JURNAL ILMU KOMPUTER
KHARISMA .TECH

Jurnal KHARISMA Tech | ISSN:1907-2317 | e-ISSN: 2810-0344
https://jurnal.kharisma.ac.id/kharismatech
published by: Pusat Penelitian STMIK KHARISMA Makassar

Volume: 20 no. 02 – Oktober 2025- hlm. 25-34

IMPLEMENTASI *FINITE STATE MACHINE* UNTUK MENGATUR ALUR NARASI *GAME VISUAL NOVEL* MYSTIC PAGES

Oleh:

Clarissa Yason[1], Mohammad Fajar[2*], Baizul Zaman[3]
[1,2,3] Teknik Informatika, STMIK Kharisma Makassar
e-mail: [1]clarissayason_22@kharisma.ac.id, [2] fajar@kharisma.ac.id, [3] baizul@kharisma.ac.id

Abstrak: *Visual novel merupakan salah satu genre video game yang menitikberatkan pada penyampaian narasi interaktif, di mana pemain terlibat dalam alur cerita melalui serangkaian pilihan yang memengaruhi arah dan akhir cerita. Dalam pengembangan game Mystic Pages, sebuah visual novel bertema misteri, menghadapi tantangan pada pengelolaan cerita bercabang. Penggunaan metode seperti struktur if-else dan sistem flag dinilai tidak efisien dan rentan terhadap kesalahan logika dalam skala besar. Oleh karena itu, penelitian ini bertujuan untuk menerapkan Finite State Machine (FSM) sebagai solusi untuk mengelola alur narasi yang dinamis dan terstruktur. Skenario cerita pada novel direpresentasikan sebagai state-state dalam diagram FSM, dimulai dari state "start dialog" hingga state "dialog pack end 5", dengan jalur cerita yang berkembang melalui transisi berdasarkan pilihan pemain. Hasil pengujian menunjukkan bahwa dengan adanya dokumentasi dan visualisasi alur cerita Mystic Page dalam bentuk diagram FSM, proses modifikasi dapat dilakukan dengan mudah. Pengembang hanya perlu menambahkan state dan transisi baru ke dalam struktur FSM tanpa mengganggu logika yang telah ada sebelumnya. Selain itu, pengujian menggunakan metode black-box testing menunjukkan bahwa seluruh transisi antar state berjalan dengan baik, tanpa kendala seperti dead-end atau infinite loop dalam alur cerita. Semua percabangan berhasil dijalankan sesuai dengan rancangan, dengan tingkat keberhasilan mencapai 100%.*

Kata kunci: *Visual novel, Mystic Pages, Finite State Machine, Ren'Py, Cerita Bercabang*

Abstract: *Visual novels are a genre of video games that emphasized the delivery of interactive narratives, where players engaged with the storyline through a series of choices that influenced its direction and outcomes. In the development of Mystic Pages, a mystery-themed visual novel, challenges were encountered in managing branching storylines. Conventional methods such as if-else structures and flag systems were found to be inefficient and prone to logical errors when applied to large-scale scenarios. Therefore, this study aimed to implement a Finite State Machine (FSM) as a solution to manage dynamic and structured narrative flows. The storyline was represented as states within an FSM diagram, starting from the "start dialog" state to the "dialog pack end 5" state, with narrative progression determined by player choices. The results demonstrated that the documentation and visualization of Mystic Pages' branching storylines using FSM facilitated modifications, as developers only needed to add new states and transitions without disrupting existing logic. Furthermore, black-box testing confirmed that all transitions between states operated correctly, with no issues such as dead ends or infinite loops. All branches executed as designed, achieving a 100% success rate.*

Keywords: *Visual novel, Mystic Pages, Finite State Machine, Ren'Py, Branching Narrative*

[*] Corresponding author : Mohammad Fajar (fajar@kharisma.ac.id)

Diterima : *Agustus, 2025* ***Disetujui :*** *Agustus, 2025* ***Dipublikasikan :*** *Oktober, 2025*

