Jurnal Sains Informatika Terapan (JSIT)

E-ISSN: 2828-1659, Volume: 4, Issue: 3, Year: 2025
Page: 800-805, Available online at: https://rcf-indonesia.org/home/

IMPLEMENTASI LOGIKA FINITE STATE MACHINE DALAM PERILAKU ENEMY BOSS DALAM GAME 2D

Reka Adrian[1)], Mutaqin Akbar[2)]

[1,2]Informatika, Universitas Mercu Buana Yogyakarta

Corresponding Author: [1] adrianreka71@gmail.com, [2] mutaqin@mercubuna-yogya.ac.id

Article Info

Article history:

Received: Juny 26, 2025
Revised: July, 20, 2025
Accepted: sept, 24, 2025
Published: Okt, 30,2025

Keywords:

Finite State Machine,
AI Behavior,
2D Game,
Godot Engine,
GDScript,
Adaptive Enemy AI

ABSTRACT

This study presents the implementation of a Finite State Machine (FSM) to control enemy behavior in a 2D game environment based on the player's attack patterns. FSM is applied to manage dynamic transitions between states such as Idle, Attack, Teleport, and Summon. Each state is triggered by contextual gameplay variables including player proximity, number of consecutive hits, and enemy health level. The AI system was developed using the Godot Engine and GDScript, incorporating visual assets via sprite sheets linked to each behavioral state. The FSM framework allows the enemy character to react adaptively teleporting after repeated attacks and summoning reinforcements thus simulating intelligent behavior. System evaluation was conducted using black-box testing in simulated combat scenarios. Results demonstrate that the FSM-based AI provides enhanced gameplay variability, responsiveness, and strategic challenge, while maintaining system modularity and scalability. The findings support the effectiveness of FSM for real-time enemy behavior modeling in 2D games and offer a reference framework for future development of adaptive AI in interactive digital environments.

1. INTRODUCTION

Industri game merupakan salah satu sektor ekonomi digital dengan pertumbuhan paling pesat dalam beberapa dekade terakhir. Menurut laporan dari Newzoo (2024), pendapatan industri game global diperkirakan telah melampaui USD 200 miliar, didorong oleh pertumbuhan teknologi, penetrasi internet, dan adopsi perangkat mobile secara masif [9]. Perkembangan ini tidak hanya memengaruhi aspek hiburan, tetapi juga memperluas dampak game dalam bidang pendidikan, simulasi, hingga sosial budaya [16], Dalam konteks gameplay, game digital saat ini terbagi menjadi dua kategori utama, yakni game offline dan game online. Game offline bersifat lokal dan tidak memerlukan koneksi jaringan, dengan fokus pada pengalaman naratif dan kontrol tunggal oleh pemain. Sebaliknya, game online menekankan konektivitas sosial dan kompetisi waktu nyata melalui jaringan internet, dan telah melahirkan model bisnis baru seperti *freemium*, *microtransaction*, dan *Games as a Service (GaaS)* [16]. Kombinasi keduanya bahkan menjadi tren umum, dengan banyak game menawarkan mode campuran antara offline dan online secara adaptif, Salah satu aspek penting dalam pengembangan game modern adalah perilaku musuh atau *enemy behavior*, terutama pada game bergenre aksi dan petualangan. Musuh yang dapat beradaptasi terhadap pola permainan pemain dinilai mampu meningkatkan tantangan, kepuasan, dan tingkat keterlibatan pemain [1], [2]. Untuk mencapai tingkat kecerdasan tersebut, pengembang umumnya menggunakan pendekatan berbasis kecerdasan buatan (*Artificial Intelligence*/AI). Beberapa algoritma umum yang digunakan antara lain *Finite State Machine* (FSM) [3], [5], [7].

