(2024), 2 (7): 44–54 Scientica 3021-8209
Jurnal Ilmiah Sains dan Teknologi

PENERAPAN FINITE STATE MACHINE DALAM GAME PADA MODEL KARAKTER PEMAIN MENGGUNAKAN GODOT ENGINE

THE APPLICATION OF FINITE STATE MACHINE IN GAMES FOR PLAYABLE CHARACTER MODEL USING GODOT ENGINE

Mhd Galih Khairi
Program Studi Ilmu Komputer, Fakultas Sains dan Teknologi,
Universitas Islam Negeri Sumatera Utara
E-mail: mgalihkhairi1@gmail.com

Abstract (English)

The Finite State Machine (FSM) is a theoretical computing concept that can be applied in various aspects of everyday life. As a simple mathematical model that describes a system with a limited and defined number of states and transitions between states, FSM is found in many systems and human behaviors. In everyday life, FSM can be used to explain decision-making processes, habitual behaviors, as well as workflow and routine activities. For example, a person's morning routine, such as waking up, taking a shower, having breakfast, and going to the office, can be described using the FSM concept. Even more simply, a person who is still, then walks, then runs, and then becomes still again can also be categorized as an FSM with clear states and transitions between states. Similarly, the system processes in electronic devices such as making coffee or operating a washing machine can be modeled using the FSM concept. This shows that human systems and behaviors can be combined and conceptualized in a game in the form of an FSM. Therefore, this research aims to understand the concept of FSM in a framework of playable character models in games using the Godot engine software. So that the results of this research in the future can be used as a reference for creating an FSM system on player character models in a game.

Article History
Submitted: 25 May 2024
Accepted: 4 June 2024
Published: 5 June 2024

Key Words
Finite State Machine, Playable Character, Game, Godot Engine

Abstrak (Indonesia)

FSM (*Finite State Machine*) merupakan konsep komputasi teoritis yang dapat diaplikasikan dalam berbagai aspek kehidupan sehari-hari. Sebagai model matematika sederhana yang menggambarkan sistem dengan jumlah status dan transisi antar status yang terbatas dan terdefinisi, FSM ditemukan pada banyak sistem serta perilaku manusia. Dalam kehidupan sehari-hari, FSM dapat digunakan untuk menjelaskan proses pengambilan keputusan, perilaku kebiasaan, maupun alur kerja dan aktivitas rutin. Sebagai contoh, rutinitas pagi hari seseorang, seperti bangun tidur, mandi, sarapan, dan berangkat ke kantor, dapat digambarkan menggunakan konsep FSM. Lebih sederhana lagi, seseorang yang diam, berjalan, kemudian berlari dan kembali lagi diam juga dapat dikategorikan sebagai suatu FSM dengan *state-state* dan transisi antar *state* yang jelas. Demikian pula, proses sistem dalam alat elektronik seperti membuat kopi atau pengoperasian mesin cuci juga dapat dimodelkan menggunakan konsep FSM. Hal ini menunjukkan bahwa sistem dan perilaku manusia dapat digabungkan dan dikonsepkan dalam sebuah game dalam bentuk FSM. Sehingga penelitian ini bertujuan untuk memahami konsep FSM pada sebuah kerangka model karakter yang pemain gunakan (*Playable Character*) di dalam *game* menggunakan *software godot engine*. Sehingga hasil dari penelitian ini kedepannya dapat digunakan sebagai acuan pembuatan sistem FSM pada model karakter pemain dalam sebuah *game*.

Sejarah Artikel
Submitted: 25 May 2024
Accepted: 4 June 2024
Published: 5 June 2024

Kata Kunci
Finite State Machine, Playable Character , Game, Godot Engine.

1. PENDAHULUAN

Di abad 21 ini, kemajuan teknologi berkembang dengan cepat, dan telah merasuki berbagai bidang, terutama dalam hal hiburan. Kehidupan yang sibuk dan padat di zaman sekarang membuat manusia rentan mengalami kelelahan dan stres. Sebagai solusinya,

diciptakanlah sebuah bentuk hiburan. Dengan memanfaatkan kemajuan teknologi komputer yang sudah cukup maju, diciptakanlah suatu bentuk hiburan yang dikenal dengan sebutan *game*. *Game* adalah suatu bentuk permainan yang menggunakan teknologi elektronik sebagai medianya. *Game* sangat diminati oleh berbagai kalangan sebagai sumber hiburan dan sarana pembelajaran. Dalam perkembangannya, *game* terus mengalami inovasi dengan berbagai pilihan *genre* seperti *maze game, board game, card game, battle game, quiz game, puzzle game, shooting game, first person shooting game, side scrolling game, fighting game, racing game, simulation game, strategy game, role playing game, adventure game, sport game*, dan *edutainment game* [1]. Game memiliki peran penting dalam menyegarkan pikiran dan meredakan kepenatan akibat aktivitas dan rutinitas harian [2]. Bagi pengguna komputer, game menjadi kebutuhan yang signifikan, karena untuk menikmati pengalaman bermain game dengan baik, diperlukan komponen-komponen komputer yang berkualitas tinggi, terutama kartu grafis (*VGA card*).

