BAB II

KAJIAN PUSTAKA

A. Kajian Teoritis

1. Game

Menurut Najuah et al., (2022) game edukasi merupakan permainan yang dirancang khusus untuk menyisipkan konten pembelajaran atau materi edukatif, dengan tujuan mengembangkan kompetensi pemain (siswa). Melalui pendekatan ini, game menciptakan pengalaman belajar yang inovatif, termasuk memunculkan emosi positif seperti kebahagiaan dan kesenangan selama proses pembelajaran.

Game digital merupakan bentuk permainan berbasis teknologi yang dapat diakses melalui perangkat elektronik seperti smartphone, komputer, maupun konsol. Dengan visual yang interaktif dan fitur daring, permainan digital mampu menarik minat pengguna, baik secara individu maupun dalam jaringan multiplayer, memberikan pengalaman bermain yang dinamis dan fleksibel (Khoiriyah, 2024).

Dari kutipan di atas dapat disimpulkan, game merupakan media hiburan yang digemari oleh berbagai kalangan karena kemudahannya diakses melalui berbagai perangkat digital modern. Selain sebagai sarana hiburan, game juga dapat dirancang untuk tujuan edukatif, yaitu menyisipkan materi pembelajaran guna mengembangkan kompetensi pemain. Dengan pendekatan yang tepat, game tidak hanya menyenangkan,

tetapi juga mampu menciptakan pengalaman belajar yang efektif dan menyenangkan.

2. *Survival*

Menurut Madana et al. (2021) Game *survival* merupakan sub-genre dari permainan aksi yang menempatkan pemain dalam lingkungan terbuka yang penuh tantangan, di mana kondisi sekitar cenderung tidak bersahabat dan sumber daya sangat terbatas. Pemain biasanya memulai permainan dengan perlengkapan minimal, lalu dituntut untuk bertahan hidup melalui eksplorasi, pengumpulan sumber daya penting, pembuatan alat serta senjata, dan pembangunan tempat berlindung. Fokus utama dari genre ini bukan semata-mata pada pertempuran, melainkan lebih pada bagaimana pemain mengelola keterbatasan dan menghadapi ancaman secara strategis. Beberapa aspek penting yang sering ditonjolkan meliputi pemenuhan kebutuhan dasar seperti makanan, air, dan perlindungan, serta kemampuan adaptasi terhadap kondisi yang berubah-ubah.

Game *Survival* didesain untuk menciptakan pengalaman bertahan hidup yang menantang dan mendekati kenyataan, dengan memanfaatkan kemajuan teknologi termasuk realitas virtual guna memperdalam keterlibatan pemain. Elemen-elemen fundamental seperti pengelolaan sumber daya, pemenuhan kebutuhan pokok, serta interaksi dengan lingkungan yang terus berubah merupakan aspek-aspek kunci dalam genre permainan ini (Habibi & Athoillah, 2025)

Dari kutipan-kutipan di atas dapat disimpulkan bahwa game survival adalah jenis permainan yang menempatkan pemain dalam kondisi penuh tantangan dan keterbatasan, di mana mereka harus bertahan hidup dengan memanfaatkan sumber daya yang tersedia secara strategis. Permainan ini menekankan pada aspek pengelolaan kebutuhan dasar seperti makanan, perlindungan, dan keselamatan, serta interaksi dengan lingkungan yang dinamis dan berubah-ubah. Selain menguji kemampuan adaptasi pemain, game survival juga dirancang untuk menciptakan pengalaman yang imersif dan mendekati realitas, sehingga memberikan sensasi bertahan hidup yang lebih intens dan mendalam.

3. Kerusakan Lingkungan

Aptasari et al., (2025) Menjelaskan Kerusakan lingkungan di wilayah pesisir Indonesia, terutama yang disebabkan oleh pencemaran plastik sekali pakai dan dampak perubahan iklim, telah memicu degradasi ekosistem laut, menurunnya hasil tangkapan ikan, dan ancaman terhadap ketahanan pangan. Selain itu, perubahan suhu laut akibat krisis iklim mengganggu ekosistem perikanan tropis.