1. PENDAHULUAN

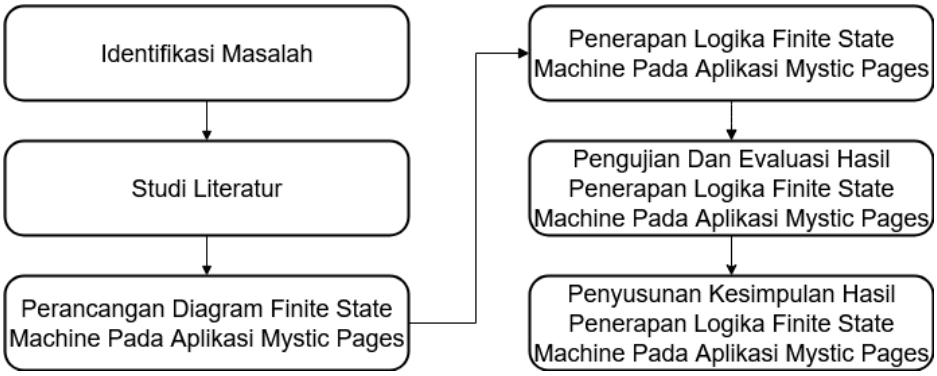
Visual novel merupakan salah satu genre video game yang mengutamakan narasi sebagai elemen utama, di mana pemain terlibat langsung melalui pilihan-pilihan yang memengaruhi arah cerita dan menghasilkan beragam akhir yang berbeda, dan menciptakan percabangan narasi yang saling berhubungan [1], [2], . Salah satu *visual novel* yang sedang dikembangkan adalah Mystic Pages, sebuah game dengan latar suasana misteri yang membawa pemain menjalani kisah petualangan berkemah bersama tiga sahabatnya. Pengelolaan cerita bercabang pada aplikasi *visual novel* Mystic Pages menghadirkan sejumlah tantangan [3], terutama terkait kompleksitas logika serta konsistensi antarjalur cerita yang semakin sulit dikelola apabila menggunakan pendekatan manual [4]. Secara umum, beberapa metode telah digunakan dalam pengaturan alur cerita *visual novel*. Metode paling sederhana adalah struktur *if-else*, meski mudah diimplementasikan, pendekatan ini menjadi tidak efisien dan sulit dipelihara dalam skenario berskala besar karena meningkatnya kompleksitas logika dan potensi kesalahan [5]. Pendekatan lain adalah sistem *flag*, sistem ini memberikan fleksibilitas lebih dibanding *if-else*, namun masih menyulitkan dalam pelacakan dependensi dan rentan terhadap konflik logika. Selain itu, *decision tree* atau pohon keputusan juga dapat digunakan sebagai visualisasi jalur cerita yang sistematis. Namun, decision tree memiliki kelemahan dalam hal fleksibilitas dan skalabilitas, karena perubahan kecil pada narasi bisa berdampak pada keseluruhan struktur pohon yang telah dibangun [6].

Dibandingkan dengan metode-metode tersebut, pendekatan *Finite State Machine* (FSM) menawarkan pendekatan yang lebih terstruktur dan efisien. FSM adalah model komputasi berbasis *state* dimana setiap *state* yang berbeda saling terhubung berdasarkan suatu kondisi tertentu [7]. Dalam game *visual novel*, FSM dapat digunakan untuk memetakan setiap cabang cerita yang direpresentasikan sebagai status dalam mesin tersebut, dengan transisi yang ditetapkan berdasarkan pilihan pengguna. Pendekatan ini membantu mengurangi risiko kesalahan logika dan mempermudah proses pemeliharaan serta modifikasi alur cerita di masa mendatang [8], [9]. Sejumlah penelitian terdahulu telah menunjukkan efektivitas FSM dalam pengembangan game, baik untuk pengelolaan interaksi dialog [10], peningkatan dinamika gameplay edukasi [11], penyederhanaan alur cerita yang kompleks dalam game berbasis sejarah atau misteri [3], [12]. Adapun perbedaan penelitian ini dengan sejumlah studi tersebut terletak pada kasus game yang dikembangkan, yang tentunya memiliki karateristik-karateristik alur narasi yang berbeda. Sehingga, ini menunjukkan adanya peluang untuk mengaplikasikan pendekatan serupa pada pengembangan Mystic Pages.

