

Implementasi Behavior Tree pada Hierarki Perilaku Musuh Game TPS 3D

Rizal Maulana Syahrul Jamil¹, Kusnadi², Dzulfikar Fickri Rosyid³

^{1,2}Program Studi Teknik Informatika, Fakultas Teknologi Informasi, Universitas Catur Insan Cendekia, Indonesia

³Program Studi Desain Komunikasi Visual, Fakultas Teknologi Informasi, Universitas Catur Insan Cendekia, Indonesia

¹rizal.jamil.ti.22@cic.ac.id, ²kusnadi@cic.ac.id, ³dzulfikar.fickrirosyid@cic.ac.id

Abstrak

Karakter musuh berbasis kecerdasan buatan pada *game third-person shooter* memerlukan hierarki perilaku yang eksplisit agar patroli, pengejaran, dan serangan dipilih sesuai kondisi permainan. Penelitian ini bertujuan merancang, mengimplementasikan, dan mengevaluasi *Behavior Tree* pada *game* 3D *The Flooded Corruption*. Pengembangan mengikuti *Game Development Life Cycle* yang mencakup *concept*, *pre-production*, *production*, *testing*, dan *post-production*. Sistem dibangun menggunakan *Unreal Engine* 5.6.1 melalui integrasi *Root Selector*, cabang *Combat* dan *Passive*, *Blackboard*, *AI Perception Sight*, *Blueprint Task*, serta *Navigation Mesh*. Evaluasi dilakukan melalui pengujian *black-box* pada tiga kondisi operasional: *AttackTarget* valid dengan jarak maksimal 100 *Unreal Unit* menghasilkan *Attack*, *AttackTarget* valid pada jarak lebih dari 100 sampai 2.000 *Unreal Unit* menghasilkan *Chase*, dan *AttackTarget* tidak valid pada jarak lebih dari 2.000 *Unreal Unit* menghasilkan *Patrol*. Setiap skenario diulang 30 kali untuk mengamati kemungkinan inkonsistensi eksekusi akibat *timing runtime*, sehingga diperoleh 90 keputusan. Kesesuaian keluaran terhadap *ground truth* diukur menggunakan *Decision Accuracy* dan *Error Rate*. Seluruh keputusan sesuai dengan aksi yang diharapkan, menghasilkan *Decision Accuracy* 100% dan *Error Rate* 0% pada skenario yang diuji. Hasil tersebut menunjukkan konsistensi fungsional dan keterlacakan alur persepsi-data-keputusan-aksi, bukan akurasi prediktif atau generalisasi ke kondisi di luar pengujian. Kontribusi penelitian berupa arsitektur implementasi terintegrasi dan prosedur evaluasi keputusan yang dapat diaudit serta direplikasi.

Kata kunci: *behavior tree*, kecerdasan buatan *game*, *decision accuracy*, *game development life cycle*, *third-person shooter*, *unreal engine*.

1. Pendahuluan

Kecerdasan buatan pada *game* digunakan untuk membentuk respons karakter non-pemain terhadap kondisi lingkungan dan tindakan pemain. Pada genre *third-person shooter* (TPS), musuh harus beralih secara tepat antara patroli, pengejaran, dan serangan. Kesalahan transisi dapat menghasilkan aksi yang tidak logis, seperti menyerang di luar jangkauan atau terus mengejar ketika *target* tidak lagi tersedia. Keterbacaan respons tersebut penting karena persepsi pemain terhadap kecerdasan lawan terbentuk melalui pengamatan atas gerak dan keputusan karakter selama permainan (Al Enezi & Verbrugge, 2023; del Bosque et al., 2025).

Behavior Tree (BT) menyediakan arsitektur keputusan hierarkis yang memisahkan *node* kontrol, kondisi, dan aksi *terminal*. *Selector* mengevaluasi cabang berdasarkan prioritas, *Sequence* menjalankan *child* secara berurutan, *Decorator* membatasi eksekusi, sedangkan *Task* mewujudkan aksi. Struktur ini banyak digunakan karena modular, dapat ditelusuri, dan memungkinkan perubahan perilaku tanpa membangun ulang seluruh logika (Iovino et al.,

2022; Partlan et al., 2022).

Penelitian terdahulu menerapkan BT pada *game sidescroller* (Junaidi et al., 2021), *game Android* yang menggabungkan A* dan BT (Setiawan et al., 2023), *platformer* 2D (Senoaji et al., 2023), *serious game* Warik's Adventure (Siswoko et al., 2023), serta gim balap (Soefana et al., 2021); pada kelompok studi ini, evaluasi umumnya bersifat fungsional atau kualitatif pada *game* 2D maupun *mobile*, tanpa pengukuran kuantitatif berulang yang terhubung langsung ke data persepsi. Pendekatan lain mengukur keputusan *Non-Player Character* (NPC) melalui *utility-based* AI dan algoritma (Nugroho et al., 2024), yakni evaluasi berbasis kebugaran statistik melalui *ELO rating* dan korelasi *Pearson*, bukan verifikasi deterministik terhadap *ground truth* yang eksplisit. Perilaku NPC dan pencarian rute pada lingkungan 3D juga dapat dievaluasi melalui hubungan antara kondisi uji dan keluaran sistem, tetapi berfokus pada optimasi rute menggunakan algoritma A*, bukan hierarki keputusan *Patrol-Chase-Attack* (Saeroji & Riska, 2026; Zulkarnain et al., 2024). Ketiga kelompok studi tersebut, yaitu BT fungsional pada *game* 2D/*mobile*, AI berbasis utilitas dengan evaluasi statistik, dan algo-