Hayati & Ismayani, (2025) menjelaskan bahwa degradasi lingkungan, baik yang bersumber dari fenomena alam maupun intervensi antropogenik, berpotensi menurunkan standar kehidupan serta memberikan dampak negatif pada aspek sosial, ekonomi, dan kesehatan publik.

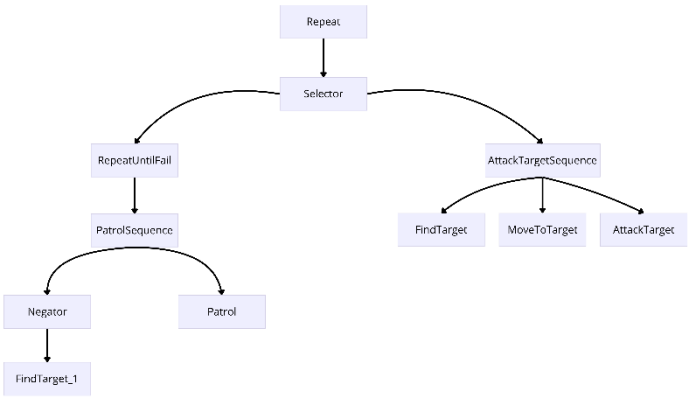
Berdasarkan kutipan tersebut, dapat disimpulkan bahwa kerusakan lingkungan merupakan akibat dari perilaku manusia yang mengeksploitasi

alam secara berlebihan tanpa memperhatikan keberlanjutan, serta dampaknya tidak hanya terbatas pada lingkungan fisik, tetapi juga memengaruhi kualitas hidup manusia secara sosial, ekonomi, dan kesehatan.

4. *Behavior Tree*

Behavior Tree merupakan salah satu metode yang digunakan untuk memetakan perilaku AI dalam pengembangan karakter *non-pemain* (NPC) di dalam gim. Metode ini terdiri dari struktur hierarkis yang tersusun atas *node-node*, seperti *Sequence* dan *Selector*, yang mengatur logika pengambilan keputusan berdasarkan status yang dikembalikan oleh masing-masing node. Metode ini efektif dalam menciptakan perilaku NPC yang adaptif dan modular dalam permainan (Soefana et al., 2021).

Menurut Senoaji et al. (2023) *Behaviour Tree* adalah suatu model matematis berbentuk seperti pohon (tree) yang dirancang untuk menampilkan perilaku dari suatu entitas komputer yang diprogram. Metode ini bersifat fleksibel dan modular, memungkinkan perubahan atau pengembangan perilaku dasar menjadi cabang baru sebagai alternatif pencapaian tujuan. *Behaviour Tree* digunakan untuk mengontrol perilaku NPC (*Non-Playable Character*) agar lebih dinamis dan tidak monoton, berbeda dari metode *Finite State Machine* (FSM) yang kurang fleksibel dalam menangani transisi antar state. Penerapan Behaviour Tree dalam game memungkinkan NPC memiliki perilaku seperti *Idle*, *Patrol*, *Attack*, dan *Death*, yang masing-masing dipecah ke dalam node aksi dan kondisi.



Gambar 2.1 *Behaviour Tree*

Sumber *: https://baldurgames.com/posts/behaviour-trees-godot*

Dari penjelasan di atas dapat di simpulkan *Behaviour Tree* merupakan salah satu metode yang umum digunakan dalam pengembangan kecerdasan buatan untuk karakter non-pemain di dalam game. Metode ini menggunakan struktur hierarkis berbentuk pohon yang terdiri dari berbagai node, seperti *Sequence, Selector, Condition*, dan *Action*, yang masing-masing memiliki peran dalam proses pengambilan keputusan. Setiap node akan mengembalikan status seperti *success, failure*, atau *running* yang menentukan alur eksekusi logika berikutnya.

Keunggulan utama dari Behaviour Tree adalah fleksibilitas dan modularitasnya. Hal ini memungkinkan pengembang untuk memisahkan dan mengatur berbagai perilaku kompleks NPC ke dalam komponen yang lebih kecil dan mudah dikendalikan. Misalnya, perilaku seperti *Idle, Patrol, Attack*, dan *Death* dapat dikelola dalam cabang-cabang berbeda dalam struktur pohon, yang akan dieksekusi berdasarkan kondisi tertentu seperti jarak pemain atau interaksi langsung dengan NPC.

