpulkan bahwa algoritma Finite State Machine (FSM) berhasil diimplementasikan untuk mengatur sistem AI musuh pada game Code Crusader: Anti-Virus Assault. AI musuh mampu berpindah antar state secara sistematis sesuai kondisi permainan, yaitu mulai dari Idle, Patrolling, Chasing, Attacking, hingga Dead. Hasil pengujian menggunakan metode black-box testing membuktikan bahwa seluruh fungsi AI berjalan sesuai desain tanpa error. Hasil pengujian pengguna membuktikan Sistem AI ini meningkatkan dinamika interaksi antara pemain dengan musuh, serta memberikan pengalaman bermain yang responsif, adaptif dan lebih menantang.

Saran untuk penelitian selanjutnya adalah pengembangan AI musuh yang lebih adaptif dengan menambahkan parameter tambahan seperti health detection, area-of-effect attack detection, dan kelompok AI musuh (group AI). Selain itu, metode FSM dapat dikombinasikan dengan algoritma AI lainnya seperti Behavior Tree atau Decision Tree untuk menghasilkan perilaku AI yang lebih kompleks dan realistis dalam permainan tiga dimensi.

DAFTAR PUSTAKA

[1] D. Arifudin, M. Fransjaya, Y. El Fakhry, and H. A. Syahrizaldy, “Optimization of Player Experience and Enemy AI using A* Algorithm in Game,” *Sinkron*, vol. 9, no. 1, pp. 434–443, Feb. 2025, doi: 10.33395/sinkron.v9i1.14349.

[2] A. F. Syahri and M. Rizqi, “Sistem Perlawanan Musuh Dengan Artificial Intelligence (AI) Finite State Machine (FSM) Pada Game Petualangan,” *Jurnal Ilmu Komputer dan Bisnis*, vol. 15, no. 2, pp. 51–66, Nov. 2024, doi: 10.47927/jikb.v15i2.765.

[3] J. Khatib Sulaiman Dalam No, J. Wiratama, M. Evelin Johan, and H. Santoso, “Enhancing Gaming Engagement through the Integration of Game Design Document and Finite State Machine: A Study on Optimizing Non-playable Character Responsiveness,” *Indonesian Journal of Computer Science Attribution*, vol. 12, no. 4, pp. 2023–1627, Aug. 2023.

[4] A. Fauzi, A. A. Kusuma, M. A. Muin, and F. Syah, “Penerapan Finite State Machine Dalam Sistem Pertarungan Role Playing Game,” *Jurnal Dinamika Informatika*, vol. 14, no. 1, 2025.

[5] A. F. Syahri and M. Rizqi, “Sistem Perlawanan Musuh Dengan Artificial Intelligence (AI) Finite State Machine (FSM) Pada Game Petualangan,” *Jurnal Ilmu Komputer dan Bisnis*, vol. 15, no. 2, pp. 51–66, Nov. 2024, doi: 10.47927/jikb.v15i2.765.

[6] M. A. Ghani, A. B. Pohan, D. Gunawan, and Y. Saputra, “Penerapan Game Edukasi 3D Endless

Runner Berbasis Android Sebagai Media Belajar Matematika Anak," *Infotek: Jurnal Informatika dan Teknologi*, vol. 7, no. 1, pp. 288–298, Jan. 2024, doi: 10.29408/jit.v7i1.24194.

[7] R. Z. Alfarizi, Y. I. Kurniawan, and I. Permadi, "Game Edukasi 3D Berbasis Android untuk Memperkenalkan Peralatan Pendakian Gunung," *Jurnal Pendidikan dan Teknologi Indonesia*, vol. 4, no. 10, pp. 465–475, Feb. 2025, doi: 10.52436/1.jpti.569.

[8] N. F. Ramadhanti, M. Lamada, and M. Riska, "Pengembangan Aplikasi Game Edukasi 3D 'Finding Geometry' Berbasis Unity Sebagai Media Pembelajaran Bangun Ruang Matematika," *Jurnal MediaTIK*, vol. 4, no. 2, 2021, doi: 10.59562/mediatik.v4i2.3076.

[9] A. S. Aziz *et al.*, "Peningkatan Fleksibilitas dan Kecepatan Pengembangan Permainan Melalui Penerapan Pola Desain Pada Komponen Unity UI," *Teknika*, vol. 12, no. 3, pp. 232–242, Nov. 2023, doi: 10.34148/teknika.v12i3.679.

[10] T. R. S. Saqib, F. Nazir, and . M Awan, "Advanced AI Mechanics in Unity 3D for Immersive Gameplay. A Study on Finite State Machines & Artificial Intelligence," *IJIST*, Dec. 2024.

[11] T. Syahdina Riyan, A. M. H Pardede, and F. Yustika Manik, "Journal of Artificial Intelligence and Engineering Applications Implementation of Finite State Machine Models on the Artificial Intelligence System of Characters in The Game 'MMORPG' using RPG Maker," 2023. [Online]. Available: https://ioinformatic.org/