ritma pencarian rute pada lingkungan 3D, menunjukkan pola yang sama: pengukuran kuantitatif berulang belum ditautkan langsung pada hierarki keputusan eksplisit dalam satu arsitektur yang terintegrasi. Dalam kumpulan studi yang ditinjau, integrasi *AI Perception*, *Blackboard*, *Behavior Tree*, *Blueprint Task*, dan *Navigation Mesh* pada TPS 3D berbasis *Unreal Engine 5* yang disertai verifikasi kuantitatif berulang terhadap keputusan *Patrol*, *Chase*, dan *Attack* belum dilaporkan secara bersamaan.

Kesenjangan tersebut relevan untuk ditutup karena keterlacakan dan replikasi keputusan AI musuh menjadi semakin penting seiring bertambahnya kompleksitas perilaku pada *game* TPS 3D yang bergantung pada persepsi sensor waktu nyata; tanpa arsitektur yang mengintegrasikan persepsi, penyimpanan kondisi, dan hierarki keputusan sekaligus disertai prosedur verifikasi yang eksplisit, kesesuaian antara desain dan implementasi BT sulit diaudit maupun direplikasi pengembang lain. Penelitian ini bertujuan mengimplementasikan hierarki *Patrol-Chase-Attack* pada *game* *The Flooded Corruption* dan mengukur kesesuaian keputusan terhadap *ground truth*. Kontribusinya meliputi: (1) arsitektur terintegrasi yang memisahkan persepsi, penyimpanan kondisi, evaluasi prioritas, dan aktuasi; (2) pemetaan kondisi operasional yang dapat diaudit; serta (3) prosedur verifikasi keputusan menggunakan *Decision Accuracy* (DA) dan *Error Rate* (ER) pada tiga skenario deterministik. Penelitian tidak mengklaim algoritma BT baru atau keunggulan komputasi terhadap *Finite State Machine* maupun *utility-based AI* karena perbandingan langsung tidak dilakukan. Ruang lingkupnya juga tidak mencakup pengalaman pemain, optimalitas rute, koordinasi multiagen, atau performa komputasi.

2. Metode Implementasi dan Pengujian

Metode penelitian menggabungkan *Game Development Life Cycle* (GDLC) sebagai kerangka pengembangan sistem *Behavior Tree* pada *game* TPS 3D *The Flooded Corruption* menggunakan *Unreal Engine 5.6.1*. Tahapan GDLC mencakup penetapan konsep permainan, perancangan struktur perilaku dan aturan keputusan, implementasi komponen sistem, pengujian fungsional, serta pengemasan aplikasi. Arsitektur yang dibangun mengintegrasikan *AI Perception Sight* untuk mendeteksi pemain, *Blackboard* untuk menyimpan kondisi sistem, *Behavior Tree* dan *Blueprint Task* untuk menentukan serta menjalankan aksi, dan *Navigation Mesh* untuk menyediakan area pergerakan karakter musuh. Integrasi tersebut membentuk alur persepsi-data-keputusan-aksi yang memisahkan proses pendeteksian, penyimpanan kondisi, evaluasi prioritas, dan aktuasi perilaku. Pemisahan antarkomponen memudahkan penelusuran perubahan data hingga keluaran aksi.

Pengujian *black-box* dijalankan pada tiga skenario operasional berdasarkan validitas *AttackTarget*,

State pada *Blackboard*, dan jarak karakter musuh terhadap pemain. Setiap skenario dipetakan terlebih dahulu ke *ground truth* berupa aksi *Attack*, *Chase*, atau *Patrol*, kemudian diulang sebanyak 30 kali sehingga diperoleh total 90 keputusan. Pengulangan dilakukan untuk mengamati konsistensi eksekusi sistem terhadap kemungkinan variasi *runtime*, bukan sebagai sampel statistik untuk generalisasi. Kesesuaian setiap keputusan aktual terhadap *ground truth* diukur menggunakan *Decision Accuracy* dan *Error Rate* sebagai indikator verifikasi fungsional implementasi *Behavior Tree*.

2.1 Rancangan Penelitian

Penelitian ini menggunakan pendekatan rekayasa perangkat lunak eksperimental dengan *game* TPS 3D *The Flooded Corruption* sebagai lingkungan implementasi. Peta *Ascendant* memuat enam instans musuh dengan *Behavior Tree* identik.

