

Implementasi Kecerdasan Buatan dalam Mengatur Mekanik Musuh pada Game Lily's Dreamland: Chaos in the Storybook

Muhammad Aksal Alshal N. Surianto^a, Dedy Atmajaya^b, Syahrul Mubarak Abdullah^c

Universitas Muslim Indonesia, Makassar, Indonesia

^a13020190300@umi.ac.id; ^bdedy.atmajaya@umi.ac.id; ^csyahrul.mubarak@umi.ac.id

Received: 23-01-2025 | Revised: 22-02-2025 | Accepted: 15-03-2025 | Published: 29-03-2025

Abstrak

Game merupakan salah satu media hiburan yang sangat populer dan terus berkembang di era digital saat ini. Salah satu elemen penting dalam game adalah *Non-Playable Character* (NPC) yang dapat memengaruhi pengalaman bermain pemain. NPC musuh yang memiliki perilaku dinamis dan adaptif dapat meningkatkan tantangan dan kepuasan dalam bermain. Penelitian ini bertujuan untuk mengimplementasikan kecerdasan buatan atau biasa disebut *Artificial Intelligence* (AI) dengan metode *Finite State Machine* (FSM) dalam mengatur mekanik NPC musuh pada game *Lily's Dreamland: Chaos in the Storybook*, sebuah game 2D *platformer* yang dibuat dengan Unity Engine. Metode yang digunakan meliputi perancangan FSM yang terdiri dari *state*, *event*, dan *action* untuk mengatur perilaku NPC seperti menyerang, menghindari, dan berpatroli. Algoritma FSM akan memungkinkan NPC untuk merespons situasi yang berubah-ubah dan menyesuaikan strategi berdasarkan tindakan pemain. Berdasarkan hasil, penerapan FSM pada NPC musuh dalam game ini bekerja dengan baik dan diharapkan dapat menciptakan perilaku NPC yang lebih dinamis dan adaptif, sehingga meningkatkan kualitas dan kepuasan dalam bermain game.

Kata kunci: kecerdasan buatan, *finite state machine*, *non playable character*, game 2D *platformer*, unity engine.

Pendahuluan

Game merupakan sebuah media hiburan yang sangat populer dan berkembang pesat di era digital [1], [2]. Game sering kali diklasifikasikan berdasarkan genre, game memiliki banyak genre yaitu, aksi, petualangan, simulasi, *puzzle*, dll [3]. Ada banyak aspek dalam sebuah game, salah satunya yaitu mekanik game, mekanik game merupakan aturan dan sistem yang mengatur bagaimana game berfungsi dan berinteraksi dengan pemain [4]. Mekanik game dapat mencakup berbagai hal, seperti kontrol, grafis, suara, cerita, karakter, level, dll. Salah satu mekanik game yang sering digunakan dan menarik perhatian pemain adalah *non playable character* (NPC). NPC merupakan karakter dalam sebuah game yang tidak dapat dimainkan oleh pemain, tetapi hanya dapat dikendalikan oleh komputer [5]. Dalam dunia game NPC memiliki beberapa jenis, yaitu NPC teman, NPC misi, NPC pendukung cerita, NPC musuh, dll [6].

NPC musuh merupakan karakter yang memiliki peran untuk menghalangi tujuan pemain. Dengan adanya NPC musuh dapat memberikan tantangan, variasi, dan dinamika terhadap game. Oleh karena itu, tingkat kecerdasan NPC musuh dapat memengaruhi pengalaman bermain pemain [6]. Maka dari itu, membuat NPC musuh yang menantang dan bervariasi adalah salah satu tantangan dalam pengembangan game.

Salah satu metode untuk mencapai hal tersebut yaitu dengan menggunakan kecerdasan buatan atau biasa disebut *Artificial Intelligence* (AI). Kecerdasan buatan adalah suatu proses yang diterapkan pada teknologi untuk meniru cara berpikir manusia dan membuat mesin dapat melakukan tugas-tugas yang biasa dilakukan oleh manusia [7]. AI dapat memberikan perilaku yang lebih dinamis dan adaptif kepada NPC musuh, sehingga NPC musuh dapat merespons situasi yang berubah-ubah, menyesuaikan strategi, atau bahkan belajar dari pemain [8].

Pada penelitian yang dilakukan oleh Argia Pranselga, menunjukkan bahwa FSM berhasil memberikan kecerdasan buatan pada NPC yang merespons tindakan pemain, sehingga game terasa lebih hidup dan tidak membosankan. Dari kuesioner pengujian game, 45% responden menyatakan sangat sesuai, 44% sesuai, 11% tidak sesuai, dan 0% sangat tidak sesuai [1]. Pada penelitian lainnya, yang dilakukan oleh Wahyu Safitra, menunjukkan bahwa FSM cocok untuk diterapkan pada game yang ia buat, sehingga game tersebut dapat meningkatkan tantangan dan melatih kemampuan berpikir pemain [9]. Selain itu, Penelitian oleh Muhamad Faris Al Kautsar membahas penerapan FSM dalam perancangan perilaku musuh pada game 2D *platformer* bertema sejarah *Glory of Matawis*. Dalam penelitian tersebut, FSM digunakan untuk mengatur aksi musuh berdasarkan parameter seperti jarak dengan pemain dan jumlah health points (HP) yang tersisa. Hasilnya

menunjukkan bahwa musuh dapat menampilkan perilaku dinamis yang berubah sesuai dengan kondisi yang diterima, sehingga meningkatkan pengalaman bermain yang lebih menarik dan interaktif [10].

Implementasi ini menerapkan salah satu metode AI, yaitu *Finite State Machine* (FSM). FSM merupakan sebuah mesin abstrak yang berfungsi untuk mendefinisikan serangkaian kondisi dengan menentukan kapan suatu keadaan akan berubah [11]. Menurut Rahadian, FSM adalah sebuah metodologi perancangan sistem kontrol yang menggambarkan tingkah laku atau prinsip kerja sistem dengan menggunakan tiga elemen berikut: keadaan (*state*), kejadian (*event*) dan aksi (*action*) [12]. Dengan menggunakan FSM, perilaku musuh dapat dibagi menjadi beberapa *state* yang saling berkaitan, seperti menyerang, menghindar, berpatroli dll [9], [13].