Dalam sebuah *game*, terdapat berbagai karakter, baik itu karakter NPC (*Non-Playable Character*) yang bergerak secara mandiri, maupun karakter pemain yang dapat dikendalikan melalui alat atau komponen komputer seperti *controller* stik atau *keyboard*. Pergerakan karakter-karakter tersebut bukanlah tanpa sebab, melainkan dikarenakan adanya program komputer yang disebut FSM (*Finite State Machine*) yang tertanam dalam model karakter tersebut. FSM (*Finite State Machine*) sendiri merupakan suatu metode desain sistem yang dapat menggambarkan perilaku sistem dengan menggunakan tiga komponen utama, yaitu keadaan, kejadian, dan aksi. Sistem memiliki kemampuan untuk berpindah atau bertransisi ke keadaan lain ketika menerima masukan atau peristiwa tertentu, baik dari perangkat eksternal maupun komponen internal sistem itu sendiri. Transisi keadaan (*state*) ini biasanya juga disertai dengan tindakan yang dilakukan oleh sistem sebagai respons terhadap masukan yang terjadi. Tindakan yang dilakukan dapat berupa tindakan sederhana ataupun melibatkan serangkaian proses yang cukup kompleks [3]. Dengan menerapkan FSM dalam sebuah game, maka karakter yang kita mainkan dapat dapat memiliki perilaku yang terstruktur dan dapat diprediksi. FSM memungkinkan gerakan karakter untuk berpindah dari satu keadaan (*state*) ke keadaan lain berdasarkan kondisi tertentu. Hal ini memungkinkan karakter yang dikendalikan pemain untuk merespons dan berinteraksi dengan lingkungan game sesuai keadaan (*state*) yang telah dimasukkan sebelumnya [4].

2. TINJAUAN PUSTAKA

2.1 *Finite State Machine*

FSM (*Finite State Machine*) adalah metode yang digunakan untuk memodelkan perilaku karakter dalam game berdasarkan keadaan yang telah ditentukan sebelumnya. FSM merupakan representasi matematis dari sebuah sistem yang terdiri dari sejumlah keadaan yang berubah dari satu keadaan ke keadaan lain melalui transisi. Keadaan ini mencerminkan kondisi internal yang unik dari sistem. Secara formal, FSM terdiri dari lima entitas utama, yaitu status simbolik, sinyal *input*, sinyal *output*, fungsi keadaan berikutnya, dan fungsi *output*. Setiap keadaan menunjukkan kondisi internal yang unik dari sistem. Seiring berjalannya waktu, FSM berpindah dari satu keadaan ke keadaan lainnya. Keadaan baru ditentukan oleh fungsi keadaan berikutnya, yang merupakan fungsi dari keadaan saat ini dan sinyal input [5]. Dalam *Synchronous Finite State Machine*, transisi antar keadaan dikendalikan oleh sinyal jam dan hanya terjadi pada saat tepat ketika sinyal jam dinyalakan atau dimatikan. Jadi, mesin ini hanya berpindah ke keadaan baru pada momen-momen tertentu yang ditentukan oleh sinyal jam. Fungsi output bertanggung jawab untuk menghasilkan nilai output yang sesuai. Ada dua jenis mesin yang umum digunakan yaitu Moore dan Mealy. Pada mesin Moore, nilai *output*

ditentukan berdasarkan keadaan mesin pada saat itu. Sedangkan pada mesin Mealy, nilai *output* juga bergantung pada sinyal *input* yang diterima [6].

2.2 *Game*

Game dapat didefinisikan sebagai sebuah program komputer yang dirancang khusus untuk kebutuhan hiburan dan menggunakan teknik serta metode animasi. Sebagai karakteristiknya, *game* memiliki aturan dan panduan yang ditetapkan untuk para pemainnya, agar mereka dapat mencapai tujuan atau goal tertentu yang telah ditentukan dalam game tersebut. *Game* dapat hadir dalam berbagai jenis, seperti game dengan banyak level yang menunjukkan tingkat kesulitan yang berbeda-beda, atau game tanpa level yang dapat dimainkan berulang-ulang dengan aturan dan tingkat kesulitan yang sama. Keberagaman jenis game ini memungkinkan para pemain untuk memilih dan menikmati pengalaman bermain yang sesuai dengan preferensi mereka. Saat ini, game telah menjadi aktivitas yang tak terpisahkan dari kehidupan orang-orang, karena game dianggap sebagai program yang dirancang khusus untuk memenuhi kebutuhan hiburan. Untuk dapat menciptakan sebuah game, diperlukan *game engine* sebagai alat bantu bagi pengembang untuk mengatur animasi, grafis, dan konten yang akan ada di dalam game tersebut. Dengan adanya game engine, proses pembuatan game menjadi lebih terstruktur dan efisien [7].