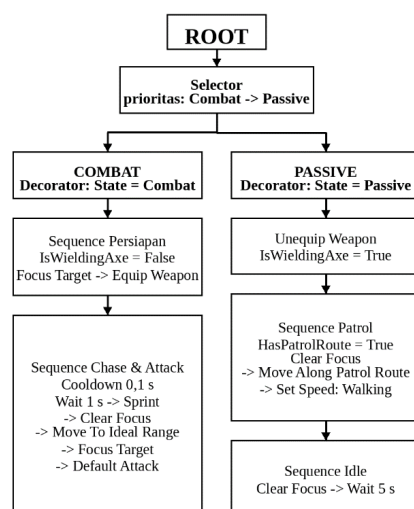
Unit analisis adalah satu keputusan *Patrol*, *Chase*, atau *Attack* per percobaan; koordinasi antarinstantans tidak dievaluasi. Variabel masukan mencakup validitas *AttackTarget*, *State* pada *Blackboard*, dan jarak (1 UU = 1 cm), sedangkan variabel keluaran berupa aksi yang tereksekusi.

Pengembangan mengikuti *Game Development Life Cycle* (GDLC) yang iteratif antara rancangan perilaku, aset, implementasi, dan pengujian, dengan keluaran setiap tahap menjadi masukan langsung bagi tahap berikutnya. Tahap *concept* menetapkan genre TPS dan tiga perilaku musuh (*Patrol*, *Chase*, *Attack*) yang menjadi acuan cabang keputusan BT pada tahap berikutnya. *Pre-production* menerjemahkan perilaku tersebut ke dalam *Behavior Tree*, *Blackboard*, parameter persepsi, dan *ground truth*, sehingga setiap *node* dan parameter yang ditetapkan pada tahap ini memiliki padanan langsung pada implementasi. *Production* mengimplementasikan rancangan tersebut menjadi AI *Controller*, *Blueprint Task*, dan *Navigation Mesh* pada *Unreal Engine 5.6.1*, sehingga kesesuaian antara Gambar 1 (rancangan) dan Gambar 2 (implementasi) dapat diperiksa langsung. *Testing* menjalankan tiga skenario *black-box* dengan 30 pengulangan untuk memverifikasi apakah aksi hasil implementasi sesuai dengan *ground truth* yang ditetapkan pada *pre-production*; pada siklus pengembangan sebelum pengujian formal, temuan pada tahap ini menjadi umpan balik untuk menyelaraskan *production* dan *pre-production*, sebagaimana tercermin pada kesesuaian sempurna yang dilaporkan pada Tabel 3. Rincian lengkap disajikan pada Tabel 1.

Tahap *post-production* hanya mencakup pengemasan build *Windows 64-bit* dan distribusi aplikasi, tanpa pengukuran performa eksternal. Pembatasan ini menjaga evaluasi tetap fokus pada kebenaran logika *Behavior Tree*, bukan variasi perangkat keras. Dengan demikian, setiap keputusan dapat ditelusuri dan diukur konsistensinya melalui metrik yang telah dite-

Tabel 1. Penerapan *Game Development Life Cycle*

Tahap	Aktivitas dan Keluaran
Concept	Menetapkan genre TPS, tujuan pemain, lingkungan uji, serta perilaku Patrol-Chase-Attack.
Pre-production	Merancang BT, Blackboard, AI Perception, aset, peta, antarmuka, dan ground truth.
Production	Mengimplementasikan AI Controller, BT, Blueprint Task, animasi, dan Navigation Mesh pada Unreal Engine 5.6.1.
Testing	Menjalankan tiga skenario black-box, masing-masing 30 pengulangan, kemudian menghitung DA dan ER.
Post-production	Mengemas build Windows 64-bit sebagai aplikasi mandiri.

Gambar 1. Rancangan Asli Struktur *Behavior Tree* Karakter Musuh

tapkan.

2.2 Arsitektur dan Konfigurasi Sistem

Arsitektur sistem memisahkan sensor, penyimpanan data, evaluasi prioritas, dan aktuasi. *AI Perception Sight* membentuk stimulus *target*; *EnemyAI Controller* memperbarui *AttackTarget* dan *State*; *Blackboard* menjadi sumber data bersama; *Decorator* memilih cabang *Combat* atau *Passive*; *Task* menjalankan fokus, pergerakan, persiapan senjata, serangan, atau patroli; dan *Navigation Mesh* menyediakan area lintasan. Hubungan komponen dan aturan keputusan ditunjukkan pada Gambar 1.

Root Selector mengevaluasi cabang *Combat* lebih dahulu agar *target* valid segera diproses. Di dalam *Combat*, *AttackRadius* sebesar 100 UU membedakan *Attack* dan *Chase*, sedangkan cabang *Passive* menjadi *fallback* ketika *target* tidak valid dan menjalankan *Patrol*. *Blackboard* menyimpan *SelfActor*, *AttackTarget*, *State*, *PointOfInterest*, *AttackRadius*, dan *DefendRadius*; *DefendRadius* disiapkan untuk pengembangan lanjutan dan tidak digunakan dalam eksperimen. *AI Perception Sight* dikonfigurasi den-