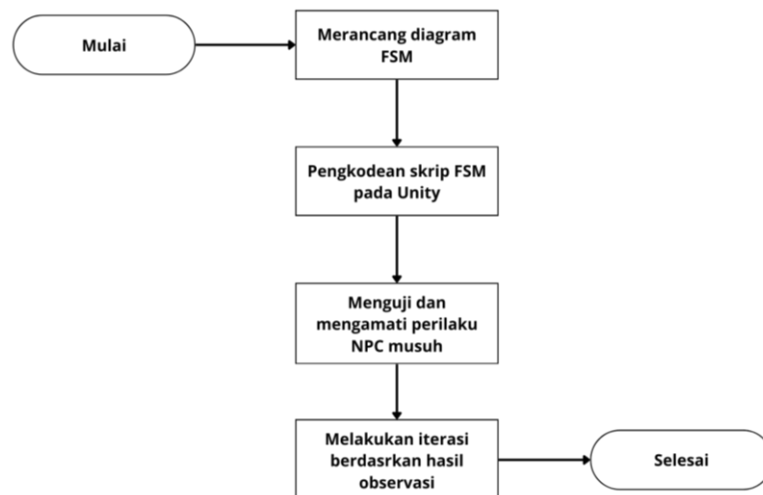
Tujuan dan fokus implementasi ini, untuk mengimplementasikan AI dengan metode FSM dalam mengatur mekanik NPC musuh pada game *Lily's Dreamland: Chaos in the Storybook*. Game ini adalah game genre 2D platformer yang dibuat dengan Unity Engine. Game genre 2D platformer adalah game yang menekankan pada aksi melompat atau memanjat antara platform yang berbeda di layar [14]. Game *Lily's Dreamland: Chaos in the Storybook* bercerita tentang petualangan Lily, seorang gadis kecil yang suka membaca buku dongeng. Suatu hari, ia bermimpi masuk ke dalam dunia dongeng pada buku yang ia baca dan terjebak dalam dunia dongeng tersebut yang penuh dengan karakter aneh dan musuh-musuh jahat. Lily harus mencari jalan keluar dari dunia dongeng tersebut dengan menghadapi berbagai rintangan dan teka-teki.

Metode

Implementasi AI ini menggunakan metode FSM dalam mengatur mekanik NPC musuh dalam game *Lily's Dreamland Chaos in the Storybook*. FSM merupakan sebuah metodologi perancangan sistem Kontrol yang menggambarkan tingkah laku atau prinsip kerja sistem dengan menggunakan tiga hal berikut [15]:

- A. Keadaan (*State*): Menyatakan kondisi tertentu dari NPC, seperti menyerang, menghindar, berpatroli, dll.
- B. Kejadian (*Event*): Menyatakan kejadian yang memicu perubahan dari satu *state* ke *state* lain, seperti mendeteksi pemain, terkena serangan, dll.
- C. Aksi (*Action*): Menyatakan tindakan yang dilakukan oleh NPC saat berada dalam suatu *state* atau saat transisi antara *state*.

Dalam Implementasi FSM pada NPC musuh pada game *Lily's Dreamland: Chaos in the Storybook*, dilakukan langkah-langkah berikut:



Gambar 1. Langkah-langkah Implementasi FSM

Gambar 1 menunjukkan langkah-langkah implementasi FSM pada NPC musuh, yang dimulai dengan merancang diagram FSM yang menggambarkan *state* dan transisi yang diinginkan, kemudian ke tahap mengkodekan *state* dan transisi menggunakan C# pada Unity Engine, tahap selanjutnya melakukan pengujian dan mengamati perilaku NPC dalam game untuk memastikan FSM bekerja sesuai dengan yang diharapkan, setelah itu dilakukan iterasi dan penyesuaian jika diperlukan berdasarkan hasil observasi.

FSM dalam penelitian ini dirancang dengan beberapa *state* dasar untuk NPC musuh, seperti *Idle*, *Patrol*, *Chase*, *Attack*, *Dodge*, *Take Damage*, dan *Dead*. Data yang digunakan dalam penelitian ini adalah hasil implementasi FSM pada NPC musuh dalam game *Lily's Dreamland: Chaos in the Storybook*. Data diperoleh melalui observasi perilaku NPC musuh. Adapun alat dan teknologi yang digunakan dalam penelitian ini adalah:

A. Unity Engine

Unity Engine merupakan aplikasi pengembang game lintas *platform* yang dirancang untuk kemudahan penggunaan. Unity mendukung grafis 3D dan 2D. Bahasa pemrograman yang digunakan pada Unity yaitu bahasa C-Sharp (C#). Selain itu Unity juga bisa digunakan secara gratis maupun berbayar, dokumentasi atau tutorial yang sudah banyak membuat Unity mudah dipelajari [16].

B. Visual Studio Code

Visual Studio Code merupakan aplikasi editor *code*, yang kompatibel dengan sistem operasi Windows, macOS, dan Linux. Visual Studio Code mendukung berbagai bahasa pemrograman seperti Javascript, PHP, C, Go, dll, selain itu Visual Studio Code juga memiliki *library extension* yang banyak sehingga memudahkan para *developer* melakukan pekerjaannya [17].

C. Aseprite

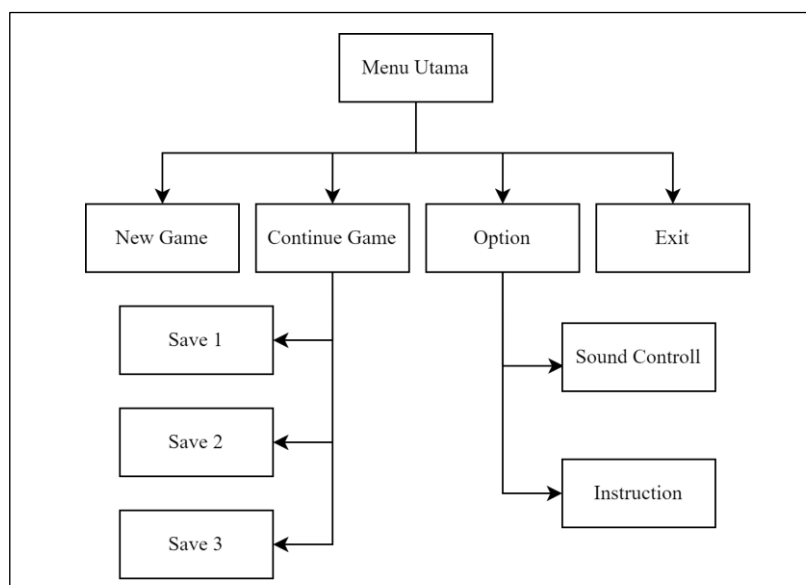
Aseprite merupakan aplikasi editor sprite animasi dan alat seni piksel yang tersedia untuk Windows, Mac, dan Linux, dikembangkan oleh Igar Studio. Ada berbagai alat gambar dan manipulasi piksel dalam Aseprite ini, sehingga memudahkan dalam pembuatan seni piksel [18].