Oleh karena itu, penelitian ini bertujuan untuk mengimplementasikan FSM sebagai kerangka logika dalam mengatur alur cerita bercabang pada Mystic Pages. Pemakaian pendekatan FSM ini diharapkan dapat menawarkan kejelasan struktur bagi pengembang, tetapi juga memastikan pengalaman pemain tetap imersif tanpa terganggu oleh masalah teknis. Dengan pengujian black-box testing, validitas transisi antar *state* dapat dipastikan berjalan sesuai dengan desain naratif yang direncanakan.

2. METODE PENELITIAN

Penelitian dimulai dengan identifikasi masalah untuk memahami kendala pengelolaan alur bercabang pada Mystic Pages. Selanjutnya dilakukan studi literatur mencakup kajian terkait metode pengaturan alur cerita *visual novel* serta penerapan *finite state machine* (FSM) dalam *game*. Peneliti lalu merancang model alur cerita Mystic Pages menggunakan diagram FSM. Rancangan tersebut kemudian diimplementasikan ke dalam aplikasi Mystic Pages menggunakan *engine* Ren'Py berbasis Python. Pengujian dan evaluasi dilakukan untuk memastikan penerapan telah berjalan dengan baik, hasil pengujian lalu dievalusi untuk digunakan dalam pengambilan kesimpulan terkait hasil penelitian. Kesimpulan kemudian disusun dengan mengacu pada hasil analisis, perancangan, implementasi, dan pengujian sistem yang telah dilakukan sebelumnya. Tahapan-tahapan penelitian yang akan dilakukan pada penelitian ini disajikan pada Gambar 1.



Gambar 1. Tahapan Penelitian

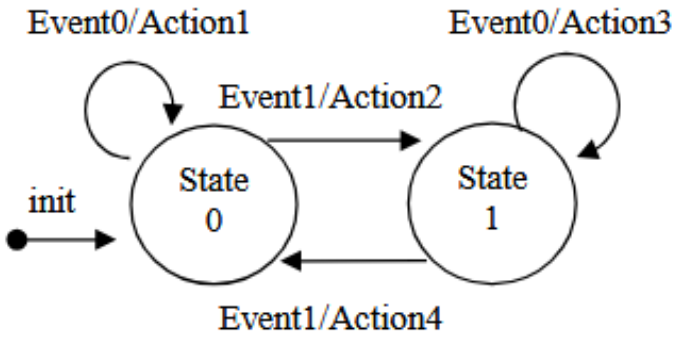
Pengujian sistem dilakukan dengan metode *black-box testing* untuk memverifikasi ketepatan transisi antar*state* dan konsistensi jalur cerita. Proses pengumpulan data dilakukan melalui pencatatan setiap transisi, keberhasilan jalur percabangan, serta validasi terhadap potensi kesalahan logika seperti *dead end* atau *infinite loop*. Data hasil pengujian kemudian dianalisis secara deskriptif untuk menilai efektivitas penerapan FSM dalam mengelola alur cerita bercabang.

2.1. Finite State Machine

Finite State Machine (FSM) adalah model komputasi yang digunakan untuk memodelkan sistem yang memiliki sejumlah status terbatas dengan transisi antar status berdasarkan kondisi tertentu [7]. FSM menggambarkan tingkah laku sistem melalui 3 hal yaitu, *state* (keadaan), *event* (kejadian), dan *action* (aksi) [13]. Pada satu saat dalam periode waktu yang cukup signifikan, sistem akan berada dalam *state* yang aktif. Sistem dapat beralih dari satu *state* ke *state* lainnya ketika menerima sebuah masukan, baik dari luar maupun dari dalam komponen sistem itu sendiri, namun mesin ini hanya dapat berada pada satu *state* pada satu waktu saja [14].