Dibandingkan dengan metode *Finite State Machine* (FSM), *Behaviour Tree* lebih unggul dalam mengelola transisi antar perilaku yang kompleks karena pendekatannya yang lebih terstruktur dan skalabel. FSM sering kali mengalami kesulitan ketika menghadapi banyak transisi state yang saling berkaitan, sementara *Behaviour Tree* mampu mengurangi kompleksitas tersebut melalui sistem node yang terpisah namun saling berhubungan secara logis.

Secara keseluruhan*, Behaviour Tree* menjadi metode yang efektif dalam menciptakan perilaku NPC yang adaptif, responsif terhadap lingkungan, dan lebih realistis dalam interaksinya terhadap pemain

5. Unity

Menurut Wibowo (2022) Unity adalah *game engine* profesional yang digunakan untuk membuat video game pada berbagai platform. Unity tidak hanya bertindak sebagai *game engine* tetapi juga sebagai *Integrated Development Environment* (IDE) yang menyediakan alat lengkap untuk merancang, mengatur aset, serta mengatur logika dan interaktivitas game melalui scripting. Keunggulan Unity terletak pada alur kerja visual yang produktif, sistem komponen modular, serta dukungan lintas platform yang memungkinkan pengembang menciptakan game untuk PC, web, mobile, hingga konsol hanya dengan satu lingkungan pengembangan. Selain itu, Unity juga memiliki sistem visual editor yang intuitif, memungkinkan pengembang untuk membangun dan menguji prototipe game dengan cepat dan efisien.

Unity adalah sebuah game engine yang berfungsi sebagai perangkat lunak pengolah gambar, grafik, suara, input, dan elemen lainnya yang digunakan untuk membuat game, baik berbasis 2D maupun 3D. Unity memiliki keunggulan sebagai platform pengembangan multiplatform, yang memungkinkan hasil proyek dapat dipublikasikan ke berbagai platform seperti *Standalone* (.exe), Web, Android, iOS, hingga konsol seperti XBOX dan PS3. Meskipun untuk platform tertentu memerlukan lisensi khusus, Unity juga menyediakan versi gratis yang cukup lengkap untuk digunakan dalam pengembangan game edukasi, hiburan, maupun simulasi interaktif lainnya (Habdi & Supardi, 2021).

Dari Kutiapn di atas dapat di simpulkan unity merupakan game engine profesional sekaligus *Integrated Development Environment* (IDE) yang menyediakan berbagai alat untuk merancang aset, menyusun logika permainan, dan membangun interaktivitas melalui scripting. Unity mendukung pengembangan game 2D dan 3D serta memiliki sistem kerja visual yang intuitif dan modular. Keunggulan utamanya terletak pada kemampuan multiplatform-nya, yang memungkinkan pengembang mempublikasikan game ke berbagai platform seperti PC, web, Android, iOS, hingga konsol seperti XBOX dan PS3. Unity juga menyediakan versi gratis yang cukup lengkap, menjadikannya pilihan populer untuk pengembangan game edukatif, hiburan, dan simulasi interaktif secara efisien dan fleksibel.

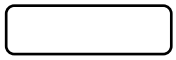
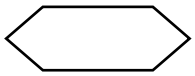
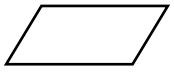
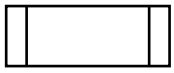
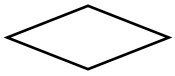
6. *Flowchart*

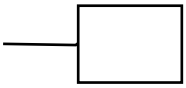
Menurut Dianta (2021) *Flowchart* merupakan representasi grafis yang menggambarkan tahapan dan urutan prosedur dalam suatu program. Alat ini memfasilitasi analis dan programmer dalam mengurai masalah menjadi komponen-komponen lebih sederhana sekaligus mengevaluasi berbagai opsi operasional. Keunggulan *Flowchart* terletak pada kemampuannya untuk merancang dan memvisualisasikan program secara efektif, karena dapat menyajikan logika prosedur kompleks dengan jelas. Selain itu, *Flowchart* berfungsi sebagai alat komunikasi yang efektif berkat penggunaan simbol-simbol standar yang telah diterima secara universal.