D. C#

C# (dibaca C Sharp) merupakan bahasa pemrograman berorientasi objek yang dikembangkan oleh Microsoft. Bahasa ini dibangun berdasarkan C++ dan dipengaruhi oleh berbagai aspek dan fitur dari bahasa pemrograman lain seperti Java, Delphi, Visual Basic, dan lainnya, dengan beberapa penyederhanaan. Dalam Unity, C# digunakan untuk *scripting* dan sangat mudah digunakan untuk rotasi dan skala objek hanya dengan sebaris kode. Begitu pula dengan duplikasi, penghapusan, dan perubahan properti [19].

Perancangan

A. Struktur Navigasi Game



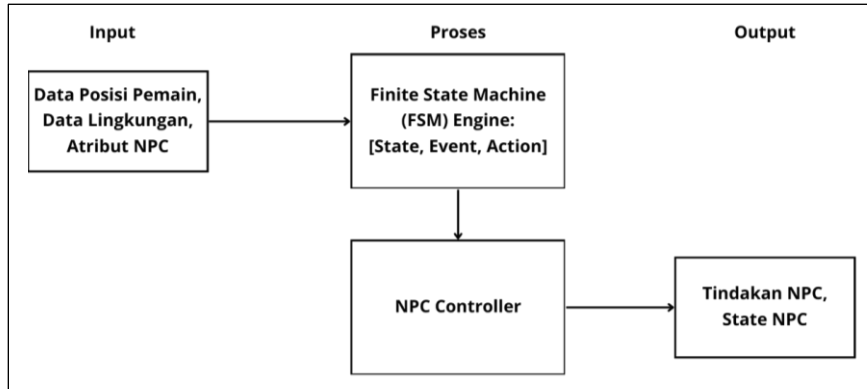
Gambar 2. Struktur Navigasi Game

Gambar 2 menunjukkan struktur navigasi menu utama game *Lily's Dreamland: Chaos In the Storybook*. Pada menu utama terdapat empat pilihan yaitu:

1. *New Game*, yaitu pilihan untuk memulai permainan baru dari awal.

2. *Continue Game*, yaitu pilihan untuk melanjutkan permainan yang sudah tersimpan dari *slot save* yang tersedia
3. *Option*, yaitu pilihan untuk menuju pada sub-menu “Sound Control”, yang berfungsi untuk mengatur suara pada game dan juga sub-menu “Instruction” untuk melihat petunjuk permainan.
4. *Exit*, yaitu pilihan untuk keluar dari game.

B. Perancangan Input/Output

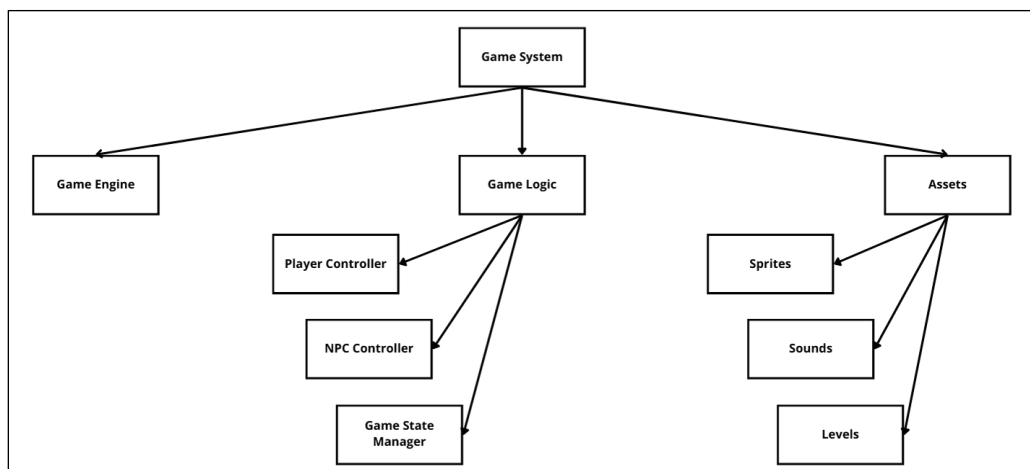


Gambar 3. Diagram Input/Output

Gambar 3 menunjukkan bagaimana sebuah NPC dalam sebuah permainan memproses data masukan dan menghasilkan tindakan keluaran berdasarkan *state* yang dimilikinya. Deskripsinya sebagai berikut:

1. Input
 - a. Data posisi pemain (koordinat x, y)
 - b. Data lingkungan (tata letak platform, objek sekitar)
 - c. Atribut NPC (kecepatan, kekuatan, radius deteksi)
2. Proses
 - a. Pengembalian keputusan berdasarkan input
 - b. Transisi antar *state* FSM
 - c. Penentuan tindakan berdasarkan *state* dan *event*
 - d. Menggunakan keputusan dari FSM *engine* untuk memperbarui perilaku NPC
3. Output
 - a. Tindakan NPC, hasil akhir dari proses FSM dan NPC Controller, seperti bergerak, menyerang, atau menghindar.
 - b. Status NPC, *State* NPC setelah proses pengambilan keputusan oleh FSM *engine*.