Untuk menggambarkan *Finite State Machine* (FSM), digunakanan simbol-simbol khusus untuk merepresentasikan berbagai komponen dalam diagramnya. *State* (keadaan)

digambarkan sebagai lingkaran atau elips yang mewakili berbagai kondisi operasional mesin, dan biasanya diberi nama unik untuk memudahkan identifikasi. Peralihan antar-*state* (transisi) ditunjukkan dengan panah berlabel di mana labelnya menunjukkan *input* atau *event* yang memicu perpindahan *state*. Selain itu, *initial state* (keadaan awal) sebagai titik awal operasi mesin, sering ditandai dengan panah pendek menuju *state* pertama. Dalam beberapa representasi, *final state* (keadaan akhir), yang menandakan akhir proses atau komputasi, digambarkan sebagai lingkaran ganda atau lingkaran dengan titik di tengah. Contoh diagram *state* FSM sederhana diperlihatkan pada Gambar 2.



Gambar 2. Diagram Finite State Machine

3. HASIL DAN PEMBAHASAN

3.1. Analisis Sistem Mystic Pages

Aplikasi Mystic Pages menyajikan pengalaman bermain berbasis narasi bercabang yang menarik. Pemain akan memerankan karakter utama dalam sebuah cerita misteri yang melibatkan elemen supranatural, di mana pemain akan diberi pilihan-pilihan keputusan yang dapat memengaruhi jalannya permainan dan akhir cerita yang didapatkan. Oleh karena itu, untuk mendukung pengalaman bermain pengguna, aplikasi *visual novel* Mystic Pages dilengkapi dengan sejumlah fitur yang bertujuan untuk meningkatkan kenyamanan dan keterlibatan pemain selama permainan berlangsung. Fitur-fitur yang tersedia dalam aplikasi dapat dilihat pada tabel 1 berikut:

Tabel 1. Rincian fitur aplikasi

No.	Fungsional	Keterangan
1.	Aplikasi dapat memulai permainan	Sistem menampilkan layar awal dan memulai permainan dari awal cerita saat pemain memilih opsi *Start*.
2.	Aplikasi dapat menampilkan dialog cerita	Sistem menampilkan teks narasi dan dialog karakter secara berurutan sesuai alur cerita yang dijalankan.

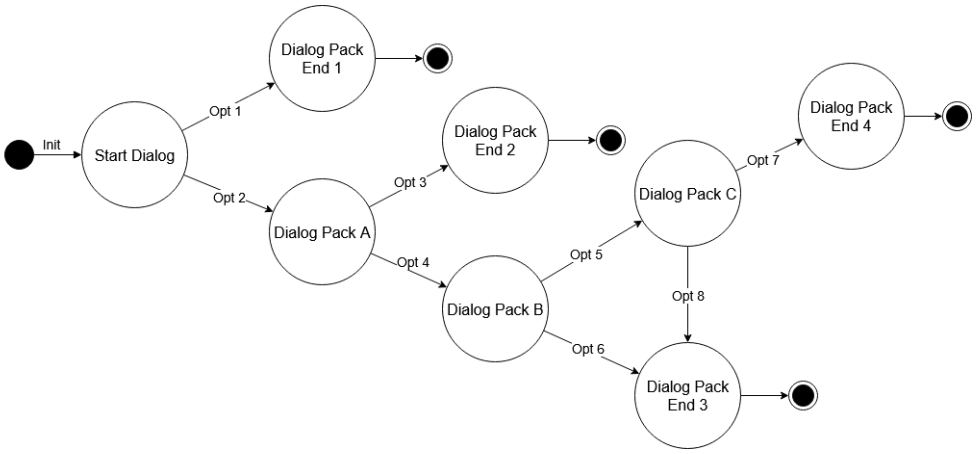
3.	Aplikasi dapat menampilkan ilustrasi latar, karakter, dan efek suara sesuai alur cerita permainan	Sistem menampilkan elemen visual (*background*, sprite karakter) dan efek audio sesuai dengan adegan cerita.
4.	Aplikasi dapat menampilkan pilihan interaktif bagi pemain	Sistem menyediakan pilihan jawaban yang dapat dipilih pemain, di mana setiap pilihan dapat membawa cerita ke jalur cerita yang berbeda .
5.	Aplikasi dapat menyimpan progres pemain	Sistem menyediakan fitur *Save* untuk menyimpan posisi terakhir pemain dalam cerita.
6.	Aplikasi dapat melanjutkan permainan dari progres pemain yang disimpan	Sistem memuat kembali data permainan terakhir yang disimpan melalui fitur *Load*.
7.	Aplikasi dapat menampilkan dialog yang terlewat	Sistem menyediakan log percakapan yang menampilkan kembali teks dialog yang sudah muncul sebelumnya.
8.	Aplikasi dapat menjalankan cerita secara otomatis	Sistem menyediakan fitur *Auto* yang menampilkan dialog secara otomatis tanpa perlu diklik pemain.
9.	Aplikasi dapat melewati cerita yang telah dibaca	Sistem menyediakan fitur *Skip* untuk mempercepat cerita yang sudah pernah dibaca sebelumnya.
10.	Aplikasi dapat mengatur volume efek suara, musik, dan kecepatan teks ditampilkan	Sistem menyediakan menu pengaturan untuk mengontrol *volume* BGM, SFX, dan kecepatan tampil teks.
11.	Aplikasi dapat menampilkan bantuan pemain	Sistem menampilkan informasi panduan penggunaan tombol dan navigasi dalam menu *Help*.
12.	Aplikasi dapat menghentikan permainan	Sistem menyediakan opsi untuk keluar dari permainan atau kembali ke menu utama melalui tombol *Exit* atau *Main Menu*.