Tabel 2. *Ground Truth* Keputusan Karakter Musuh

Kode	Kondisi Operasional	State	Aksi
S1	<i>AttackTarget</i> valid; jarak $\leq$ 100 UU	<i>Combat</i>	<i>Attack</i>
S2	<i>AttackTarget</i> valid; 100 UU < jarak $\leq$ 2.000 UU	<i>Combat</i>	<i>Chase</i>
S3	<i>AttackTarget</i> tidak valid; jarak > 2.000 UU	<i>Passive</i>	<i>Patrol</i>

gan *Sight Radius* 2.000 UU, *Lose Sight Radius* 2.400 UU, *Peripheral Vision Half Angle* 60 derajat, dan *Max Age* 5 detik. *Sight Radius* dan *AttackRadius* ditetapkan dengan mempertimbangkan dimensi dan proporsi karakter terhadap skala dunia permainan, sehingga jarak deteksi dan jarak serang menghasilkan respons AI yang realistis dan tidak terlalu agresif maupun terlalu pasif; *AttackRadius* 100 UU secara khusus mengacu pada jangkauan efektif animasi serangan *melee* karakter musuh. *Lose Sight Radius*, *Peripheral Vision Half Angle*, dan *Max Age* ditetapkan bersamaan pada tahap *pre-production* sebagai bagian dari konfigurasi *AI Perception*, tanpa pengujian sensitivitas terpisah untuk masing-masing nilai. Nilai tersebut merupakan konfigurasi implementasi, tetapi 90 keputusan formal hanya mengevaluasi tiga rentang jarak pada Tabel 2. Pengaruh rentang 2.000-2.400 UU, sudut pandang, oklusi, dan retensi stimulus tidak diuji secara terpisah.

2.3 *Ground Truth* dan Prosedur Pengujian

Ground truth ditetapkan sebelum pengujian berdasarkan kombinasi validitas *AttackTarget*, *State*, dan jarak yang ditetapkan pada tahap perancangan. Ketiga kondisi pada Tabel 2 bersifat saling eksklusif pada pengujian formal, sehingga setiap masukan dipetakan ke satu aksi yang diharapkan dan keputusan aktual dapat diklasifikasikan sebagai Sesuai atau Tidak Sesuai. Ketiga kondisi tersebut diturunkan langsung dari ambang yang telah ditetapkan pada konfigurasi BT dan *AI Perception* (Bagian 2.2): *AttackRadius* 100 UU menjadi batas antara *Attack* dan *Chase*, sedangkan *Sight Radius* 2.000 UU menjadi batas validitas *AttackTarget* antara *Chase* dan *Patrol*. Keduanya ditetapkan berdasarkan proporsi karakter terhadap skala dunia permainan. Dengan menu-runkan *ground truth* dari ambang yang sama dengan yang mengendalikan percabangan BT, penilaian Sesuai/Tidak Sesuai menguji kesesuaian implementasi terhadap spesifikasinya sendiri.

Pengujian dilakukan melalui *Play-In-Editor* (PIE). Setiap skenario diulang 30 kali sehingga menghasilkan 90 keputusan. Pengulangan digunakan sebagai *repeated functional runs* untuk mengamati apakah faktor *runtime*, seperti siklus *physics tick*, *State* mesin animasi, dan variasi *frame timing*, menimbulkan ketidakkonsistenan evaluasi cabang atau eksekusi aksi meskipun sistem bersifat deterministik; pengulangan

ini bukan sampel statistik untuk generalisasi.

Pada setiap percobaan, pemain ditempatkan sesuai kondisi skenario, sistem dibiarkan berjalan sampai *AttackTarget* dan *State* stabil, aksi instans yang diamati dicatat, kemudian dibandingkan dengan *ground truth*. Klasifikasi hasil tetap berbasis hubungan masukan-keluaran *black-box*, sedangkan *debugger Blackboard* dan *Behavior Tree* hanya digunakan untuk keterlacakan ketika memeriksa status eksekusi (Gunawan et al., 2023).

$$DA = (N_{sesuai}/N_{total}) \times 100\% \quad (1)$$

$$ER = 100\% - DA \quad (2)$$

DA menyatakan proporsi keputusan yang sesuai terhadap aturan, sedangkan ER menyatakan proporsi ketidaksesuaian. Kedua metrik digunakan sebagai indikator verifikasi fungsional, bukan akurasi prediktif *machine learning*.

Pengujian dijalankan pada *Windows 11* 64-bit dengan AMD Ryzen 7 5700G, RAM 32 GB, NVIDIA GeForce RTX 3060 12 GB, dan *Unreal Engine 5.6.1*. *Microsoft Excel 2016* digunakan untuk rekapitulasi data. Seluruh percobaan menggunakan peta, konfigurasi BT, parameter persepsi, dan nilai *AttackRadius* yang sama; pencatatan diarahkan pada instans AI yang sedang diamati.

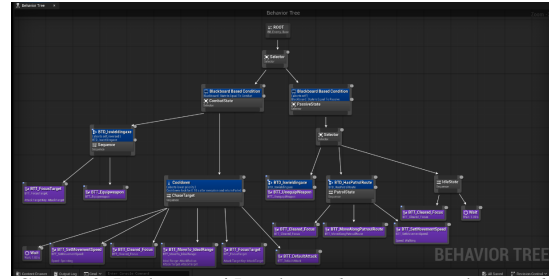
3. Hasil dan Pembahasan

Bagian ini menyajikan hasil implementasi *Behavior Tree* beserta integrasi komponen pendukung, realisasi perilaku *Patrol*, *Chase*, dan *Attack*, serta hasil pengujian kesesuaian keputusan terhadap *ground truth*. Pembahasan juga mencakup perbandingan dengan penelitian terdahulu, implikasi arsitektur terhadap keterlacakan sistem, ancaman validitas, prosedur replikasi, dan rancangan pengujian lanjutan. Penyajian mengikuti alur implementasi hingga evaluasi agar keterkaitan setiap hasil tetap jelas.