2.3 *Godot Engine*

Godot Engine adalah sebuah game *engine* lintas *platform* yang sangat populer dan digunakan oleh para pengembang game di seluruh dunia. Dengan fokus pada pengembangan game 2D dan 3D, *Godot Engine* menyediakan berbagai fitur dan alat yang komprehensif untuk mendukung seluruh proses pembuatan *game*. Salah satu keunggulan utama dari Godot Engine adalah kemampuannya dalam menyediakan berbagai tools yang mempermudah pengguna dalam mengembangkan game. Dengan adanya fitur-fitur ini, pengguna dapat dengan mudah membuat game yang menarik dengan kekayaan elemen visual dan gameplay yang beragam. *Godot Engine* juga menawarkan fleksibilitas dalam mengekspor *game* ke berbagai *platform*. Pengguna dapat dengan mudah mengekspor game mereka dengan satu kali klik ke platform desktop seperti Windows, Linux, dan MacOS. Selain itu, *Godot Engine* juga mendukung platform mobile seperti Android dan iOS, serta platform berbasis web seperti HTML5 [8]. Selain fitur-fitur yang telah disebutkan, Godot Engine juga menawarkan dukungan untuk bahasa pemrograman yang beragam, termasuk *GDScript* (bahasa *scripting* khusus untuk *Godot Engine*), C#, dan C++. Hal ini memberikan fleksibilitas kepada pengembang dalam memilih bahasa pemrograman yang paling sesuai dengan kebutuhan proyek mereka. Dengan keseluruhan fitur-fitur dan dukungan yang dimilikinya, *Godot Engine* menjadi pilihan yang sangat menarik bagi pengembang game yang ingin menciptakan game yang kreatif dan berkualitas tinggi. *Platform* ini terus berkembang dan memiliki komunitas yang aktif, yang berarti pengguna dapat dengan mudah mencari bantuan dan berbagi pengetahuan dengan pengembang lainnya [9].

2.4 *Framework*

Framework dalam *game engine* dapat didefinisikan sebagai struktur dasar atau fondasi yang menyediakan alat, pustaka, dan fungsionalitas siap pakai untuk membantu dan memudahkan dalam pengembangan game. *Framework* ini berperan sebagai jembatan antara pengembang dan proses pengembangan game [10]. Beberapa karakteristik utama dari *framework game engine* antara lain:

1. Arsitektur terstruktur, di mana *framework* biasanya memiliki struktur yang terorganisir, membagi game menjadi berbagai modul atau komponen yang saling terhubung, seperti manajemen grafik, input, audio, dan fisika.

2. Fungsionalitas inti, di mana *framework* menyediakan fungsi-fungsi dasar yang dibutuhkan dalam pengembangan game, seperti rendering grafik, manajemen aset, deteksi tabrakan, dan kontrol input. Hal ini membantu pengembang tidak perlu membangun semua komponen dasar game dari awal.
3. Abstraksi dan modularitas, di mana *framework* menyediakan lapisan abstraksi yang menyembunyikan kerumitan implementasi teknis, memungkinkan pengembang fokus pada logika dan *gameplay game*. Struktur modular juga memudahkan penggantian atau perluasan komponen tertentu.
4. Portabilitas dan interoperabilitas, di mana banyak *framework* bersifat *multiplatform*, sehingga game dapat dijalankan di berbagai platform dengan sedikit atau tanpa modifikasi kode. *Framework* juga sering terintegrasi dengan berbagai pustaka dan alat pengembangan lainnya.
5. Komunitas dan dokumentasi, di mana *framework* populer biasanya didukung oleh komunitas besar yang menyediakan dokumentasi, tutorial, dan solusi untuk masalah umum, sangat membantu bagi pengembang baru.