13.	Aplikasi dapat menampilkan informasi pembuat game	Sistem menampilkan informasi mengenai tim pengembang, penulis cerita, ilustrator, dan kontributor lainnya melalui menu atau *Credit*.

Fitur-fitur tersebut mendukung kebutuhan permainan naratif, sekaligus menegaskan pentingnya pengelolaan percabangan cerita yang konsisten dan bebas konflik logika untuk mendukung fitur pilihan interaktif yang menyebabkan percabangan cerita.

3.2. Rancangan Sistem FSM

Implementasi *Finite State Machine (*FSM) pada *visual novel* Mystic Pages menghasilkan struktur naratif yang terorganisir dalam bentuk *state* dan transisi. Narasi permainan dimodelkan dalam bentuk *state* dan transisi, di mana setiap *state* merepresentasikan bagian cerita, sedangkan transisi mengatur perpindahan antar *state* berdasarkan pilihan pemain. Pendekatan ini memberikan kejelasan struktur, sehingga pengembang dapat menambahkan *state* baru tanpa mengganggu alur cerita sebelumnya. Hasil perancangan FSM di tampilkan pada gambar 3 berikut.



Gambar 3. Hasil rancangan FSM

3.3. Implementasi Sistem FSM

Implementasi sistem merupakan tahap penerapan hasil rancangan yang telah dibuat ke dalam aplikasi Mystic Pages. Pada tahap ini, logika dari diagram *finite state machine* yang telah dirancang sebelumnya diterapkan menggunakan *Ren'Py*, sebuah *game engine* berbasis Python yang secara khusus dirancang untuk pengembangan *visual novel* [15].

Proses implementasi mengacu pada rancangan alur cerita bercabang yang telah disusun sebelumnya sebagaimana disajikan pada Gambar 3, di mana setiap *state* dalam diagram mewakili bagian dialog atau adegan dalam cerita, sementara transisi antar *state* ditentukan oleh pilihan pemain. *State* dengan label "End" menunjukkan titik akhir dari suatu jalur cerita, yang bisa memiliki konteks atau konsekuensi berbeda tergantung pada pilihan sebelumnya.