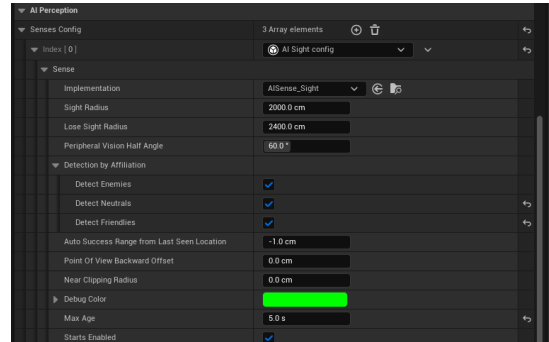
3.1 Implementasi Struktur *Behavior Tree*

Implementasi pada *Behavior Tree Editor* mempertahankan pemisahan cabang *Combat* dan *Passive*. *Combat* mencakup persiapan senjata, pengejaran, fokus *target*, dan serangan, sedangkan *Passive* menangani pelepasan senjata, *Patrol*, dan *Idle*. Gambar 2 memperlihatkan struktur implementasi lengkap untuk memeriksa kesesuaian urutan prioritas dengan rancangan pada Gambar 1. Pemisahan *node* kontrol, kondisi, dan Task memungkinkan perubahan fungsi lokal tanpa menyusun ulang seluruh pohon.

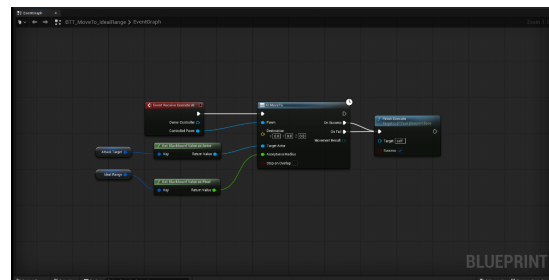
Decorator pada cabang utama membaca *State* dari *Blackboard*. Ketika *AI Perception* mengenali pemain, *Controller* mengisi *AttackTarget* dan mengubah *State* menjadi *Combat*. Ketika stimulus tidak lagi valid, referensi *target* dikosongkan dan *State* kembali *Passive*. Aliran data satu arah tersebut mengurangi ketergantungan langsung antartask dan memudahkan



Gambar 2. Implementasi Lengkap *Behavior Tree* pada *Unreal Engine 5.6.1*



Gambar 3. Konfigurasi *AI Perception Sight*



Gambar 4. *Blueprint Task BTT_MoveTo_IdealRange*

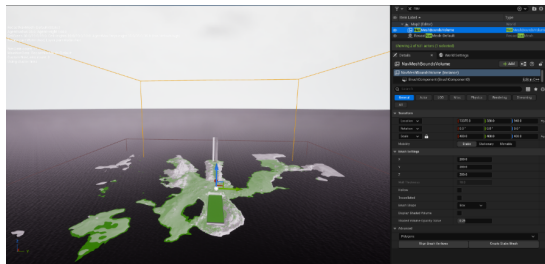
pemeriksaan melalui *debugger Blackboard*.

3.2 Integrasi Persepsi, Pergerakan, dan Navigasi

Konfigurasi *AI Perception* pada Gambar 3 menentukan pembentukan dan retensi *target*. *Sight Radius* 2.000 UU menjadi batas deteksi yang digunakan pada *ground truth*, sedangkan *Lose Sight Radius* 2.400 UU, *Max Age* 5 detik, dan *Peripheral Vision Half Angle* 60 derajat merupakan parameter implementasi tambahan. Efek parameter tambahan tersebut belum menjadi variabel pada 90 keputusan yang dilaporkan; karena itu, naskah tidak menafsirkan hasil DA sebagai bukti ketahanan terhadap batas retensi, perubahan sudut pandang, atau oklusi.

Blueprint Task BTT_MoveTo_IdealRange pada Gambar 4 membaca *AttackTarget* dan *AttackRadius* dari *Blackboard*, kemudian meneruskan lokasi *target* dan *acceptable* radius ke fungsi *AI MoveTo*. BT menentukan kapan *Task* dijalankan, sedangkan *Task* mengubah keputusan menjadi perintah gerak. Pemisahan ini mendukung modularitas BT (Iovino et al., 2022).

Navigation Mesh pada Gambar 5 menyediakan permukaan valid untuk *Patrol* dan *Chase*. Area



Gambar 5. *Navigation Mesh* pada Peta *Ascendant*



Gambar 6. Rute *Waypoint Patrol* dan Instans AI pada Peta *Ascendant*



Gambar 7. Realisasi Perilaku *Chase* pada Sudut Pandang Permainan

laut tanpa *NavMesh* tidak dapat dilalui sehingga AI tetap berada pada ruang navigasi yang dirancang. Penelitian ini memeriksa perpindahan perilaku, tetapi tidak membandingkan panjang jalur atau waktu komputasi dengan algoritma pencarian rute lain seperti A* (Saeroji & Riska, 2026; Zulkarnain et al., 2024).