Sutanti et al. (2020) menjelaskan bahwa *Flowchart* merupakan representasi grafis yang menampilkan tahapan prosedural dan alur kerja suatu program. Secara praktis, alat ini membantu analis dan programmer dalam memecah masalah kompleks menjadi bagian-bagian yang lebih sederhana sekaligus mengevaluasi berbagai opsi penyelesaian. *Flowchart* sangat bermanfaat untuk menangani masalah yang memerlukan kajian mendalam karena kemampuannya menyajikan alur logika secara visual. Dalam bentuknya, *Flowchart* adalah diagram alir dengan pola sekuensial yang dapat bergerak dalam satu atau dua arah. Fungsi utamanya mencakup perancangan program sekaligus representasi visual dari sistem yang dibuat. Oleh karena itu, *Flowchart* yang baik harus mampu menggambarkan semua komponen pemrograman secara akurat dan lengkap.

Dari kutipan di atas dapat di simpulkan, *Flowchart* adalah representasi grafis dari urutan langkah atau prosedur dalam suatu program yang digunakan untuk membantu analis dan programmer dalam memecah permasalahan menjadi bagian-bagian yang lebih sederhana. Alat ini memudahkan perancangan, evaluasi alternatif solusi, serta visualisasi logika program secara jelas dan sistematis. *Flowchart* menggunakan simbol-simbol standar yang diakui secara universal dan mampu menggambarkan alur kerja program dalam bentuk diagram sekuensial satu arah atau dua arah. Selain sebagai alat bantu desain, *Flowchart* juga berperan penting sebagai media komunikasi yang efektif dalam proses pengembangan sistem.

Tabel 2.1 Simbol *Flowchart*

Simbol	Nama	Keterangan
	*Terminator*	Awal atau akhir program
	*Flow*	Arah Aliran Program
	*Preparation*	Inisialisasi/pemberian nilai awal
	*Process*	Proses/ Pengolahan data
	*Input/output data*	*Input/output* data
	*Sub Program*	Sub program
	*Decision*	Seleksi atau kondisi
	*On Page Connector*	Penghubung bagian-bagian *Flowchart* pada halaman yang sama

	*Comment*	Tentang komentar suatu proses
	*Off Page Connector*	Penghubung bagian-bagian *Flowchart* pada halaman yang beda

7. UML (*Unified Modeling Language*)

Nistrina & Sahidah, (2022) menjelaskan bahwa *Unified Modeling Language* (UML) merupakan bahasa pemodelan standar dalam pengembangan sistem berbasis objek. UML berperan penting dalam memfasilitasi proses spesifikasi dan perancangan sistem *software*, terutama untuk aplikasi yang mengadopsi paradigma object-oriented programming. Bahasa pemodelan ini merupakan sintesis dari berbagai metode pemodelan grafis berbasis objek yang telah berkembang sejak era 1980-an. Keistimewaan UML mencakup tiga aspek utama: kemampuan visualisasi sistem, dokumentasi proses pengembangan, dan generasi kode program secara otomatis. UML menyediakan beragam diagram seperti use case diagram untuk pemetaan interaksi, class diagram untuk representasi struktur kelas, *activity diagram* untuk visualisasi alur kerja, serta *sequence diagram* untuk menggambarkan logika eksekusi sistem. kelas, alur kerja, dan urutan logika sistem.

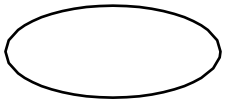
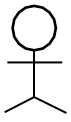
Penelitian lain mengatakan *Unified Modeling Language* (UML) merupakan sebuah bahasa yang divisualisasikan dalam bentuk gambar atau grafik yang berfungsi untuk memberikan gambaran dan spesifikasi dalam pembangunan serta dokumentasi dari sistem berbasis object-oriented. UML

menyediakan standar perancangan sistem seperti konsep proses bisnis, pembuatan kelas (class), rancangan basis data, dan komponen lainnya dalam pengembangan sistem. Diagram UML terdiri dari berbagai jenis, namun yang paling umum digunakan adalah *use case diagram*, *activity diagr*am, dan *sequence diagram*, yang masing-masing menggambarkan interaksi antar aktor, alur aktivitas sistem, serta komunikasi antar objek dalam sistem (Narulita et al., 2024)

a. *Use Case Diagram*

Use case diagram digolongkan sebagai behavior diagram karena menggambarkan alur kerja sistem, baik yang sedang berjalan maupun yang akan dibangun. Diagram ini juga menjelaskan peran dan fungsi setiap aktor dalam berinteraksi dengan sistem yang dirancang (Nistrina & Sahidah, 2022). Berikut merupakan notasi *usecase diagram* yang dapat dilihat pada Tabel