Struktur FSM diubah ke dalam skrip *Ren'Py* melalui penggunaaan *label* dan *jump*, serta *variabel* untuk menyimpan keputusan pemain. Berikut beberapa potongan kode program hasil implementasi FSM pada sistem.

a. Inisialisasi *state* awal

```
def __init__(self):
    self.state = "start_dialog"
```

Potongan kode di atas memperlihatkan bagaimana *state* awal diinisialisasi sebagai start_dialog. Dengan demikian, setiap kali permainan dimulai atau FSM di-*reset*, alur cerita akan selalu dimulai dari titik yang sama, yaitu *state* "start_dialog"

b. Transisi opt 1 dan opt 2

```
def transition(self, event):
    if self.state == "start_dialog":
        if event == "opt1":
            self.state = "Dialog_Pack_End1"
        elif event == "opt2":
            self.state = "Dialog_Pack_A"
        …
```

Potongan kode di atas merupakan bagian dari fungsi *transition* yang bertugas menangani perpindahan antar *state* dalam struktur Finite State Machine (FSM). Fungsi ini menerima parameter *event* sebagai representasi pilihan pemain, kemudian memeriksa apakah *state* aktif saat ini adalah start_dialog. Jika kondisi tersebut terpenuhi, maka nilai event akan menentukan arah transisi: pilihan opt1 akan mengubah *state* menjadi Dialog_Pack_End1, sedangkan pilihan opt2 akan mengarah ke *state* Dialog_Pack_A.

```
label start:
    …
    menu :
        "What would you do?"
        "Follow your instinct":
            $ fsm.transition("opt2")
        "Trust Arden":
            $ fsm.transition("opt1")
    jump state_handler
```

Potongan kode di atas menampilkan dua pilihan awal bagi pemain. Setiap opsi memicu transisi FSM sesuai dengan logika yang telah ditentukan, kemudian diarahkan ke label state_handler. Label ini bertugas memeriksa *state* aktif dan mengarahkan alur cerita ke jalur yang sesuai.

c. *State* Dialog_Pack_End1

```
def get_state(self):
    return self.state
```

Potongan kode diatas digunakan untuk mengambil nilai *state* saat ini dari objek FSM. Fungsi ini memungkinkan sistem untuk mengetahui posisi atau status terkini dari alur cerita berdasarkan transisi yang telah dilakukan sebelumnya oleh pemain.

```
label state_handler:
    if fsm.get_state() == "Dialog_Pack_End1":
        jump wait
    ...
```

Kode diatas digunakan untuk menangani alur cerita berdasarkan *state* yang sedang aktif. Pilihan pemain akan diproses oleh state handler untuk mengarahkan jalannya cerita ke bagian yang relevan. Fungsi fsm.get_state() dipanggil untuk memeriksa kondisi *state* terkini, dan bila kondisi tersebut adalah "Dialog_Pack_End1", maka alur cerita akan diarahkan ke label wait, yang merepresentasikan salah satu bagian cerita.

3.4. Pengujian Sistem FSM

Pengujian dilakukan menggunakan metode black-box testing untuk memastikan validitas seluruh *state* dan transisi dapat dijalankan sesuai rancangan, tanpa ditemui *dead-end* maupun *infinite loop*. Tabel 2 menyajikan hasil pengujian FSM pada jalur utama cerita.

Tabel 2. Hasil pengujian implementasi FSM

No.	*State* Awal	*Input*	*State* Tujuan	Keterangan
1.	start_dialog	opt1	Dialog_Pack_End1	Sesuai
2.	start_dialog	opt2	Dialog_Pack_A	Sesuai
3.	Dialog_Pack_A	opt3	Dialog_Pack_End2	Sesuai
4.	Dialog_Pack_A	opt4	Dialog_Pack_B	Sesuai
5.	Dialog_Pack_B	opt5	Dialog_Pack_C	Sesuai
6.	Dialog_Pack_B	opt6	Dialog_Pack_End3	Sesuai
7.	Dialog_Pack_C	opt7	Dialog_Pack_End4	Sesuai
8.	Dialog_Pack_C	opt8	Dialog_Pack_End3	Sesuai

Hasil pengujian menunjukkan tingkat keberhasilan mencapai 100%, yang membuktikan bahwa FSM mampu mengelola percabangan cerita sesuai dengan rancangan. Selanjutnya, dilakukan pengujian lanjutan dengan menambahkan dua state baru beserta beberapa transisi tambahan di antara *state* Dialog_Pack_C dan *state* Dialog_Pack_End3. Tujuan pengembangan ini adalah untuk menguji fleksibilitas FSM dalam menangani perluasan alur cerita. Tabel 3 menyajikan hasil pengujian setelah penambahan tersebut.