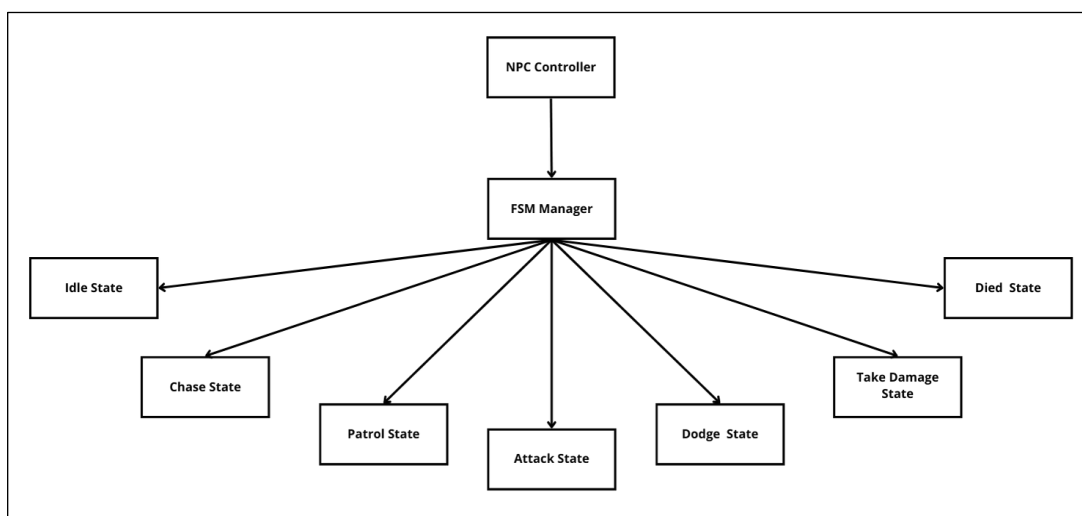
C. Arsitektur Sistem



Gambar 4. Diagram Arsitektur Sistem Game

Gambar 4 menunjukkan diagram arsitektur sistem game *Lily's Dreamland: Chaos in the Storybook*, diagram ini menunjukkan hubungan antara berbagai komponen utama dalam game seperti Game Engine, Game Logic, dan Assets, serta sub komponen di bawahnya. Deskripsinya sebagai berikut berikut:

1. *Game Engine*, menyediakan layanan inti seperti rendering, *physics*, audio, dll.
2. *Game Logic*, berisi mekanisme permainan, adapun sub komponennya sebagai berikut:
 - a. *Player Controller*, mengelola input dan aksi pemain
 - b. *NPC Controller*, mengelola perilaku NPC
 - c. *Game State Manager*, mengelola *state* game seperti menu, *gameplay*, *pause*, dll.
 - d. *Assets*, berisi konten yang ada dalam game, adapun sub-komponennya sebagai berikut:
 - e. *Sprites*, berisi gambar dan animasi karakter, dan objek.
 - f. *Sounds*, berisi efek suara dan musik latar
 - g. *Levels*, berisi data level dan peta permainan



Gambar 5. Diagram Arsitektur Sistem FSM


Gambar 5 menunjukkan diagram arsitektur sistem FSM pada NPC Musuh dalam game *Lily's Dreamland: Chaos in the Storybook*, diagram ini menunjukkan berbagai *state* yang dikelola oleh *FSM Manager*.





Keterangan:

1. *Idle State*, NPC tidak melakukan apa-apa, hanya diam ditempat.
2. *Chase State*, NPC mengejar pemain yang terdeteksi.
3. *Patrol State*, NPC melakukan patroli di area yang ditentukan.
4. *Attack State*, NPC menyerang pemain.
5. *Dodge State*, NPC menghindari serangan dari pemain.
6. *Take Damage State*, NPC terkena serangan dari pemain.
7. *Dead State*, NPC mati dan tidak melakukan aktivitas

D. Desain Karakter

Tabel 1. Desain Karakter

No.	Gambar Karakter	Keterangan
1		Karakter utama dalam game, bernama Lily. Karakteristik di darat dan tipe serangan jarak dekat

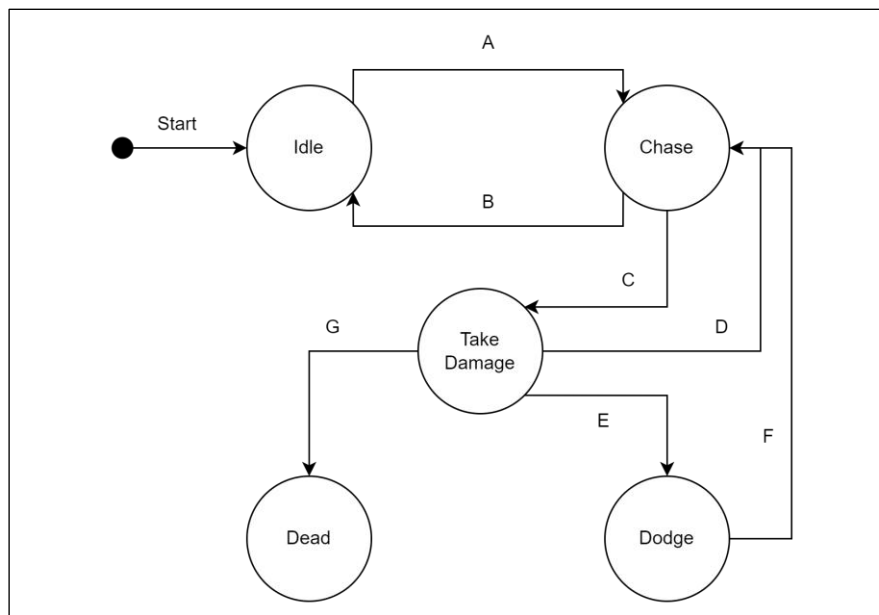
2		Karakter NPC musuh dalam game, bernama Slime. Karakteristik di darat dan tipe serangan jarak dekat.
3		Karakter NPC musuh dalam game, bernama Plant. Karakteristik di darat dan tipe serangan jarak jauh
4		Karakter NPC musuh dalam game, bernama Chicken. Karakteristik di darat dan tipe serangan jarak dekat
5		Karakter NPC musuh dalam game, bernama Bat. Karakteristik di udara dan tipe serangan jarak dekat

Tabel 1, merupakan karakter-karakter yang terdapat pada game *Lily's Dreamland: Chaos in the Storybook*. Terdapat 4 karakter yang merupakan NPC musuh, karakter-karakter inilah yang akan diterapkan FSM. Sebagian desain karakter pada game ini menggunakan aset yang diunduh dari internet dengan lisensi bebas. Aset-aset tersebut diperoleh dari situs [20].