Tabel 2.2 Simbol Use Case Diagram

Simbol	Nama	Keterangan
	*Use Case*	*Use case* merupakan interaksi antara aktor dengan sistem.
	*Actor*	*Actor* merupakan pengguna sistem yang berinteraksi dengan sistem.
	Association	*Association* merupakan penghubung antara aktor dengan *use case* sehingga keduanya dapat saling berinteraksi.

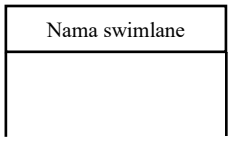
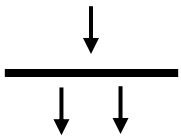
<<extend>>	*Extend*	*Extend* merupakan *use case* lain yang merupakan tambahan dari sebuah *use case* dan bersifat mandiri.
	Generalization	*Generalization* merupakan hubungan dari beberapa use case, dimana dari beberapa use case terdapat *use case* generalisasi.
<<include>>	*Include*	*Include* merupakan hubungan atau relasi antara use case dengan use case tambahan yang akan ikut dieksekusi bersama use case ketika use case dijalankan.

b. Activity Diagram

Activity diagram berfungsi untuk memvisualisasikan alur aktivitas dalam sistem, dengan fokus pada langkah-langkah internal yang dijalankan oleh sistem (Nistrina & Sahidah, 2022). Berikut merupakan notasi *activity diagram* yang dapat dilihat pada Tabel 2.3

Tabel 2.3 Simbol *Activity Diagram*

Simbol	Nama	Keterangan
	Status awal	Status awal merupakan awal dari aktivitas sistem.
Activity	*Activity*	Aktivitas bertujuan untuk menggambarkan aktivitas dari sebuah sistem.
	Decision	Percabangan merupakan aktivitas yang memiliki lebih dari satu kondisi atau pilihan.
	Join	Penggabungan digunakan untuk menghubungkan aktivitas satu dengan aktivitas yang lain.

	Status akhir	Status akhir merupakan akhir dari aktivitas sistem.
	*Swimlane*	Swimlane merupakan penanggung jawab dari setiap aktivitas.
	Line connector	Line Connector digunakan untuk menghubungkan satu simbol dengan simbol lainnya.
	*Fork*	Fork digunakan untuk percabangan secara parallel dari aktivitas.

c. *Class Diagram*

Class diagram merupakan representasi utama dalam pemodelan sistem berorientasi objek yang menggambarkan struktur dari setiap kelas beserta atribut dan operasinya. Diagram ini tidak hanya menampilkan relasi antar kelas, tetapi juga memperlihatkan bagaimana komponen sistem saling terhubung dan berinteraksi secara logis. Selain itu, class diagram membantu pengembang memahami desain sistem secara menyeluruh, terutama dalam mengidentifikasi struktur dan keterkaitan antarkomponen yang akan diimplementasikan dalam perangkat lunak. (Nistrina & Sahidah, 2022). Tabel 2.4 menyajikan simbol-simbol notasi yang umum digunakan dalam class diagram.

Tabel 2.4 Tabel *Class Diagram*

Simbol	Nama	Keterangan
nama_kelas +atribut +operasi()	*Class*	*Class* merupakan struktur dari sebuah sistem
Simbol	**Nama**	**Keterangan**
nama_interface	*Interface*	Menggunakan konsep interface yang sama pada Object Oriented Programming (OOP).
	Association	*Association* merupakan penghubung antara classifiers atau dengan yang lain.
	Asosiasi berarah / *directed association*	Hubungan dari satu kelas dengan kelas yang lain dimana pada kelas lain terdapat atribut yang sama.
	Kebergantungan/*dependency*	Kebergantungan merupakan kelas yang saling bergantung.
	Agregasi/*aggregation*	Agregasi merupakan hubungan antar kelas yang bersifat semua bagian.
	Generalisasi	Generalisasi merupakan hubungan antar kelas yang bersifat umum-khusus.