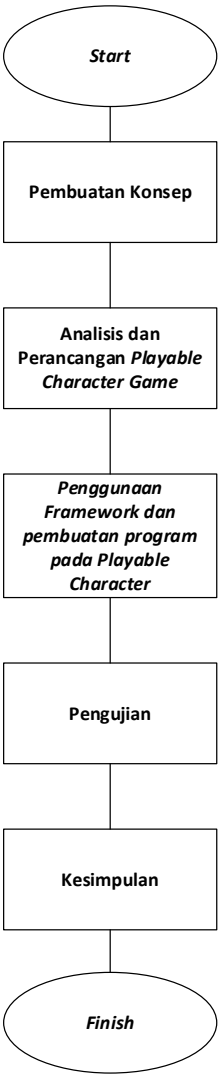
2.5 *Playable Character*

Playable character adalah karakter yang dapat dikendalikan langsung oleh pemain. *Playable character* menjadi jembatan antara pemain dan dunia game, memungkinkan pemain untuk terlibat secara aktif dan bermakna dalam pengalaman permainan. Dalam konsep *finite state machine* (FSM), *playable character* dapat dimodelkan sebagai sebuah mesin dengan berbagai *state* atau kondisi yang dapat diubah berdasarkan input dari pemain. Setiap playable character memiliki set kemampuan, statistik, dan atribut yang unik, seperti kekuatan, kecepatan, senjata, dan keahlian khusus, yang dapat dikembangkan oleh pemain selama permainan. Perpindahan dari satu *state* ke *state* lainnya terjadi melalui transisi, yang diaktifkan oleh *input* atau aksi yang dilakukan oleh pemain, seperti menekan tombol, menggerakkan *joystick*, atau mengklik *mouse*. Setiap transisi antara *state* memiliki aturan atau kondisi yang menentukan kapan transisi tersebut terjadi, dan setiap *state* dapat memiliki aksi yang dilakukan oleh *playable character*, seperti animasi gerakan, penggunaan kemampuan, atau interaksi dengan lingkungan game [11].

3. ANALISIS DAN PERANCANGAN

3.1 Tahap Perencanaan

Pada tahap perencanaan pengembangan sebuah model *playable character* game dengan menggunakan konsep FSM (*Finite State Machine*), terdapat beberapa langkah dan tahapan yang dapat kita lihat pada flowchart yang disediakan. *Flowchart* ini menggambarkan alur dan proses perancangan model game berbasis FSM secara terperinci, mulai dari tahap membuat konsep dan penggunaan *framework*, dilanjutkan dengan pembuatan program, hingga tahap pengujian dan analisis. Adapun tahapannya bisa dilihat pada *flowchart* struktur berikut ini.



Gambar 1. Struktur rincian kerja

Dalam mengembangkan game menggunakan *godot engine*, spesifikasi perangkat komputer atau laptop menjadi faktor penting yang harus diperhatikan. Jika spesifikasi yang dimiliki tidak memenuhi persyaratan minimal untuk menjalankan *software* tersebut, maka tahap perancangan dan pengembangan model game tidak dapat dilanjutkan. Oleh karena itu, pemahaman mengenai spesifikasi minimal yang dibutuhkan untuk mengoperasikan *godot engine* menjadi hal krusial bagi pengembang game.

Tabel 1. Spesifikasi minimum menggunakan godot engine

Kebutuhan Sistem	Minimum
Sistem Operasi	Windows 7
Kartu Grafis	NVIDIA GeForce 6200
Processor	Intel Core 2 Duo E8400
Memori RAM	4GB

3.2 Tahap Perancangan

Pada perancangan model karakter ini, pastinya memiliki beberapa state atau keadaan yang mempengaruhi setiap tindakan yang dilakukan pemain, sehingga state-state inilah nanti

yang akan membuat sebuah model *playable character* memiliki tindakan seperti mahluk hidup pada umumnya. Contoh halnya seperti diam, bergerak, menyerang, dan juga melompat. Setiap state tersebut memiliki kondisi dan logika masing-masing yang harus diimplementasikan dengan baik agar karakter dapat bergerak dan bertindak secara realistis dan menarik. Misalnya, saat karakter dalam state diam, ia harus tetap mampu merespon input pengguna untuk berpindah posisi, meskipun dalam kondisi diam. Saat dalam state bergerak, karakter harus dapat berpindah posisi dengan kecepatan dan arah yang sesuai dengan input pengguna. Sementara dalam state menyerang, karakter harus dapat mengeluarkan aksi menyerang yang memiliki jangkauan, kekuatan, dan durasi yang tepat. Dengan mengatur dengan cermat setiap *state* dan transisi antar *state*-nya, model karakter akan dapat bergerak dan bertindak secara natural dan meyakinkan, sehingga pemain dapat terlibat dan terhibur dengan interaksi karakter yang realistis.