FSM menjadi pendekatan yang paling banyak digunakan untuk mengatur perilaku karakter dalam game karena bentuknya yang modular, terstruktur, dan mudah dikembangkan. FSM memungkinkan sistem berpindah antar kondisi atau *state* berdasarkan stimulus lingkungan seperti jarak musuh terhadap pemain, jumlah serangan yang diterima, atau waktu jeda antar aksi [2], [4]. Millington dan Funge [1] menjelaskan bahwa FSM memberikan kerangka logika sederhana namun efektif untuk mengendalikan AI musuh. Hal ini diperkuat oleh Orkin [6] dalam studinya terhadap sistem AI game *F.E.A.R.*, yang membuktikan bahwa FSM dapat dikembangkan menjadi sistem adaptif ketika dikombinasikan dengan variabel kontekstual,

Beberapa game populer yang menggunakan FSM sebagai inti logika perilaku musuh antara lain Pac-Man dan Hollow Knight. Dalam *Pac-Man*, setiap

hantu memiliki FSM yang mengatur tiga mode perilaku utama: *Chase*, *Scatter*, dan *Frightened*. Transisi antar mode ini ditentukan oleh waktu, posisi karakter utama, serta interaksi pemain [7]. Di sisi lain, *Hollow Knight*, game aksi platformer 2D, menggunakan FSM untuk mengatur berbagai pola serangan dan respons bos dalam pertempuran. Setiap bos memiliki serangkaian *state* seperti *Idle*, *Jump*, *Attack*, dan *Dash*, yang diatur berdasarkan jarak terhadap pemain dan siklus serangan [15].

Dalam pengembangan sistem AI, pemilihan game engine menjadi aspek penting karena memengaruhi arsitektur pemrograman, manajemen animasi, dan integrasi logika permainan. Unity dan Godot adalah dua engine game yang populer dan mendukung implementasi FSM. Unity menawarkan fitur yang sangat lengkap dengan ekosistem yang besar, integrasi scripting berbasis C#, serta dukungan tools visual seperti Animator Controller untuk FSM. Namun, kompleksitas antarmuka dan ketergantungan pada lisensi menjadikannya kurang fleksibel untuk pengembang indie [14], Sebaliknya, Godot Engine bersifat open-source, ringan, dan menyediakan sistem node yang modular, memungkinkan implementasi FSM secara langsung melalui pemrograman menggunakan GDScript atau menggunakan *AnimationTree*, *StateMachine*, dan *Scripted State Manager* [8]. Dalam konteks game 2D, Godot lebih unggul dari sisi efisiensi performa dan pengaturan scene, serta memudahkan integrasi visual dengan logika status. Oleh karena itu, Godot menjadi pilihan yang tepat untuk penelitian ini, khususnya dalam merancang musuh AI yang kompleks namun tetap mudah diuji dan dikembangkan secara bertahap [10]–[13], Penelitian ini bertujuan untuk mengimplementasikan logika FSM pada sistem kecerdasan buatan musuh dalam game 2D berbasis Godot Engine, di mana musuh mampu mengenali pola serangan pemain dan merespons secara adaptif. Sistem dirancang untuk berpindah state secara real- time berdasarkan stimulus tertentu, guna menciptakan tantangan yang lebih realistis dan pengalaman bermain yang lebih dinamis.

2. MATERIALS AND METHODS

2.1. Materials

Material utama yang digunakan dalam penelitian ini terdiri dari perangkat lunak pengembang, bahasa pemrograman, serta aset grafis untuk karakter musuh. Pengembangan dilakukan menggunakan Godot Engine, yaitu mesin pengembangan game *open-source* yang mendukung penggunaan bahasa GDScript. Godot dipilih karena memiliki fitur pengelolaan animasi dan logika yang cocok untuk penerapan *Finite State Machine* (FSM), Selain perangkat lunak, beberapa aset visual digunakan untuk mendukung representasi perilaku karakter musuh. Aset-aset tersebut disusun dalam bentuk sprite sheet yang masing-masing terdiri dari beberapa frame animasi. Aset tersebut meliputi:

1. Idle: Animasi diam ketika musuh belum mendeteksi pemain.
2. Follow: Animasi ini bergerak mengambang mengejar player.
3. Attack: Gerakan menyerang yang aktif saat AI dalam state Attack
4. Skill: Animasi teleportasi saat musuh menghindar dari serangan.
5. Summon: Gerakan pemanggilan unit musuh tambahan.
6. Death: Animasi kematian ketika musuh dikalahkan.