Pemodelan

A. Perancangan FSM pada NPC Musuh

1. Perancangan FSM pada NPC Musuh 1 (Slime)

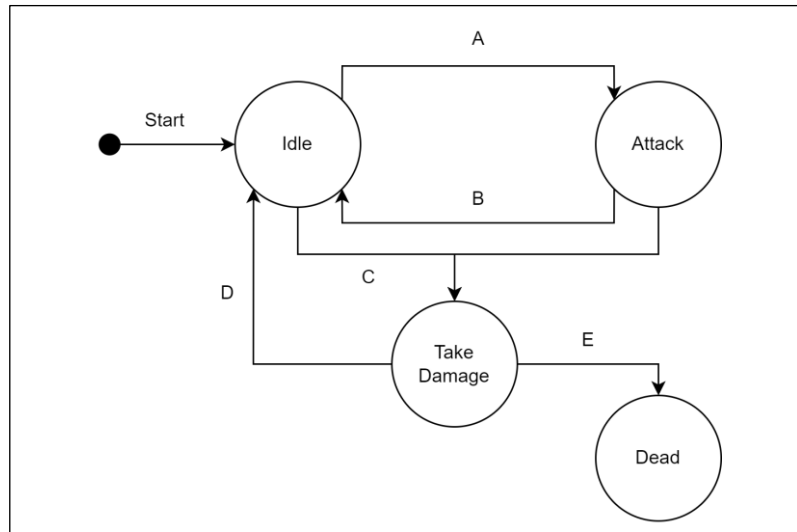


Gambar 6. Diagram State FSM NPC Musuh 1

Keterangan:

- a. A = Pemain berada dalam radius deteksi NPC musuh
- b. B = Pemain tidak berada dalam radius deteksi NPC musuh
- c. C = NPC musuh menerima serangan dari pemain
- d. D = Health poin NPC musuh lebih dari 0
- e. E = NPC musuh menghindari serangan yang diterima dari pemain
- f. F = Health poin NPC musuh lebih dari 0
- g. G = Health poin NPC musuh kurang atau sama dengan 0

2. Perancangan FSM pada NPC Musuh 2 (Plant)

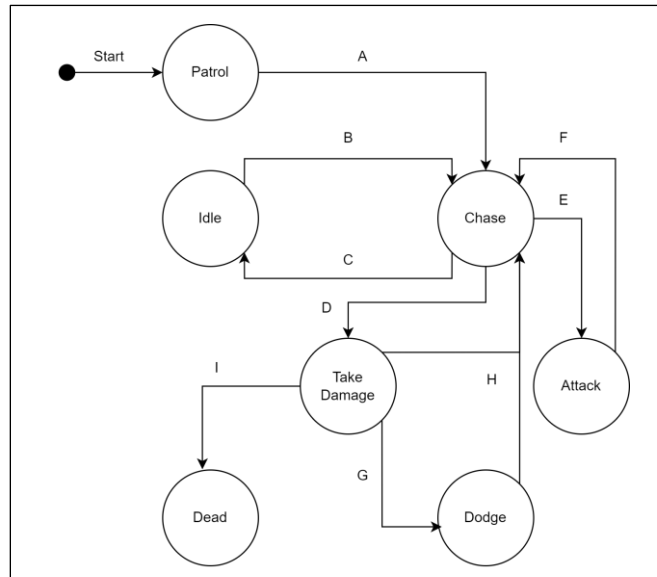


Gambar 7. Diagram State FSM NPC Musuh 2

Keterangan:

- a. A = Pemain berada dalam radius deteksi NPC musuh
- b. B = Pemain tidak berada dalam radius deteksi NPC musuh
- c. C = NPC musuh menerima serangan dari pemain
- d. D = *Health* poin NPC musuh lebih dari 0
- e. E = *Health* poin NPC musuh kurang atau sama dengan 0

3. Perancangan FSM pada NPC Musuh 3 (Chicken)



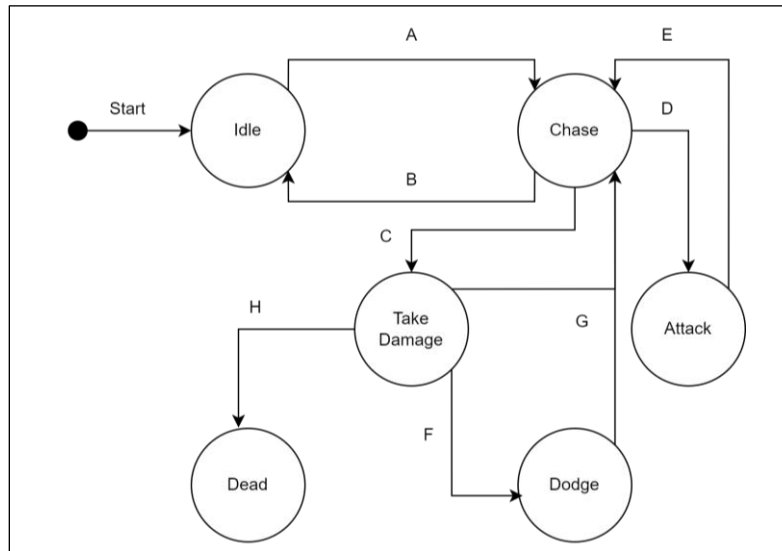
Gambar 8. Diagram State FSM NPC Musuh 3

Keterangan:

- a. A = Pemain berada dalam radius deteksi NPC musuh
- b. B = Pemain berada dalam radius deteksi NPC musuh
- c. C = Pemain tidak berada dalam radius deteksi NPC musuh
- d. D = NPC musuh menerima serangan dari pemain
- e. E = Pemain berada dalam radius serangan NPC musuh
- f. F = Pemain tidak berada dalam radius serangan NPC musuh
- g. G = NPC musuh menghindari serangan dari pemain

- h. H = Health poin NPC musuh lebih dari 0
- i. I = Health poin NPC musuh kurang atau sama dengan 0