Tabel 3. Hasil pengujian setelah penambahan

No.	*State* Awal	*Input*	*State* Tujuan	Keterangan
1.	Dialog_Pack_C	opt7	Dialog_Pack_End4	Sesuai
2.	Dialog_Pack_C	opt8	Dialog_Pack_D	Sesuai
3.	Dialog_Pack_D	opt9	Dialog_Pack_End5	Sesuai
4.	Dialog_Pack_D	opt10	Dialog_Pack_End3	Sesuai

Penambahan *state* Dialog_Pack_D dan Dialog_Pack_End5 berhasil diintegrasikan tanpa mengganggu jalur cerita yang telah ada. Seluruh *state* dan transisi baru berjalan dengan baik dengan tingkat keberhasilan 100% serta tanpa menimbulkan konflik logika. Hasil ini menunjukkan bahwa FSM memiliki fleksibilitas tinggi dalam mengakomodasi perubahan, baik berupa penambahan maupun pengurangan alur cerita. Dengan demikian, FSM terbukti mampu menjaga konsistensi sistem sekaligus memberikan ruang bagi pengembang untuk melakukan ekspansi naratif secara terstruktur.

4. KESIMPULAN

Berdasarkan penelitian yang dilakukan, penerapan *Finite State Machine* (FSM) terbukti efektif dalam mengelola kompleksitas cerita bercabang pada aplikasi *visual novel* Mystic Pages. Setiap skenario cerita dapat direpresentasikan sebagai *state* dalam diagram FSM, sementara jalur cerita berkembang melalui transisi berdasarkan pilihan pemain. Pengujian menunjukkan bahwa FSM memudahkan proses dokumentasi dan modifikasi alur cerita, di mana penambahan *state* maupun transisi baru dapat dilakukan tanpa mengganggu logika yang sudah ada. Selain itu, hasil *black-box testing* memperlihatkan bahwa seluruh transisi berjalan dengan baik tanpa adanya *dead-end* maupun *infinite loop*, dengan tingkat keberhasilan pengujian mencapai 100%. Dengan demikian, FSM dapat dijadikan metode yang andal untuk mendukung pengembangan *visual novel* atau aplikasi berbasis narasi lain yang membutuhkan pengaturan alur cerita bercabang secara sistematis, terstruktur, dan mudah dikelola.

DAFTAR PUSTAKA

[1] R. E. S. Siagian and Retno Palupi, “Pembuatan Game Visual Novel Sebagai Media Perkuliahan Menggunakan Ren’Py Berbasis Android,” *J. SAINS DAN Komput.*, vol. 8, no. 01, pp. 13–17, Jan. 2024, doi: 10.61179/jurnalinfact.v8i01.467.

[2] T. Wibowo, D. A. Adnas, Mulyanto, and A. A. Marvel, “CRAFTING AN INTERACTIVE VIDEO GAME COURT SYSTEM FOR MORAL DEVELOPMENT AND LEGAL INSIGHT,” *Zo. J. Sist. Inf.*, vol. 7, no. 1, pp. 85–94, 2024, [Online]. Available: https://journal.unilak.ac.id/index.php/zn/article/view/24145/7194.

[3] A. Abiyu, “IMPLEMENTASI FSM (FINITE STATE MACHINE) PADA GAME MALIK LOOKS FOR THE HOLY BOOK,” *JASTEN (Jurnal Apl. Sains Teknol. Nasional)*, vol. 4, no. 2, pp. 61–67, Oct. 2023, doi: 10.36040/jasten.v4i2.8155.