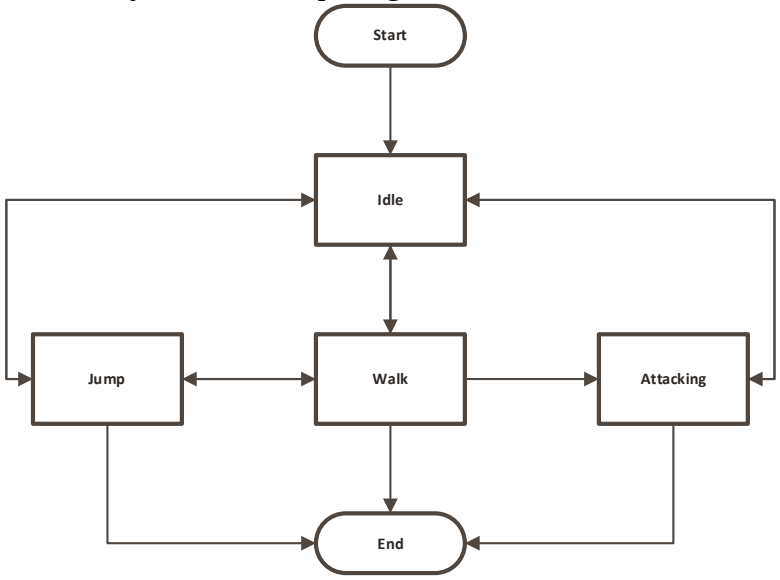
Tabel 2. Macam-macam *state playable character*

No	State	Keterangan	Input
1.	*IDLE*	Keadaan di mana karakter tidak sedang bergerak atau melakukan tindakan apa pun, karakter tersebut akan tetap diam dan tidak bergerak sama sekali. Karakter akan terus berada dalam posisi diamnya tersebut hingga menerima input atau perintah dari pemain untuk melakukan tindakan tertentu. Pemain memiliki kendali penuh atas kapan karakter harus bergerak atau melakukan aksi tertentu. Hingga pemain memberikan perintah, karakter akan tetap pasif dan tidak menunjukkan aktivitas apa pun.	Tidak ada
2.	*WALK*	Pada keadaan ini, karakter diprogram untuk dapat bergerak bebas, karakter tersebut akan melakukan gerakan berjalan ke berbagai arah sesuai dengan input atau perintah yang diberikan oleh pemain melalui kontrol kendali. Pemain dapat menggunakan tombol-tombol pada keyboard untuk mengendalikan arah pergerakan karakter, apakah itu ke kanan, kiri, atas, atau bawah. Karakter akan merespons secara real-time terhadap masukan yang diberikan oleh pemain, dengan melakukan gerakan berjalan yang halus dan natural mengikuti arah yang diperintahkan. Pergerakan karakter dapat disesuaikan dengan kecepatan atau tempo yang diinginkan, tergantung pada pengaturan atau preferensi pemain. Dengan demikian, pemain memiliki kendali penuh atas arah dan kecepatan pergerakan karakter dalam game.	A,W,S,D
		Pada keadaan ini, karakter diprogram untuk dapat melakukan gerakan melompat, pemain	

3.	*JUMP*	dapat mengaktifkan gerakan tersebut dengan menekan tombol khusus pada keyboard. Saat tombol melompat ditekan, karakter akan terangkat ke atas, melakukan gerakan melayang di udara untuk beberapa saat. Selama di udara, karakter dapat mengontrol arah loncatan dengan menggunakan tombol arah pada keyboard. Setelah mencapai titik tertinggi, karakter akan mulai jatuh kembali ke bawah karena pengaruh gravitasi. Pemain dapat mengatur timing dan ketinggian lompatan dengan menekan dan melepaskan tombol melompat pada saat yang tepat. Saat mendarat, karakter akan kembali ke posisi berdiri dengan stabil, siap untuk melakukan gerakan selanjutnya berdasarkan input pemain. Fitur lompat ini memberikan pemain kemampuan untuk melewati rintangan atau menjangkau area yang lebih tinggi dalam game.	Space
4.	*ATTACKING*	Pada keadaan di mana karakter memiliki kemampuan menyerang dengan menggunakan senjata, pemain dapat mengaktifkan gerakan menyerang tersebut dengan menekan tombol khusus yang telah ditentukan pada keyboard. Saat tombol menyerang ditekan, karakter akan secara otomatis mengeluarkan sebuah pedang dari tempat penyimpanannya. Dengan pedang ditangan, karakter akan bersiap untuk melakukan serangan. Selama melakukan serangan, karakter akan diam di tempat dan tidak dapat bergerak ke arah manapun. Gerakan penyerangan tersebut akan memakan waktu sesaat sebelum selesai. Setelah serangkaian gerakan menyerang selesai, karakter akan kembali ke posisi idle dan pemain dapat kembali mengendalikan pergerakan karakter secara bebas. Fitur menyerang dengan pedang ini memungkinkan karakter untuk melakukan serangan jarak dekat terhadap lawan atau rintangan, namun di sisi lain mengorbankan mobilitas sementara waktu.	F

Dengan keadaan-keadaan (*states*) tersebut, maka karakter pemain akan memiliki berbagai pilihan gerakan dan tindakan yang dapat dilakukan oleh karakter dalam game. Pemain dapat memanfaatkan keadaan diam untuk mempersiapkan langkah selanjutnya, lalu menggunakan kemampuan berjalan untuk menjelajahi area permainan. Saat dihadapkan pada

rintangan, pemain dapat mengaktifkan kemampuan melompat untuk melewati atau mendekati target. Di saat yang tepat, pemain juga dapat mengeluarkan senjata dan melakukan serangan untuk mengatasi musuh atau menyelesaikan misi. Kombinasi dari berbagai keadaan (*states*) ini memungkinkan pemain untuk beradaptasi dengan berbagai situasi yang muncul dalam game. Bentuk dari alur *flowchart*nya bisa dilihat pada gambar berikut ini.