Setiap animasi dihubungkan langsung dengan sistem FSM dalam game, dan diatur menggunakan node *AnimatedSprite2D* atau *AnimationPlayer* untuk memastikan transisi visual berjalan sesuai dengan logika AI yang dirancang.

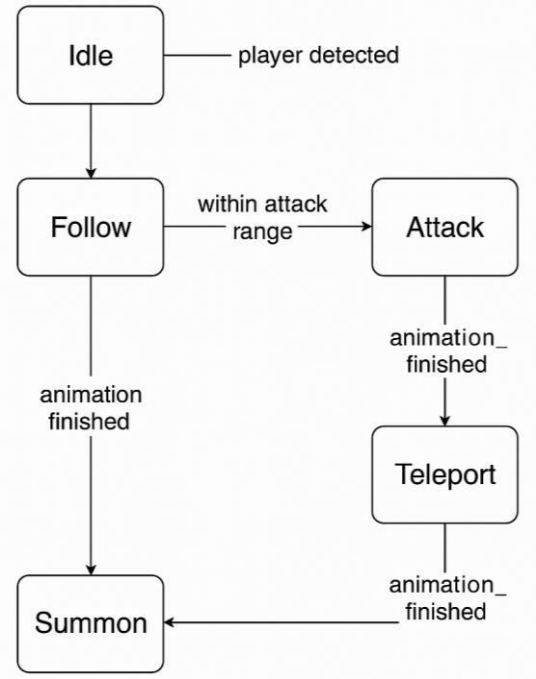
2.2. Metodologi

Penelitian ini dilakukan melalui pendekatan rekayasa perangkat lunak secara eksperimental yang terdiri atas beberapa tahapan sistematis, yaitu analisis kebutuhan, perancangan sistem, implementasi, serta pengujian.

Tahap pertama adalah analisis kebutuhan, yang bertujuan untuk mengidentifikasi fitur-fitur utama yang harus dimiliki oleh sistem kecerdasan buatan (AI) musuh dalam game 2D. Dalam konteks ini, musuh dirancang agar mampu merespons pola permainan pemain secara adaptif melalui mekanisme transisi antar status (*state*), seperti diam (*Idle*), mengejar (*Follow*), menyerang (*Attack*), menghindar (*Teleport*), dan memanggil musuh tambahan (*Summon*). Selain itu, dipertimbangkan pula kebutuhan nonfungsional seperti efisiensi sistem, keterbacaan kode, modularitas, serta kecepatan transisi dalam waktu nyata.

Selanjutnya dilakukan perancangan sistem, yang berfokus pada pembangunan arsitektur logika *Finite State Machine* (FSM). FSM dirancang dengan pendekatan *state pattern*, di mana setiap state didefinisikan dalam skrip terpisah yang merupakan turunan dari kelas dasar State.gd. Transisi antar state diatur berdasarkan kondisi permainan seperti jarak pemain terhadap musuh (distance_to_player), jumlah serangan yang diterima (hit_counter), dan status kesehatan musuh (health). Struktur transisi antar state divisualisasikan melalui diagram alur untuk memastikan keterurutan logika yang tepat dan dapat diuji, Tahap berikutnya adalah implementasi sistem, yang dilaksanakan menggunakan Godot Engine dengan bahasa pemrograman GDScript. Implementasi dimulai dengan membangun skrip dasar State.gd, yang menjadi kerangka semua status. *Fungsi enter()* dipanggil saat sebuah state diaktifkan, *exit()* ketika state ditinggalkan, dan *update()* untuk

logika berkelanjutan. Bisa kita lihat pada Gambar diagram alur di bawah:



Gambar 1. diagram alur state

Dengan pendekatan ini, sistem FSM dirancang secara modular, fleksibel, dan terintegrasi erat dengan animasi serta struktur node di Godot Engine. Setiap *state* dapat dikembangkan dan dimodifikasi secara independen tanpa memengaruhi bagian lain dari system untuk memungkinkan pengembangan berkelanjutan. Implementasi ini mendemonstrasikan penerapan FSM yang adaptif dan mendukung dinamika permainan secara real-time, sekaligus menunjukkan bahwa pendekatan berbasis FSM sangat cocok untuk membangun sistem kecerdasan musuh yang responsif dan kompleks dalam permainan 2D.

Pengujian sistem dilakukan untuk mengevaluasi apakah logika *Finite State Machine* (FSM) yang diimplementasikan telah berjalan sesuai dengan kebutuhan fungsional dan desain sistem. Pengujian ini menggunakan pendekatan *black-box testing*, yaitu metode pengujian yang berfokus pada respons sistem terhadap input tertentu tanpa memperhatikan struktur internal kode. Dengan pendekatan ini, sistem diuji dari sisi pengguna dengan mengamati hasil keluaran dari setiap interaksi pemain dengan musuh dalam permainan, Pengujian dilakukan secara sistematis berdasarkan skenario-skenario interaksi dalam game, seperti ketika pemain masuk ke radius deteksi musuh, melakukan serangan berturut-turut, atau menjauh dari area interaksi. Fokus utama dari pengujian adalah memastikan transisi antar state seperti dari *Idle* ke *Follow*, *Follow* ke *Attack*, *Attack* ke *Teleport*, hingga ke *Summon* berjalan sesuai dengan parameter logika yang telah ditentukan sebelumnya, Selain pengujian fungsional, sistem juga diuji dari sisi kestabilan dan kecepatan respons transisi antar state untuk memastikan sistem mampu merespons secara *real-time* tanpa hambatan. Hasil dari pengujian ini akan disajikan dan dianalisis lebih lanjut pada bab hasil dan pembahasan.

2.3 Aset

Dalam penelitian ini, digunakan sejumlah aset visual dalam bentuk *sprite sheet* untuk merepresentasikan animasi karakter musuh (boss) dalam berbagai kondisi. Aset-aset ini dirancang untuk terintegrasi langsung dengan sistem *Finite State Machine* (FSM) yang mengendalikan logika perilaku musuh di dalam permainan 2D.

Setiap aset mewakili satu state (keadaan) yang dijalankan oleh sistem FSM. Misalnya, ketika pemain belum berada dalam jangkauan deteksi, karakter musuh akan menampilkan animasi *Idle*; saat mendekat, karakter akan masuk ke status *Attack*; dan jika menerima sejumlah serangan berturut-turut, karakter akan masuk ke status *Teleport* untuk menghindar, lalu dilanjutkan dengan *Summon* guna memanggil bantuan. Jika karakter dikalahkan, maka animasi *Death* akan dijalankan secara otomatis.

Rincian aset yang digunakan disajikan dalam Tabel 1 berikut, yang mencakup nama aset dan animasi gambar atau keterkaitannya dengan FSM, serta deskripsi singkat dari masing-masing animasi.

Tabel 1: Aset Boss

Nama	Animasi
IDLE DAN FOLLOW	
ATTACK	
SKILL (TELEPORT)	
SUMMON	
DEATH	

Aset-aset ini diatur melalui node *AnimationPlayer* atau *AnimatedSprite2D* di dalam Godot Engine, dan masing-masing diaktifkan secara

otomatis oleh sistem FSM berdasarkan kondisi permainan yang terjadi. Penyesuaian kecepatan animasi dan transisi antar frame dilakukan agar menghasilkan pengalaman visual yang halus dan selaras dengan logika permainan, Penggunaan dataset aset ini sangat penting dalam membangun pengalaman bermain yang imersif dan responsif, serta memastikan setiap aksi karakter musuh memiliki umpan balik visual yang konsisten dengan logika FSM yang dirancang.