3.3 Realisasi dan Hasil Pengujian Perilaku

Keluaran BT diwujudkan melalui lintasan, animasi, dan pergerakan karakter. Gambar 6 mendokumentasikan rute *waypoint Patrol* dan posisi instans AI pada peta *Ascendant*, sedangkan Gambar 7 dan Gambar 8 menunjukkan realisasi *Chase* dan *Attack* dari sudut pandang permainan. Pada *State Passive*, AI mengikuti rute *Patrol*. Ketika *AttackTarget* valid pada jarak lebih dari 100 sampai 2.000 UU, AI menjalankan *Chase*; ketika jarak mencapai atau kurang dari 100 UU, cabang *Attack* dijalankan.

Ketiga gambar berfungsi sebagai dokumentasi implementasi, bukan dasar tunggal klasifikasi keputusan. Penentuan Sesuai atau Tidak Sesuai menggunakan kondisi masukan, nilai *State* dan *AttackTar*



Gambar 8. Eksekusi Animasi *Attack* pada Jarak Serang

Tabel 3. Rekapitulasi Hasil Pengujian

Kode	Aksi	Sesuai	Tidak Sesuai	DA	ER
S1	Attack	30	0	100%	0%
S2	Chase	30	0	100%	0%
S3	Patrol	30	0	100%	0%
Total	–	90	0	100%	0%

get yang stabil, serta aksi yang dieksekusi pada tiap percobaan. Penelitian tidak mengukur pengalaman, kepuasan, atau persepsi pemain terhadap kualitas perilaku (AI Enezi & Verbrugge, 2023).

Rekapitulasi pada Tabel 3 menunjukkan bahwa S1 selalu menghasilkan *Attack*, S2 menghasilkan *Chase*, dan S3 menghasilkan *Patrol*. Seluruh 90 keputusan sesuai dengan *ground truth*, sehingga DA mencapai 100% dan ER 0% pada tiga kondisi yang diuji.

Hasil 100% merupakan keluaran yang diharapkan ketika implementasi deterministik mengeksekusi pemetaan satu-ke-satu pada Tabel 2 tanpa deviasi. Nilai tersebut menunjukkan konsistensi implementasi terhadap spesifikasi, bukan bukti bahwa AI optimal, adaptif, atau unggul atas arsitektur lain. Variasi yang belum diuji meliputi nilai batas, oklusi kompleks, perubahan *frame rate*, lebih dari satu *target*, interaksi beberapa instans AI, serta kualitas taktis dan pengalaman pemain.

3.4 Perbandingan, Implikasi, dan Validitas

Kontribusi penelitian ini bukan terletak pada penciptaan *Behavior Tree* sebagai metode baru, melainkan pada penerapan dan dokumentasi arsitektur *Behavior Tree* secara terintegrasi dalam game TPS 3D berbasis *Unreal Engine 5.6.1*. Integrasi tersebut mencakup AI *Perception Sight*, *Blackboard*, *Behavior Tree*, *Blueprint Task*, dan *Navigation Mesh* untuk membentuk alur persepsi, penyimpanan kondisi, pengambilan keputusan, dan pelaksanaan aksi karakter musuh. Posisi penelitian terhadap karya terdahulu diringkas pada Tabel 4 dengan meninjau konteks game, metode kecerdasan buatan, bentuk evaluasi, serta komponen sistem yang digunakan. Penyajian perbandingan tersebut bertujuan memperjelas perbedaan ruang lingkup dan fokus implementasi penelitian ini terhadap penelitian sebelumnya. Perbandin

Tabel 4. Perbandingan dengan Penelitian Terkait

Penelitian	Konteks	Evaluasi	Pembeda
(Junaidi et al., 2021)	BT, <i>sidescroller</i>	Fungsional	<i>Unity</i> dan <i>game</i> 2D
(Setiawan et al., 2023)	A* dan BT, <i>Android</i>	Fungsional	Menggabungkan navigasi A*
(Senoaji et al., 2023)	BT, <i>platformer</i> 2D	Fungsional	<i>Idle</i> , <i>Patrol</i> , <i>Attack</i> , <i>Death</i>
(Nugroho et al., 2024)	<i>Utility AI</i> , <i>fighting game</i>	Kuantitatif	Optimasi dengan algoritma genetika
Penelitian ini	BT, <i>Unreal Engine 5</i> , TPS 3D	DA dan ER; 90 keputusan	Integrasi persepsi- <i>Blackboard</i> -BT- <i>NavMesh</i>

gan dilakukan secara konseptual berdasarkan informasi yang dilaporkan dalam masing-masing publikasi, bukan melalui *benchmark* langsung, pengujian ulang, atau pengukuran performa pada lingkungan eksperimen yang sama. Oleh karena itu, Tabel 4 tidak digunakan untuk menyimpulkan keunggulan metode, melainkan untuk menunjukkan posisi dan karakteristik kontribusi penelitian secara proporsional.