Gambar 2. *Flowchart* alur *state playable character*

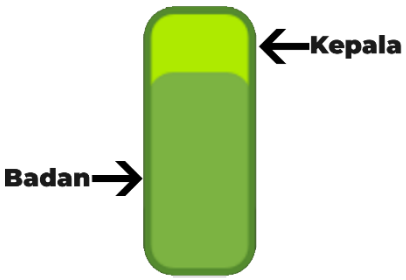
Flowchart ini mengilustrasikan bagaimana transisi antar state pada model karakter diimplementasikan. Di dalam *flowchart* tersebut, kita dapat melihat berbagai state utama yang dimiliki oleh karakter, seperti state diam, bergerak, menyerang, dan melompat. Setiap *state* ini memiliki kondisi atau trigger yang akan menentukan kapan karakter akan berpindah dari satu state ke state lainnya. Misalnya, saat karakter berada di state diam, jika ada input pergerakan dari pemain, maka karakter akan berpindah ke state bergerak. Saat di *state* bergerak, jika pemain memberikan input untuk menyerang, maka karakter akan beralih ke state menyerang. Dan jika pemain memberi perintah untuk melompat, karakter akan masuk ke *state* melompat. Alur transisi antar state ini harus dirancang dengan hati-hati agar pergerakan dan tindakan karakter terasa natural dan responsif terhadap input pemain. Setiap state juga perlu memiliki logika internal yang detail, seperti kecepatan bergerak, jangkauan serangan, tinggi lompatan, dan lain-lain. Dengan *flowchart* dan implementasi *state machine* yang baik, model karakter dapat berperilaku layaknya makhluk hidup yang interaktif dan menarik untuk dimainkan.

4. Hasil dan Pembahasan

4.1 Implementasi

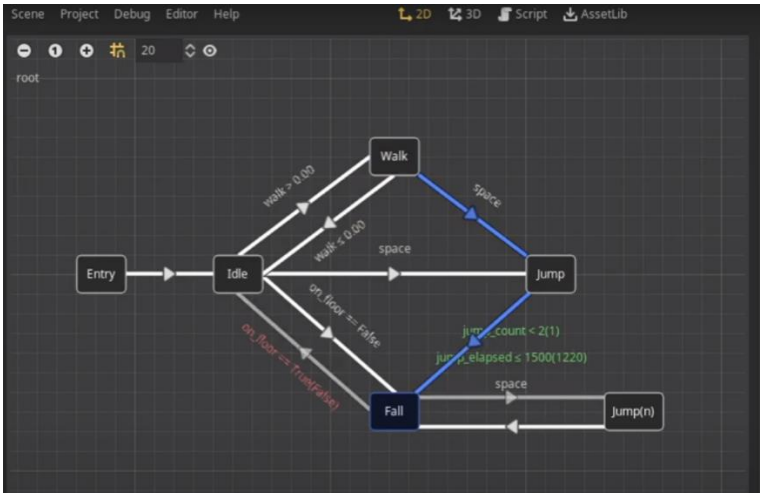
Dalam tahap implementasi proyek pembuatan game ini, langkah pertama yang dilakukan adalah menginstal *Godot Engine* sebagai perangkat lunak utama yang akan digunakan untuk membangun model karakter *playable* (yang dapat dimainkan oleh pengguna) dalam game tersebut. Program untuk menciptakan karakter *playable* dalam game tersebut tidak dibuat dari awal atau dari nol, melainkan menggunakan *framework* (kerangka kerja) yang telah tersedia di dalam *Godot Engine*. Dengan menggunakan *framework* yang sudah ada, proses pembuatan karakter playable menjadi lebih mudah dan efisien, karena pengembang tidak perlu membangun segalanya dari dasar. Adapun terdapat dua framework utama yang digunakan dalam pembuatan karakter playable ini. Pertama, framework yang berfungsi sebagai model dasar bentuk karakter 2 dimensi, yaitu berupa sebuah kotak persegi panjang berwarna hijau.

Framework ini langsung tersedia di dalam perangkat lunak Godot Engine dan dapat dimodifikasi lebih lanjut sesuai kebutuhan.