Pengujian sistem dalam penelitian ini dilakukan dengan menggunakan metode black-box testing, yaitu pendekatan pengujian perangkat lunak yang berfokus pada evaluasi fungsi sistem berdasarkan respons terhadap input yang diberikan, tanpa memperhatikan struktur internal atau implementasi kode program. Setiap state diuji secara langsung melalui simulasi permainan di dalam Godot Engine, tanpa melihat atau memodifikasi kode program. Output yang diamati meliputi perubahan animasi, perpindahan status musuh.

3. RESULTS AND DISCUSSION

Pada bab ini dijelaskan hasil implementasi sistem kecerdasan buatan berbasis *Finite State Machine* (FSM) dalam membaca dan merespons pola serangan karakter utama dalam permainan 2D. Sistem dirancang agar musuh (enemy musuh) mampu merespons secara dinamis terhadap tindakan pemain melalui transisi antar *state* yang terdefinisi secara sistematis. Setiap *state* dikaitkan dengan animasi visual menggunakan aset *sprite sheet* yang telah dikembangkan dan disesuaikan dengan logika permainan. Berikut ini adalah uraian aset yang digunakan serta implementasinya dalam sistem FSM, mulai dari idle, follow, attack, skill/teleport, summon, dan death.

3.1. Idle

Gambar 8. Idle boss

State Idle merupakan kondisi awal atau dasar ketika musuh tidak mendeteksi kehadiran pemain. Dari state ini, karakter hanya dapat berpindah ke *Follow* jika pemain masuk dalam radius atau deteksi tertentu. Tidak dapat langsung berpindah ke Attack, Teleport, atau Summon, Aset Idle berisi delapan frame animasi bertema grim reaper yang menggambarkan karakter dalam posisi diam namun tetap dinamis. Gerakan kecil pada tubuh dan senjata menciptakan kesan siap siaga. Aset ini diatur melalui node *AnimatedSprite2D* atau *AnimationPlayer* di Godot Engine.

3.2. Follow

Gambar 9. Follow

State Follow merupakan fase ketika musuh mengejar player dengan cara melayang pada state ini menggunakan aset idle yang di kembangkan agara bias mengejar player, Pada saat musuh berada dalam state *Follow*, sistem mengarahkan karakter untuk mendekati pemain. Transisi dari state ini dapat menuju ke beberapa state dengan ketentuannya masing-masing seperti state *Follow* dapat ke state *Attack* jika posisi pemain sudah berada dalam jangkauan serang dan state *Follow* ke state *Idle* jika pemain keluar dari radius deteksi musuh serta state *Follow* ke state *Teleport* jika musuh menerima serangan secara beruntun sebanyak 3 kali dalam jangka waktu pendek.

3.3. Attack

Gambar 10. attack

Pada bagian ini Boss akan berpindah dari *Idle* ke *follow* dan musuh akan menyerang ketika musuh berada dalam jarak serangan, sistem akan masuk ke state *Attack*, State Attack berfungsi untuk mengekskusi animasi serangan, Dari state ini transisinya hanya dapat di lakukan ke beberapa state dengan ketentuannya masing-masing, State Teleport jika serangan pemain mengenai musuh sebanyak 3 kali secara beruntun maka state Attack akan stop beralih ke state Teleport

3.4. Skill/Teleport

Gambar 11. Skill/teleport

Setelah musuh menyerang musuh akan menerima serangan berturut-turut sebanyak tiga kali, Serangan ini yang memeicu transisi ke state *Teleport*. Efek visual seperti bayangan atau distorsi digunakan untuk menggambarkan karakter menghilang dari lokasi semula dan muncul di lokasi lain. State *Teleport* merupakan kondisi penghindaran, Setelah animasi *Teleport* selsai, Transisi hanya diizinkan ke *Summon* sebagai bentuk strategi kelanjutan setelah menghindar. State *Teleport* tidak diperbolehkan langsung ke state *Attack* atau *Idle* melainkan state *Summon* terlebih dahulu.