[4] B. Ngaw, G. Jena, J. Sedoc, and A. Normoyle, “Towards Authoring Open-Ended Behaviors for Narrative Puzzle Games with Large Language Model Support,” in *Proceedings of the 19th International Conference on the Foundations of Digital*

Games, May 2024, vol. 14, no. 2, pp. 1–4, doi: 10.1145/3649921.3656975.

[5] M. Riedl and R. Young, “From Linear Story Generation to Branching Story Graphs,” *Proc. AAAI Conf. Artif. Intell. Interact. Digit. Entertain.*, vol. 1, no. 1, pp. 111–116, Sep. 2021, doi: 10.1609/aiide.v1i1.18725.

[6] Y. Bi, “Tree narrative and AI interaction: Optimization and narrative bias handling prospects,” *Appl. Comput. Eng.*, vol. 71, no. 1, pp. 163–167, Sep. 2024, doi: 10.54254/2755-2721/71/20241616.

[7] D. Jagdale, “Finite State Machine in Game Development,” *Int. J. Adv. Res. Sci. Commun. Technol.*, vol. 10, no. 1, pp. 384–390, Oct. 2021, doi: 10.48175/IJARSCT-2062.

[8] N. Muchsin Sanin, “IMPLEMENTASI FSM (FINITE STATE MACHINE) PADA GAME MUSLIMS EXPLORER’S FAITHFUL MAZE ADVENTURE,” *JASTEN (Jurnal Apl. Sains Teknol. Nasional)*, vol. 5, no. 1, pp. 46–54, Apr. 2024, doi: 10.36040/jasten.v5i1.8170.

[9] A. Riyadi and S. I. A. Y. Syah, “Development of Finite State Machine Agent in Idle Breeder Game,” *CESS (Journal Comput. Eng. Syst. Sci.*, vol. 8, no. 2, p. 542, Jul. 2023, doi: 10.24114/cess.v8i2.47143.

[10] M. Mustofa, V. Maarif, A. Novel, S. Sunanto, and C. Kesuma, “Penerapan Finite State Machine Dalam Inteaksi Dialog Dalam Novel Game Forest Life,” *Comput. Sci.*, vol. 2, no. 2, pp. 127–136, Jul. 2022, doi: 10.31294/coscience.v2i2.1291.

[11] J. Saputra and K. Handoko, “IMPLEMENTASI FINITE STATE MACHINE DALAM GAME EDUKASI BAHASA JEPANG,” *J. Comasie*, vol. 11, no. 02, pp. 119–128, 2024, [Online]. Available: https://ejournal.upbatam.ac.id/index.php/comasiejournal/article/view/9084/4180.

[12] A. N. Gumilang, H. Haryanto, and E. Dolphina, “Pengembangan Game RPG dan Story Dengan Elemen Gameplay Menggunakan Metodologi Finite State Machine (FSM) Pada Game Kisah Tjepoe,” *TECHNO Creat.*, vol. 1, no. 2, p. 129, Jan. 2024, doi: 10.62411/tcv.v1i2.2069.

[13] M. Hasan Syu’aibi, A. Mahmudi, and K. Auliasari, “PERANCANGAN DAN IMPLEMENTASI METODE FSM (FINITE STATE MACHINE) PADA GAME MILITARY DEFENCE 2D BERBASIS ANDROID,” *JATI (Jurnal Mhs. Tek. Inform.*, vol. 7, no. 4, pp. 2349–2357, Dec. 2023, doi: 10.36040/jati.v7i4.7508.

[14] A. Ayu Setyaningrum, A. Panji Sasmito, and H. Zulfia Zahro’, “PENERAPAN METODE FINITE STATE MACHINE PADA GAME ADVENTURE NOIR,” *JATI (Jurnal Mhs. Tek. Inform.*, vol. 8, no. 2, pp. 1298–1305, Apr. 2024, doi: 10.36040/jati.v8i2.9098.

[15] N. C. Rikandi and S. R. Nudin, “Rancang Bangun Visual Novel Peduli Lingkungan dengan Metode Procedural Content Generation,” *J. Informatics Comput. Sci.*, vol. 4, no. 01, pp. 131–142, Aug. 2022, doi: 10.26740/jinacs.v4n01.p131-142.