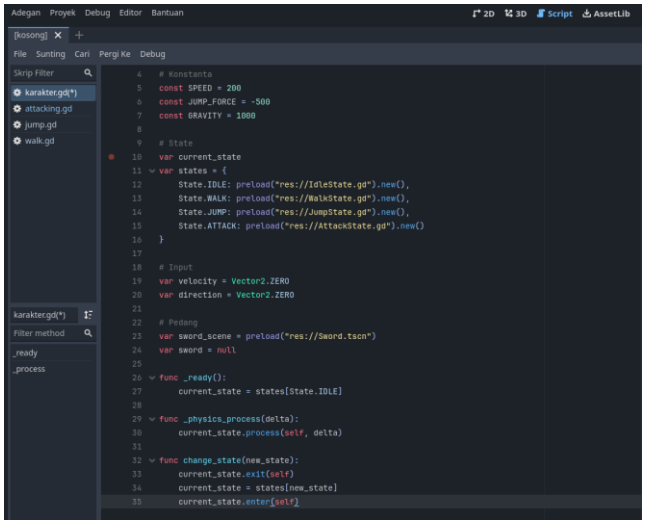
Gambar 3. Model karakter yang digunakan pemain

Selain itu, terdapat sebuah *framework* bernama gd-YAFSM (*Yet Another Finite State Machine*). *Framework* ini dirancang untuk memudahkan pengembang dalam menghubungkan *file-file state* program satu sama lain. Biasanya, untuk menghubungkan berbagai file state, pengembang harus melakukan proses penghubungan secara manual melalui kode program. Namun dengan gd-YAFSM, pengembang cukup menghubungkan *file-file state* secara visual seperti menyusun sebuah *flowchart*. *Framework* ini akan secara otomatis menghubungkan *file-file state* program tersebut sehingga tidak perlu lagi melakukan penghubungan secara manual melalui kode program. Dengan demikian, proses pengembangan aplikasi berbasis finite state machine menjadi lebih efisien dan terstruktur.



Gambar 4. Kegunaan dari *framework* gd-YAFSM

Setelah menggunakan kedua *framework* ini, maka tinggal mengintegrasikan program-program di dalam setiap *file state*, seperti *Idle*, *Walk*, *Jump*, dan *Attacking*. Setiap *file state* tersebut memiliki program yang berbeda-beda namun saling terhubung. Sebagai contoh, *file Idle* berisi program untuk gerakan dengan kecepatan 0 atau tanpa kecepatan sama sekali, yang kemudian dapat terhubung ke *state Walk*, *Jump*, dan *Attacking*.

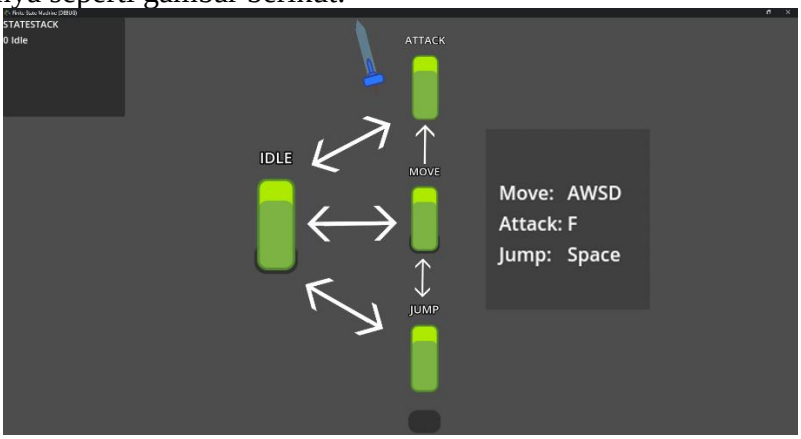


Gambar 5. Program *file state* karakter pemain di *Godot Engine*

File Walk sendiri memiliki program yang berisi kecepatan gerakan karakter di atas 0 dan inputan arah, sehingga karakter mampu bergerak ke berbagai arah. Sementara itu, *file Jump* berisi program untuk membuat karakter melayang sementara, kemudian jatuh kembali saat tombol khusus pada *keyboard* ditekan. Terakhir, *file Attacking* berisi program untuk membuat karakter melakukan serangan dengan mengeluarkan pedang yang bergerak, di mana model pedang tersebut sebelumnya berasal dari *framework*. Saat menyerang, karakter akan diam atau dalam keadaan Idle untuk sementara waktu, sampai selesai menyerang, sehingga tidak dapat melakukan gerakan lain seperti melompat dan berjalan.

4.2 Pengujian

Hasil rancangan model playable character menampilkan model karakter pmain yang dapat begerak ke berbagai state yang terintegrasi, seperti *Idle, Walk, Jump*, dan *Attacking*. Setiap *state* memiliki animasi dan gerakan yang natural dan responsif terhadap input pemain. Adapun bentuknya seperti gambar berikut.