8. *Game Development Life Cycle* (GDLC)

Menurut Wahyu, (2022) *Game Development Life Cycle* (GDLC) merupakan suatu metode iteratif yang digunakan dalam proses pengembangan game, yang terdiri dari enam tahapan utama, yaitu inisiasi (pembentukan konsep game), pra-produksi, produksi, pengujian, rilis beta, dan peluncuran akhir. Metode ini tidak hanya digunakan untuk memastikan setiap proses berjalan secara sistematis, tetapi juga membantu tim pengembang dalam mengatur alur kerja secara terstruktur agar hasil akhirnya sesuai dengan tujuan yang telah direncanakan. Sementara itu,

Ariyana et al., (2022) menerapkan metode GDLC pada pengembangan game edukatif bertema motif batik khas Yogyakarta. Mereka menegaskan bahwa setiap fase dalam GDLC memberikan kontribusi penting terhadap kualitas hasil akhir, mulai dari tahap perencanaan, pengumpulan aset, pengujian performa, hingga tahap rilis produk. Dengan mengikuti alur pengembangan berdasarkan GDLC, proyek game yang dibangun dapat lebih terarah, terorganisasi, dan memiliki nilai edukatif yang kuat. Oleh karena itu, GDLC menjadi salah satu pendekatan yang relevan dan efektif dalam pengembangan game, khususnya game yang memiliki muatan pesan moral, budaya, atau lingkungan.

Dari kutipan di atas dapat disimpulkan, *Game Development Life Cycle* (GDLC) merupakan suatu pendekatan iteratif yang sangat tepat digunakan dalam pengembangan game, karena mampu mengarahkan proses produksi secara sistematis melalui enam tahapan inti, yaitu inisiasi, pra-

produksi, produksi, pengujian, rilis beta, dan rilis akhir. Dengan struktur tahapan yang jelas, GDLC tidak hanya membantu tim pengembang dalam mengelola proyek secara terorganisir, tetapi juga memastikan bahwa hasil akhir memiliki kualitas yang sesuai tujuan. Metode ini sangat relevan diterapkan pada game-game edukatif maupun yang mengangkat nilai-nilai budaya dan lingkungan, karena mampu mengintegrasikan proses kreatif, teknis, dan evaluatif dalam satu kerangka kerja yang menyeluruh dan efisien.

9. *Black Box*

Wicaksono (2021) *Black Box Testing* adalah metode pengujian perangkat lunak yang menguji sistem dari sisi fungsional tanpa memperhatikan struktur internal atau logika program yang ada di balik layar. Pengujian ini berfokus pada input dan output dari perangkat lunak, bukan bagaimana proses tersebut terjadi di dalamnya. *Black Box* testing bertujuan untuk memastikan bahwa perangkat lunak berjalan sesuai dengan spesifikasi yang telah ditentukan, dan metode ini sangat efektif untuk menemukan kesalahan dalam fungsi sistem

Menurut Sultansyah et al. (2025) Metode pengujian black box berfokus pada pemeriksaan input dan output sistem tanpa memperhatikan struktur internal program. Pendekatan ini dinilai efektif dalam menilai keandalan fitur serta kesesuaian hasil keluaran sistem dengan kebutuhan pengguna. Dalam konteks layanan publik digital, pengujian ini penting untuk memastikan bahwa sistem merespons dengan benar terhadap berbagai

skenario penggunaan, tanpa harus mengevaluasi kode program secara langsung.

10. *System Usavility Scale* (SUS)

System Usability Scale (SUS) merupakan metode evaluasi kebergunaan yang dikenal karena kepraktisannya. SUS bersifat cepat dan mudah digunakan, terdiri dari sepuluh pernyataan sederhana yang menghasilkan skor tunggal antara 0–100. Hal ini memudahkan tim pengembang dalam memahami tingkat usability dari suatu sistem (Kaban et al., 2020)

Selain itu, menurut Kesuma (2021) keunggulan SUS juga terletak pada sifatnya yang efisien secara waktu dan biaya, terutama jika dilaksanakan secara online. Meskipun sederhana, SUS tetap memberikan hasil evaluasi yang valid secara statistik dan cukup akurat untuk menggambarkan tingkat kegunaan sistem.