4. Perancangan FSM pada NPC Musuh 4 (Bat)



Gambar 9. Diagram *State* FSM NPC Musuh 4

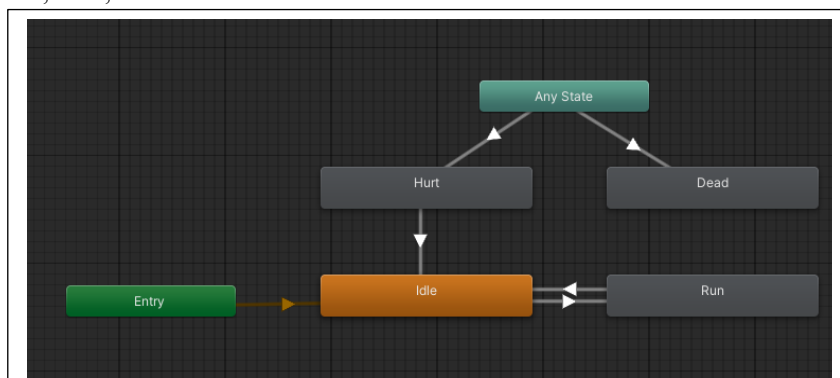
Keterangan:

- a. A = Pemain berada dalam radius deteksi NPC musuh
- b. B = Pemain tidak berada dalam radius deteksi NPC musuh
- c. C = NPC musuh menerima serangan dari pemain
- d. D = Pemain berada dalam radius serangan NPC musuh
- e. E = Pemain tidak berada dalam radius serangan NPC musuh
- f. F = NPC musuh menghindari serangan dari pemain
- g. G = Health poin NPC musuh lebih dari 0
- h. H = Health poin NPC musuh kurang atau sama dengan 0

B. Penerapan FSM pada NPC Musuh

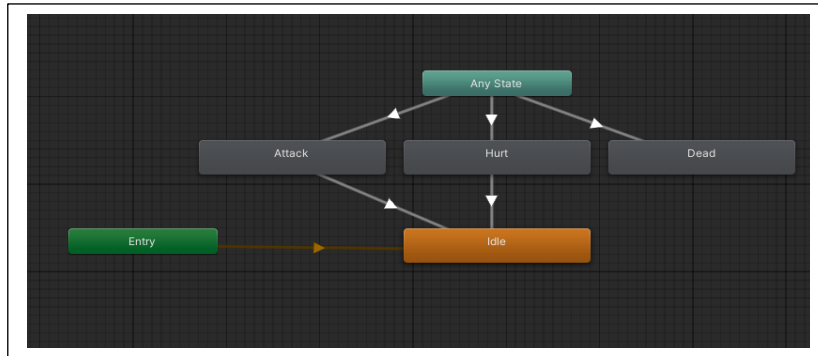
1. Penerapan FSM pada NPC Musuh 1 (Slime)

Gambar 10 merupakan penerapan FSM pada animasi NPC Slime, pada NPC Slime terdapat animasi Idle, Run, Hurt, dan Dead.



Gambar 10. Penerapan FSM pada NPC Musuh 1 di Unity

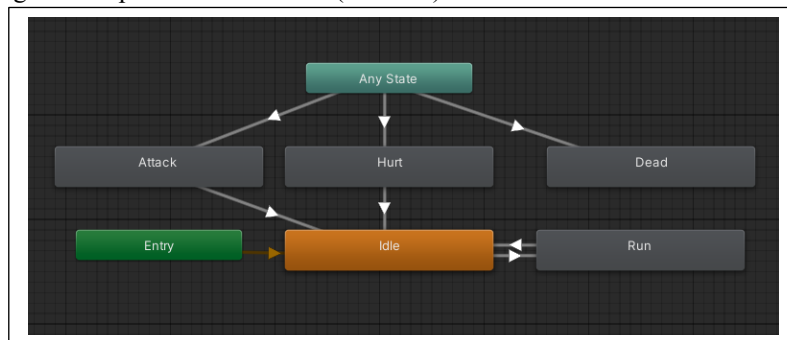
2. Perancangan FSM pada NPC Musuh 2 (Plant)



Gambar 11. Penerapan FSM pada NPC Musuh 2 di Unity

Gambar 11 merupakan penerapan FSM pada animasi NPC Plant, pada NPC Plant terdapat animasi *Idle*, *Attack*, *Hurt*, dan *Dead*.

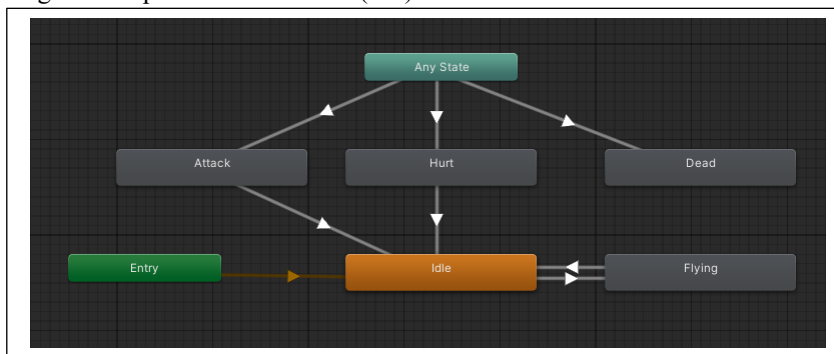
3. Perancangan FSM pada NPC Musuh 3 (Chicken)



Gambar 12. Penerapan FSM pada NPC Musuh 3 di Unity

Gambar 12 merupakan penerapan FSM pada animasi NPC Chicken, pada NPC Chicken terdapat animasi *Idle*, *Run*, *Attack*, *Hurt*, dan *Dead*.

4. Perancangan FSM pada NPC Musuh 4 (Bat)



Gambar 13. Penerapan FSM pada NPC Musuh 4 di Unity

Gambar 13 merupakan penerapan FSM pada animasi NPC Bat, pada NPC Bat terdapat animasi *Idle*, *Flying*, *Attack*, *Hurt*, dan *Dead*.