Gambar 6. Hasil program penerapan fsm pada karakter gameyang digunakan pemain

Dari gambar tersebut, terlihat bahwa FSM (*Finite State Machine*) dapat diimplementasikan pada karakter game yang kita gunakan, di mana dengan menekan tombol sesuai input yang telah dimasukkan ke dalam program, karakter dapat bergerak berdasarkan perintah input yang diberikan.

5. KESIMPULAN

Kesimpulannya adalah bahwa dalam pengembangan sebuah model *playable character game* tang menggunakan konsep FSM memiliki langkah-langkah dan tahapan tertentu yang digambarkan secara terperinci alur dan proses perancangannya, dimulai dari membuat konsep dan menggunakan *framework,* pembuatan program, hingga tahap pengujian dan analisis. Selain itu, *flowchart*-nya juga mengilustrasikan bagaimana transisi antar *state* pada model karakter diimplementasikan, dengan setiap state memiliki kondisi atau *trigger* yang menentukan perpindahan ke *state* lainnya. Tahap implementasi melibatkan penggunaan *Godot Engine* sebagai perangkat lunak utama dan *framework* gd-YAFSM untuk memudahkan pengembangan aplikasi berbasis finite state machine. Hasilnya adalah model karakter yang responsif terhadap *input* pemain dengan berbagai state yang terintegrasi, seperti *Idle, Walk, Jump,* dan *Attacking*, sehingga membuat karakter dapat bergerak dan bertindak secara natural dalam permainan.

6. DAFTAR PUSTAKA

[1] S. Ramadaniati *et al.*, "Rancang Bangun Mobile Game Adventure Of Studies Sebagai Media Pembelajaran," 2021.

[2] I. Kurniawati and H. Purnomo, "ELEMENTA: JURNAL PGSD STKIP PGRI BANJARMASIN", doi: 10.33654/pgsd.

[3] A. M. Rumakey, J. Dedy Irawan, and A. Wahid, "PEMBUATAN GAME 2D 'ESCAPE PLAN' DENGAN METODE FINITE STATE MACHINE," 2020.

[4] S. Azis, "Perancangan Dan Pembuatan Game Bertema Melawan Penyakit di Indonesia dengan Penerapan Metode Finite State Machine (FSM)," 2023. [Online]. Available: http://pijarpemikiran.com/index.php/Scientia

[5] K. Solekhan Arif, M. Hendra Sugiharto, F. Nugroho, and J. Nur Fadilah, "Rancang Bangun Game Labirin Hijaiyah Dengan Unity Menggunakan Metode Finite State Machine," vol. 4, no. 1, pp. 49–53, 2020, [Online]. Available: http://e-journal.unipma.ac.id/index.php/doubleclick

[6] V. Vijay *et al.*, "Physically Unclonable Functions Using Two-Level Finite State Machine," *Journal of VLSI Circuits and Systems*, vol. 4, no. 1, pp. 33–41, Jun. 2022, doi: 10.31838/jvcs/04.01.06.

[7] Y. I. Kurniawan and M. F. Rivaldi, "Jurnal Manajemen Informatika (JAMIKA) Game Edukasi Pengenalan dan Pembelajaran Berhitung untuk Siswa Kelas 1 Sekolah Dasar", doi: 10.34010/jamika.v11i1.

[8] T. Darmanto, "PERANCANGAN GAME 3D MR. EGGY SIDE-SCROLLING BERBASIS ANDROID DENGAN MENGGUNAKAN METODE COLLISION DETECTION."

[9] P. Harsadi, W. L. Y. Saptomo, and C. Y. Wardhana, "Implementasi Algoritma Fisher-Yates Shuffle Pada Game Edukasi Aksara Jawa Menggunakan Godot Engine," *Jurnal Teknologi Informasi dan Komunikasi (TIKomSiN)*, vol. 10, no. 1, May 2022, doi: 10.30646/tikomsin.v10i1.603.

[10] F. Daffa, N. Zhafran, F. Prasetyanto, and T. Zani, "APLIKASI MULTIMEDIA INTERAKTIF UNTUK MUSEUM GEOLOGI BANDUNG MODUL GAME EDUKASI INTERACTIVE MULTIMEDIA APPLICATIONS FOR THE BANDUNG GEOLOGY MUSEUM EDUCATIONAL GAME MODULE."

[11] M. Farrel Arrazzaq, A. Panji Sasmito, and H. Zulfia Zahro, "PERANCANGAN GAME 2D PLATFORMER 'ADVENTURE QUEST' DENGAN METODE FINITE STATE MACHINE BERBASIS ANDROID," 2023.