Dari kutipan di atas disimpulkan bahwa *System Usability Scale* (SUS) merupakan metode evaluasi kebergunaan yang sangat sesuai untuk digunakan dalam pengujian perangkat lunak karena bersifat sederhana, cepat, dan hemat biaya. SUS mampu menghasilkan data yang valid secara statistik dan mudah dipahami baik oleh pengguna maupun tim pengembang, bahkan ketika dilaksanakan secara daring. Kepraktisan ini menjadikan SUS sebagai salah satu alat evaluasi yang efisien dan tetap dapat diandalkan dalam berbagai konteks pengujian sistem.

B. Kajian Empiris

Penelitian yang dilakukan oleh Senoaji et al. (2023) berjudul "Implementasi Behaviour Tree Untuk Mengatur Perilaku NPC Musuh Pada Game 2D Platformer Cyberun" bertujuan untuk mengembangkan kecerdasan buatan pada karakter musuh (NPC) dalam game 2D platformer menggunakan metode *Behavior Tree*. Dalam penelitian ini, *Behavior Tree* digunakan untuk memetakan berbagai node perilaku musuh, seperti *Idle, Patrol, Attack,* dan *Death*. Setiap jenis musuh memiliki kombinasi node yang berbeda dan dirancang untuk merespon pemain berdasarkan jarak serta kondisi tertentu. Hasil penelitian menunjukkan bahwa metode *Behavior Tree* mampu menghasilkan perilaku NPC yang lebih dinamis dan tidak monoton, sesuai dengan parameter seperti posisi pemain dan aksi yang dilakukan. Selain itu, pengujian dilakukan menggunakan *Game Experience Questionnaire* (GEQ) terhadap beberapa responden untuk mengevaluasi pengalaman bermain. Hasilnya, game dinilai cukup menarik dan mudah dimainkan, dengan AI musuh yang memberikan tantangan secara adaptif. Penelitian ini membuktikan bahwa *Behavior Tree* merupakan pendekatan yang efektif dalam pengembangan AI musuh pada game 2D.

Penelitian yang dilakukan oleh Setiawan et al. (2023) mengembangkan game bergenre roguelike berjudul “The Last Hope” yang mengimplementasikan dua metode kecerdasan buatan: Algoritma dan *Behavior Tree* untuk mengatur perilaku NPC musuh. Algoritma digunakan agar musuh dapat memilih jalur terdekat menuju pemain tanpa menabrak

objek, sementara *Behavior Tree* diterapkan untuk menyusun perilaku musuh menjadi lebih adaptif dan terstruktur. NPC dalam game tersebut memiliki beberapa perilaku seperti stationary (diam di tempat), persue (mengejar pemain saat masuk radius tertentu), dodge (menghindar dari rintangan), dan unique skill (melancarkan serangan khusus). Penelitian ini membuktikan bahwa kombinasi Algoritma A dan *Behavior Tree* mampu membentuk AI yang fleksibel, menyesuaikan tindakan berdasarkan jarak dan kondisi lingkungan sekitar. Hasil pengujian menunjukkan bahwa game berjalan dengan baik dan perilaku musuh terasa dinamis serta menantang bagi pemain. Penelitian ini relevan dengan studi implementasi *Behavior Tree* dalam pembuatan AI musuh untuk menciptakan pengalaman bermain yang lebih imersif dan edukatif.