3.5. Summon

Gambar 12. Summon

Setelah Boss melakukan teleportasi, Boss akan memanggil unit tambahan melalui state Summon. Aset Summon terdiri dari lima frame yang menggambarkan ritual pemanggilan, dengan posisi tangan dan senjata mengarah ke atas. Efek visual seperti partikel gelap atau cahaya digunakan untuk memperkuat atmosfer magis. Setelah animasi selesai dan bala bantuan sudah menyerang, Boss baru akan muncul secara instan di lokasi tertentu menggunakan fungsi instance() di Godot.

3.6. Death

Gambar 13. Death

Aset *Death* digunakan ketika musuh kehabisan poin kesehatan, animasi ini memperlihatkan tubuh karakter yang hancur secara bertahap, Animasi ini diatur agar tidak terulang, dan setelah selesai, node karakter akan dihapus dengan *queue_free()*, Sistem dapat masuk ke state *Death* sebagai state terminal transisi pada state *Death* tidak di izinkan dari state ini ke state lainnya.

Gambar-gambar berikut merupakan prilaku musuh untuk memperkuat keterkaitan antara logika FSM dan representasi visual, sehingga transisi antar state dapat ditampilkan secara imersif dan tidak membingungkan pemain. Integrasi ini juga membantu menjaga konsistensi antara desain AI dan pengalaman bermain, Perilaku musuh dapat menjawab bahwa sistem FSM yang dirancang mampu berjalan dengan tingkat respons yang baik, Seperti pada kondisi awal, musuh berada dalam state *Idle* dan akan berubah ke *Follow* setelah pemain terdeteksi dan Ketika pemain sudah berada dalam jangkauan serang, musuh akan masuk ke state *Attack*. Jika menerima tiga serangan berturut-turut atau berada dalam ambang batas kesehatan tertentu, musuh akan *Teleport* ke posisi lain, kemudian memicu state *Summo,* State ini akan berulang hingga musuh kekurangan poin untuk hidup hingga state *Death* aktif pada akhir pertarungan game.

Setelah pengujian dilakukan berdasarkan skenario yang telah dirancang, hasil pengujian menunjukkan bahwa sistem FSM dapat merespons setiap input dari pemain dengan transisi state yang tepat dan sesuai harapan. Setiap state diuji dengan kondisi yang memicu aktivasi state tersebut, dan diamati apakah perpindahan terjadi secara mulus, logika dijalankan dengan benar, serta animasi terhubung dapat diselesaikan. Hasil pengujian dicatat dalam bentuk tabel dapat dilihat pada Tabel 2.

Tabel 2. Hasil Pengujian State

Skenario Pengujian	Kondisi Pengujian	State yang Diharapkan	Hasil
Pemain belum terdeteksi	Jarak > 300 piksel	Idle	✓
Pemain memasuki area deteksi musuh	deteksi musuh Jarak ≤ 300 piksel	Follow	✓
Pemain berada dalam jangkauan serangan	Jarak ≤ 100 piksel	Attack	✓
Musuh menerima 3 serangan berturut-turut	hit_counter ≥ 3	Teleport	✓
Animasi teleportasi selesai	animation_finished == true	Summon	✓
Summon selesai memanggil musuh tambahan	Fungsi spawn() berhasil dijalankan	Follow	✓
Pemain menjauh setelah pengejaran	Jarak kembali > 300 piksel	Idle	✓
Serangan tidak memenuhi syarat teleport	hit_counter < 3 dan animasi selesai	Follow	✓

Pengujian membuktikan bahwa seluruh transisi antar state berjalan lancar dan sesuai dengan logika sistem yang telah dirancang. Tidak ditemukan kegagalan logika, keterlambatan transisi, maupun animasi yang tidak sinkron. Waktu perpindahan antar state terjadi dengan rata-rata kurang dari 0,1 detik, membuktikan bahwa sistem FSM bekerja secara *real-time* dan responsif terhadap kondisi permainan.