Nilai *Decision Accuracy* 100% dan *Error Rate* 0% pada penelitian ini perlu dimaknai dengan hati-hati ketika disandingkan dengan penelitian terdahulu, karena sebagian besar studi pembandingan tidak melaporkan metrik akurasi keputusan yang sepadan. Senoaji et al. (2023) dan Setiawan et al. (2023) melaporkan keberhasilan transisi antarstate secara fungsional tanpa persentase kesesuaian terhadap *ground truth*, sedangkan Nugroho et al. (2024) melaporkan kinerja melalui *ELO rating* dan korelasi Pearson yang mengukur kekuatan relatif AI, bukan proporsi keputusan yang sesuai aturan. Dengan demikian, DA 100% pada penelitian ini bukan bukti superioritas terhadap studi-studi tersebut, melainkan konsekuensi logis dari desain pengujian *black-box* pada sistem deterministik: karena aturan *Combat-Passive* dan ambang *AttackRadius* bersifat tetap dan tidak melibatkan komponen stokastik atau pembelajaran, kesesuaian sempurna terhadap *ground truth* yang berasal dari aturan yang sama merupakan hasil yang diharapkan, sebagaimana juga ditunjukkan oleh sifat *rule-based* pada Junaidi et al. (2021). Kontribusi pembeda penelitian ini terletak pada cakupan integrasi kelima komponen sekaligus serta prosedur pengulangan 30 kali per skenario yang belum diterapkan pada studi pembandingan, bukan pada nilai metrik itu sendiri.

Dari sisi rekayasa perangkat lunak, pemisahan tanggung jawab menyediakan jalur diagnosis yang terstruktur: kegagalan deteksi diperiksa pada sensor, kesalahan State pada *Controller* atau *Blackboard*, kesalahan prioritas pada *Decorator*, dan kegagalan *aktuasi* pada *Task* atau *Navigation Mesh*. Temuan ini menunjukkan manfaat keterlacakan pada implementasi yang diuji, tetapi tidak membuktikan efisiensi

komputasi atau kemudahan pemeliharaan yang lebih tinggi dibandingkan *Finite State Machine* karena pengukuran komparatif tidak dilakukan. Berbeda dengan pendekatan *fuzzy* yang membentuk perilaku adaptif (Amrulloh et al., 2025), penelitian ini mempertahankan aturan deterministik agar hubungan kondisi dan aksi dapat diaudit. Cabang defensif, pemilihan *target*, atau skor utilitas dapat ditambahkan pada penelitian lanjutan (Chergarova et al., 2024; Nugroho et al., 2024). Ancaman validitas internal berasal dari pencatatan manual, variasi kondisi awal, dan keberadaan instans AI lain pada peta. Mitigasi dilakukan dengan memakai build, peta, konfigurasi *Blackboard*, parameter, dan prosedur pembentukan kondisi yang sama serta mengarahkan pencatatan pada instans yang sedang diamati. Dampak interferensi antarinstans tidak diukur, sehingga hasil tidak dapat ditafsirkan sebagai evaluasi koordinasi multiagen. Validitas konstruk terbatas karena DA hanya memeriksa kecocokan aksi, bukan kualitas taktis, waktu respons, konsumsi komputasi, atau pengalaman pemain. Validitas eksternal dibatasi oleh satu tipe musuh dan satu peta; karena itu, kesimpulan hanya berlaku pada konfigurasi yang diuji.

3.5 Keterlacakan dan Replikasi Implementasi

Keterlacakan implementasi dijaga melalui pemisahan stimulus, penyimpanan kondisi, evaluasi prioritas, dan pelaksanaan aksi. Nilai *AttackTarget* dan *State* dapat diperiksa pada *debugger Blackboard*, sedangkan cabang aktif diamati pada *Behavior Tree Editor* selama PIE. *Debugger* digunakan untuk mengonfirmasi sumber keputusan, sementara klasifikasi akhir tetap ditentukan dari hubungan kondisi skenario dan aksi aktual.

Replikasi memerlukan tipe *Blackboard key*, struktur prioritas cabang, *AttackRadius* 100 UU, *Sight Radius* 2.000 UU, peta, dan prosedur pembentukan kondisi yang sama. Setiap percobaan dimulai dengan menempatkan pemain pada rentang skenario, menunggu *AttackTarget* dan *State* stabil, mengamati instans AI yang ditetapkan, mencatat aksi aktual, kemudian mereset posisi sebelum pengulangan berikutnya. Perubahan *Lose Sight Radius*, *Max Age*, sudut pandang, atau *acceptable radius* dapat mengubah waktu transisi walaupun struktur BT tidak berubah.

Prosedur tersebut juga menyediakan pola diagnosis bertahap. Jika *target* tidak terbentuk, pemeriksaan diarahkan pada AI *Perception* dan garis pandang. Jika *target* valid tetapi cabang tidak berubah, pemeriksaan diarahkan pada *Controller*, *Blackboard*, dan *Decorator*. Jika cabang aktif tetapi karakter tidak bergerak atau menyerang, pemeriksaan dilanjutkan pada *Blueprint Task*, animasi, dan *Navigation Mesh*. Pola ini merupakan implikasi praktis dari keterlacakan arsitektur, bukan hasil pengukuran efisiensi.

3.6 Rancangan Pengujian Lanjutan

Pengujian lanjutan perlu mencakup nilai batas 99, 100, dan 101 UU untuk *AttackRadius* serta nilai di sekitar 2.000 dan 2.400 UU untuk membedakan batas deteksi dan retensi. Variasi sudut pandang, penghalang, dan durasi *Max Age* juga perlu diuji agar perubahan validitas *AttackTarget* tidak hanya diwakili oleh jarak. Penambahan kondisi tersebut akan memperluas cakupan masukan tanpa mengubah aturan dasar BT.