Penelitian oleh Soefana et al. (2021) berjudul "Penerapan *Behavior Tree* untuk Pengambilan Keputusan *Non-Player Character* pada Gim Balap" bertujuan mengimplementasikan metode *Behavior Tree* untuk mengatur perilaku NPC dalam game bergenre balapan. Penelitian ini menggunakan *game engine* Unity 3D dengan template *The Karting Microgame* sebagai basis pengembangan. Perilaku NPC dirancang dengan struktur node *Behavior Tree* yang terdiri dari *Repeater, Sequence*, dan *Selector*, serta dipadukan dengan sistem *Waypoint* dan *Raycasting* sebagai mekanisme navigasi dan deteksi rintangan di lintasan balap. Melalui pendekatan ini, NPC dapat melakukan berbagai tindakan seperti maju, mundur, belok, dan menghindar dari rintangan berdasarkan kondisi lingkungan sekitar. Pengujian dilakukan melalui metode *Black Box*, pengujian FPS, serta analisis jumlah tabrakan. Hasil menunjukkan

bahwa game dapat berjalan stabil pada rata-rata 60 FPS meskipun jumlah NPC ditambah. *Behavior Tree* juga berhasil membuat pengambilan keputusan NPC menjadi lebih dinamis dan sesuai dengan kondisi permainan. Penelitian ini menunjukkan bahwa *Behavior Tree* merupakan metode yang efektif dan fleksibel untuk mengatur perilaku NPC, terutama dalam lingkungan permainan yang membutuhkan respons adaptif terhadap lingkungan sekitar.

Penelitian yang dilakukan oleh Anang & Ibnu (2025) berfokus pada pengembangan game *Survival* horor berbasis Unity 3D dengan penerapan kecerdasan buatan (AI) pada *karakter non-pemain* (NPC). Dalam penelitian ini, NPC musuh dirancang agar dapat merespons kehadiran pemain dengan perilaku yang adaptif, seperti menyerang ketika pemain berada dalam radius tertentu, dan mengejar menggunakan sistem navigasi *Waypoint* serta algoritma pencarian jalur (*pathfinding*).

Perilaku musuh dalam game ini diatur melalui pendekatan *state machine*, yang terdiri atas berbagai kondisi seperti diam, patroli, menyerang, hingga mati. Game juga mengadopsi atmosfer menyeramkan yang terinspirasi dari Resident Evil dan Silent Hill, serta mengintegrasikan elemen psikologi ketakutan ke dalam *gameplay*. Pengujian dilakukan secara teknis (*Black Box*) dan melalui umpan balik pengguna, yang menyatakan bahwa AI musuh mampu memberikan tantangan yang sesuai dan menciptakan pengalaman bermain yang imersif. Fitur-fitur seperti sistem kesehatan, interaksi dengan sumber daya, dan respons musuh terhadap serangan pemain, berhasil diimplementasikan dengan baik. Penelitian ini menunjukkan bahwa penerapan

AI berbasis logika kondisi (state-based) mampu meningkatkan kualitas gameplay dalam game *Survival* berbasis Unity.

Penelitian yang dilakukan oleh Junaidi et al. (2021) berjudul "Implementasi Behavior Tree pada Perilaku NPC di Game Sidescroller", bertujuan untuk meningkatkan kecerdasan buatan NPC dalam game bergenre side-scroller. Dalam game ini, pemain bergerak ke satu arah untuk menyelesaikan misi, sementara NPC dirancang memiliki perilaku kompleks seperti patroli, mengejar, melarikan diri, menyerang dengan skill khusus, dan meledakkan diri. Penelitian menggunakan algoritma Behavior Tree (BT) sebagai metode utama pengambilan keputusan untuk NPC. BT dipilih karena kemampuannya yang modular, mudah dibaca, dan fleksibel dibandingkan pendekatan FSM atau HFSM. Elemen penting BT yang diterapkan meliputi Leaf Node, Condition Node, Selector, dan Sequence. Uji coba dilakukan dalam tiga tahap: pengujian logika unit, pengujian perilaku melalui Unity Console, dan User Acceptance Test (UAT) dari 10, 20, hingga 80 responden. Hasil menunjukkan bahwa penggunaan BT berhasil meningkatkan adaptivitas perilaku NPC dan menambah tantangan dalam permainan. Setelah BT diterapkan, aspek UI/UX mencapai 84,91%, efek suara (SFX & BGM) 80,8%, serta tingkat kesulitan meningkat hingga 86,45%. Secara keseluruhan, penerapan Behavior Tree terbukti efektif dalam mengatur perilaku musuh secara dinamis dan menambah daya tarik permainan.

C. Kerangka Berpikir

Gambar 2.2 Kerangka Berpikir