4. CONCLUSION

Penelitian ini menyimpulkan bahwa penerapan logika *Finite State Machine* (FSM) dalam permainan 2D terbukti efektif dalam membentuk perilaku musuh yang adaptif dan dinamis. FSM mampu mengatur transisi antar status seperti *Idle*, *Follow*, *Attack*, *Teleport*, dan *Summon* secara sistematis berdasarkan kondisi permainan, seperti jarak terhadap pemain, jumlah serangan yang diterima, serta status kesehatan musuh. Implementasi menggunakan GDScript di Godot Engine menghasilkan sistem yang stabil, modular, dan mudah dikembangkan. Struktur FSM yang terorganisir mendukung efisiensi pengembangan serta memungkinkan penambahan perilaku baru tanpa mengubah arsitektur utama. Penelitian ini memberikan kontribusi terhadap pengembangan AI dalam game 2D dan dapat menjadi acuan untuk penelitian lanjutan dalam menciptakan AI yang lebih kompleks dan responsif.

REFERENCES

[1] Muhammad B. Rizqi Alvian, S. Bukhori, & M. A. Furqon, "Implementasi Finite State Machine untuk Menentukan Perilaku NPC pada Game Simulasi Kepemimpinan," *J. Simulasi Kepemimpinan*, Nov. 2023.

[2] A. Akram, R. Tehseen, S. Saqib, F. Nazir, dan M. Mehr Awan, "Advanced AI Mechanics in Unity 3D for Immersive Gameplay: A Study on Finite State Machines & Artificial Intelligence," *Int. J. Innov. Sci. Technol.*, vol. 6, no. 4, hlm. 2047–2068, Des. 2024.

[3] Russell, S. J., & Norvig, P. (2021). *Artificial Intelligence: A Modern Approach*. Pearson.

[4] Champandard, A. J. (2004). *AI Game Development: Synthetic Creatures with Learning and Reactive Behaviors*. New Riders.

[5] D. Aversa, *Unity Artificial Intelligence Programming: Add Powerful, Believable, and Fun AI Entities in Your Game*, 5th ed., Birmingham: Packt, 2022. (meskipun bukan jurnal, edisi 2022 ini relevan untuk penerapan FSM di Unity.

[6] Orkin, J. (2006). Three States and a Plan: The AI of F.E.A.R. *GDC Lecture*.

[7] Rabin, S. (2015). *Game AI Pro: Collected Wisdom of Game AI Professionals*. CRC Press.

[8] Godot Engine Documentation. https://docs.godotengine.org.

[9] Menouar, H. et al. (2017). UAV-enabled intelligent transportation systems. *IEEE Communications Magazine*.

[10] P. Qureshi, "Gameplay Analysis Using Finite State Machines," *Conference on Automata Theory in Game Development*, Jun. 2024.

[11] Wijaya, R. (2020). Pengembangan AI Musuh dalam Game Menggunakan FSM. *Jurnal Ilmu Komputer*.

[12] C. Nugraha, A. I. Purnamasari, A. Bahtiar, dan E. Tohidi, "Implementation of Finite State Machine on NPCs to Improve Game Productivity," *J. Artif. Intell. Eng. Appl.*, vol. 4, no. 3, pp. 1673–1677, Jun. 2025.

[13] Priambodo, A. (2021). FSM Implementation on Enemy AI in 2D Action Game. *IEEE Explore*.

[14] Hartanto, A. & Suryani, L. (2020). *Design Pattern dan FSM dalam Game Development*. Jurnal Teknologi Informasi.

[15] Budiman, R. (2022). Adaptive Enemy AI Behavior in 2D Games Using FSM. *International Conference on Game Technology*.

[16] Statista, "Gaming Market Size Worldwide 2022–2027." [Online]. Available: https://www.statista.com.