C. Pengujian FSM pada NPC Musuh

1. Pengujian FSM pada NPC Musuh 1 (Slime)

Tabel 2. Pengujian FSM pada NPC Slime

No.	State	Kondisi	Hasil
1	Idle	Jika pemain di luar jarak deteksi musuh, maka musuh akan diam.	Sesuai

2	Chase	Jika pemain berada di dalam jarak deteksi musuh, maka musuh akan mengejar pemain.	Sesuai
3	Take Damage	Jika musuh menerima serangan dari pemain, maka musuh akan terluka.	Sesuai
4	Dodge	Jika musuh dapat menghindari serangan dari pemain, maka musuh tidak akan terluka.	Sesuai
5	Dead	Jika musuh menerima serangan dari pemain dan <i>Health</i> poinnya habis, maka musuh akan mati.	Sesuai

Pengujian FSM pada NPC Slime menunjukkan hasil yang sesuai pada semua *state*. Hal ini menunjukkan bahwa penerapan FSM pada NPC Slime bekerja sesuai yang diinginkan.

2. Pengujian FSM pada NPC Musuh 2 (Plant)

Tabel 3. Pengujian FSM pada NPC Plant

No.	State	Kondisi	Hasil
1	Idle	Jika pemain di luar jarak deteksi musuh, maka musuh akan diam.	Sesuai
2	Attack	Jika pemain berada dalam jarak serang musuh, maka musuh akan menyerang pemain.	Sesuai
3	Take Damage	Jika musuh menerima serangan dari pemain, maka musuh akan terluka.	Sesuai
4	Dead	Jika musuh menerima serangan dari pemain dan <i>Health</i> poinnya habis, maka musuh akan mati.	Sesuai

Pengujian FSM pada NPC Plant menunjukkan hasil yang sesuai pada semua *state*. Hal ini menunjukkan bahwa penerapan FSM pada NPC Plant bekerja sesuai yang diinginkan.

3. Pengujian FSM pada NPC Musuh 3 (Chicken)

Tabel 4. Pengujian FSM pada NPC Chicken

No.	State	Kondisi	Hasil
1	Idle	Jika pemain di luar jarak deteksi musuh, maka musuh akan diam.	Sesuai
2	Chase	Jika pemain berada di dalam jarak deteksi musuh, maka musuh akan mengejar pemain.	Sesuai
3	Attack	Jika pemain berada dalam jarak serang musuh, maka musuh akan menyerang pemain.	Sesuai
4	Take Damage	Jika musuh menerima serangan dari pemain, maka musuh akan terluka.	Sesuai
5	Dead	Jika musuh menerima serangan dari pemain dan <i>Health</i> poinnya habis, maka musuh akan mati.	Sesuai
6	Dodge	Jika musuh dapat menghindari serangan dari pemain, maka musuh tidak akan terluka.	Sesuai
7	Patrol	Jika game baru dimulai, maka musuh akan berpatroli pada titik tertentu	Sesuai

Pengujian FSM pada NPC Chicken menunjukkan hasil yang sesuai pada semua *state*. Hal ini menunjukkan bahwa penerapan FSM pada NPC Chicken bekerja sesuai yang diinginkan.

4. Pengujian FSM pada NPC Musuh 4 (Bat)

Tabel 5. Pengujian FSM pada NPC Bat

No.	State	Kondisi	Hasil
1	Idle	Jika pemain di luar jarak deteksi musuh, maka musuh akan diam.	Sesuai
2	Chase	Jika pemain berada di dalam jarak deteksi musuh, maka musuh akan mengejar pemain.	Sesuai
3	Attack	Jika pemain berada dalam jarak serang musuh, maka musuh akan menyerang pemain.	Sesuai

4	Take Damage	Jika musuh menerima serangan dari pemain, maka musuh akan terluka.	Sesuai
5	Dead	Jika musuh menerima serangan dari pemain dan <i>Health</i> poinnya habis, maka musuh akan mati.	Sesuai
6	Dodge	Jika musuh dapat menghindari serangan dari pemain, maka musuh tidak akan terluka.	Sesuai

Pengujian FSM pada NPC Bat menunjukkan hasil yang sesuai pada semua *state*. Hal ini menunjukkan bahwa penerapan FSM pada NPC Bat bekerja sesuai yang diinginkan.

D. Perhitungan Manual

Perhitungan manual dilakukan untuk menunjukkan bagaimana NPC musuh menggunakan FSM untuk beralih antar *state* yang berbeda berdasarkan *input* dari lingkungan atau pemain. Berikut merupakan contoh, bagaimana menghitung jarak antara pemain dan NPC musuh untuk menentukan apakah NPC musuh harus beralih dari *state* Idle ke *state* Chase. Dalam hal ini akan digunakan rumus menghitung jarak Euclidean:

$$d = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2} \tag{1}$$

Keterangan:

- d = jarak antara pemain dan NPC musuh
- x_1 = posisi/koordinat horizontal dari pemain
- x_2 = posisi/koordinat horizontal dari NPC musuh
- y_1 = posisi/koordinat vertikal dari pemain
- y_2 = posisi/koordinat vertikal dari NPC musuh

Diketahui:

- Posisi pemain : $(x_1, y_1) = (5px, 10px)$
- Posisi NPC musuh : $(x_2, y_2) = (2px, 4px)$

Perhitungan:

$$d = \sqrt{(2 - 5)^2 + (4 - 10)^2}$$

$$d = \sqrt{(-3)^2 + (-6)^2}$$

$$d = \sqrt{9 + 36}$$

$$d = \sqrt{45}$$

$$d = 6.7px$$

Jika radius deteksi NPC musuh adalah 10px, dan berdasarkan perhitungan di atas diketahui jarak antara NPC musuh dan pemain adalah 6.7px, maka NPC musuh akan beralih dari *state* Idle ke *state* Chase, karena posisi pemain berada dalam radius deteksi NPC musuh.

Kesimpulan

Hasil pengujian terhadap penerapan Finite State Machine (FSM) pada NPC musuh dalam game *Lily's Dreamland: Chaos in the Storybook*, menunjukkan bahwa FSM berhasil diimplementasikan dengan baik. Setiap *state* pada NPC musuh, seperti Slime, Plant, Chicken, dan Bat, berfungsi sesuai dengan yang diharapkan, tanpa adanya kesalahan atau malfungsi. Hal ini menegaskan bahwa FSM adalah metode yang efektif untuk mengatur perilaku NPC dalam game ini. Meskipun pengujian ini memastikan bahwa FSM bekerja dengan baik dalam konteks pengaturan mekanik NPC musuh, evaluasi lebih lanjut diperlukan untuk memahami dampaknya terhadap keseluruhan pengalaman bermain dan dinamika interaksi dalam permainan.

Daftar Pustaka

- [1] A. Pranselga, I. R. Setiawan, dan W. Apriandari, "Implementasi Finite State Machine pada Karakter NPC Musuh dalam Game Adventure In Java," *Jurnal Ilmiah teknik Informatika*, vol. 10, no. 3, 2021.
- [2] S. Syarif, T. Hasanuddin, dan M. Hasnawi, "Perancangan Game Puzzle Labirin Menggunakan Metode Game Development Life Cycle (GDLC) Berbasis Unreal Engine," *Buletin Sistem Informasi dan Teknologi Islam*, vol. 3, no. 1, hlm. 34–41, Feb 2022.
- [3] B. A. Putro, K. A. Sari, dan A. Wahid, "Penerapan Metode Finite State Machine pada Game Escape FROM Punk Hazard," *Jurnal Mahasiswa Teknik Informatika*, vol. 5, no. 1, 2021.
- [4] A. Novanto dan M. Rizqi, "Desain Game Mekanik Interaktif Antar Karakter Dengan Kuda pada Game," *SIMKOM*, vol. 8, no. 2, hlm. 137–149, Jul 2023, doi: 10.51717/simkom.v8i2.238.
- [5] A. Enggar, P. D. Kusuma, dan R. Astuti, "Implementasi Finite State Machine Untuk Npc Pada Game 2d Side-Scroll Shooter Implementation Of Finite-State Machine For Npc In 2d Side Scroll Shooter," dalam *e-Proceeding of Engineering*, 2022, hlm. 1080–1086.
- [6] R. Amalia, "Game Edukasi dan Cerita Interaktif Sejarah Kerajaan di Sumatra Menggunakan Algoritma Fuzzy Sugeno Untuk Mengatur Perilaku NPC," *Jurnal Informatika dan Rekayasa Perangkat Lunak (JATIKA)*, vol. 1, no. 2, hlm. 192–202, 2020.
- [7] T. Hidayat, A. Yuda Pratama, Y. Astuti, dan D. Maulina, "Rancang Bangun Kecerdasan Buatan untuk Non Player Character pada Game Platform Android," Yogyakarta, 2019.
- [8] F. Rahmat Muhammad, E. Wahyu Hidayat, dan M. Adi Khairul Anshary, "Rancang Bangun Game Side Scroller Kopasus Mission Berbasis 2D Platformer pada Perangkat Android Korespondensi," *Scientific Articles of Informatics Students*, vol. 2, no. 1, hlm. 69–75, 2019.
- [9] W. Safitra, A. Faisol, dan S. A. Wibowo, "Penerapan Metode Finite State Machine pada Non Player Character (NPC) Game Action Strategy 'Ouroboros,'" *Jurnal Mahasiswa Teknik Informatika*, vol. 4, no. 2, 2020.
- [10] M. F. Al Kautsar, H. Haryanto, E. Z. Astuti, E. Mulyanto, U. Rosyidah, dan A. Kardianawati, "Penerapan Finite State Machine dalam Perancangan Perilaku Musuh pada Game 2D Platformer bertema Sejarah," *Jurnal Ilmiah Teknologi - Informasi & Sains*, vol. 14, no. 2, hlm. 236–243, Jul 2024, doi: 10.36350/jbs.v14i2.
- [11] A. Nopriansyah, I. Kanedi, dan Prahasti, "Rancang Bangun Game 2D Shooter Menggunakan Metode Finite State Machine," *Jurnal Sains, Teknologi dan Industri*, vol. 19, no. 2, hlm. 171–177, Jun 2022.
- [12] Asrianda dan Zulfadli, "Konsep Finite State Machine dan implementasinya pada Game," *Jurnal Sistem Informasi*, vol. 6, no. 1, 2022.
- [13] H. Bening Abimanyu, S. Achmadi, dan Ariwibisono, "Rancang Bangun Game 'War of Aliens Wanokuni' Menerapkan Metode FSM (Finite State Machine)," *JATI (Jurnal Mahasiswa Teknik Informatika)*, vol. 5, no. 2, hlm. 480–486, Okt 2021, doi: 10.36040/jati.v5i2.3753.
- [14] R. K. Saputra dan K. Setiawan, "Pengembangan Game 2D Platformer Berbasis Microbit Menggunakan Unity," *Jurnal Indonesia : Manajemen Informatika dan Komunikasi*, vol. 5, no. 1, hlm. 20–25, Jan 2024, doi: 10.35870/jimik.v5i1.420.
- [15] G. Mau, "Rancang Bangun Game 2D Shooter Platformer Menggunakan Metode Finite State Machine," *Jurnal Mahasiswa Teknik Informatika*, vol. 3, no. 1, 2019.
- [16] M. G. Wellson dan W. T. Atmojo, "Implementasi Metode GDLC pada Game Taxi Rush Menggunakan Unity Engine," *Jurnal Teknoinfo*, vol. 18, no. 1, hlm. 201–214, 2024.
- [17] M. Kurniawan dan B. Kurniawan, "Implementasi Pemrograman Python Menggunakan Visual Studio Code," *JIK: Jurnal Informatika Dan Komputer*, vol. 11, no. 2, hlm. 1–9, 2020.
- [18] M. T. Kallang, "Penerapan Algoritma A Star dan Upper Confidence Bound pada Game Maze Roman Numerals Berbasis Android," Universitas Komputer Indonesia, Bandung, 2020.
- [19] W. J. Mekel, S. R. U. A. Sompie, dan B. A. Sugiarto, "Rancang Bangun Game 3D Pertahanan Kerajaan Bowontehu," *Jurnal Teknik Informatika*, vol. 14, no. 4, hlm. 455–464, 2019.
- [20] Pixel Frog, "Pixel Adventure," Itch.io. Diakses: 4 Juni 2024. [Daring]. Tersedia pada: <https://pixelfrog-assets.itch.io/pixel-adventure-1>