Evaluasi berikutnya dapat mengukur waktu respons, stabilitas pada beberapa *frame rate*, konsumsi komputasi, beberapa *target*, dan koordinasi banyak instans AI. Perbandingan terkontrol dengan *Finite State Machine* atau *utility-based AI* perlu menggunakan skenario dan metrik yang sama agar klaim modularitas, biaya komputasi, dan kualitas keputusan dapat diuji secara empiris. Pengujian pengguna dapat dilakukan terpisah untuk menilai apakah konsistensi fungsional juga menghasilkan perilaku yang mudah dipahami dan menantang.

4. Kesimpulan

Behavior Tree berhasil mengorganisasi *Patrol*, *Chase*, dan *Attack* melalui prioritas *Combat-Passive* yang membaca data bersama pada *Blackboard*. Integrasi *AI Perception Sight*, *EnemyAI Controller*, *Blueprint Task*, dan *Navigation Mesh* membentuk alur persepsi-data-keputusan-aksi pada *Unreal Engine 5.6.1*. Pengujian *black-box* terhadap tiga kondisi jarak dengan 30 pengulangan per skenario menghasilkan 90 keputusan yang seluruhnya sesuai dengan *ground truth*, sehingga DA mencapai 100% dan ER 0%. Hasil ini menunjukkan kesesuaian fungsional dan konsistensi eksekusi pada konfigurasi yang diuji, bukan akurasi prediktif, adaptivitas *universal*, atau keunggulan terhadap arsitektur AI lain. Kontribusi penelitian terletak pada dokumentasi arsitektur terintegrasi, pemetaan aturan yang dapat diaudit, dan prosedur verifikasi berulang yang dapat direplikasi. Penelitian selanjutnya perlu menguji nilai batas, oklusi dan retensi persepsi, waktu respons, variasi *frame rate*, beberapa *target*, koordinasi *multiagen*, pengalaman pemain, serta perbandingan terkontrol dengan *Finite State Machine* dan *utility-based AI*.

Daftar Pustaka

- Al Enezi, W., & Verbrugge, C. (2023): *Investigating the Influence of Behaviors and Dialogs on Player Enjoyment in Stealth Games*, Proceedings of the AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment, 19(1), 166–174. <https://doi.org/10.1609/aiide.v19i1.27512>
- Amrulloh, E. S., Krisdiawan, R. A., Lesmana, I., & Rohmawati, L. (2025): *Fuzzy-Driven Adaptive NPC Behavior in a Meme-Based Platformer Game for Android Mobile*, Nuansa Informatika, 19(2), 123–132. <https://doi.org/10.25134/ilkom.v19i2.440>
- Chergarova, V., Tomeo, M., Morgan, H., & Andrade, S. (2024): *Integrating Artificial Intelligence (AI) into Game Development to Elevate Diverse Gameplay Elements*, Issues in Information Systems, 25(2), 408–417. https://doi.org/10.48009/2_iis_2024_132
- del Bosque, A., Fernández-Arias, P., Lampropoulos, G., & Vergara, D. (2025): *The Role of Artificial Intelligence in Gaming*, Applied Sciences, 15(23), 12358. <https://doi.org/10.3390/app152312358>
- Gunawan, R., Wibisono, Y. P., Primasari, C. H., Budiyo, D., & Cininta, M. (2023): *Blackbox Testing on Virtual Reality Gamelan Saron Using Equivalence Partition Method*, Jurnal Buana Informatika, 14(01), 11–19. <https://doi.org/10.24002/jbi.v14i01.6606>
- Iovino, M., Scukins, E., Styruud, J., Ögren, P., & Smith, C. (2022): *A Survey of Behavior Trees in Robotics and AI*, Robotics and Autonomous Systems, 154, 104096. <https://doi.org/10.1016/j.robot.2022.104096>
- Junaidi, A., Yunus, A., & Wiguna, A. S. (2021): *Implementasi Behavior Tree pada Perilaku NPC di Game Sidescroller*, Kurawal - Jurnal Teknologi, Informasi dan Industri, 4(2), 92–103. <https://doi.org/10.33479/kurawal.v4i2.459>
- Nugroho, S., Affan, L. Y., & Purnomo, M. H. (2024): *An Improved Utility-Based Artificial Intelligence to Capture NPC Behaviour in Fighting Games Using Genetic Algorithm*, Jurnal Nasional Pendidikan Teknik Informatika (JANAPATI), 13(2), 394–404. <https://doi.org/10.23887/janapati.v13i2.82040>
- Partlan, N., Soto, L., Howe, J., Shrivastava, S., Seif El-Nasr, M., & Marsella, S. (2022): *Evolving Behavior: Towards Co-Creative Evolution of Behavior Trees for Game NPCs*, In K. Karpouzis, S. Gualeni, J. Pirker, & A. Fowler (Eds.), FDG '22: Proceedings of the 17th International Conference on the Foundations of Digital Games (pp. 1–13). Association for Computing Machinery. <https://doi.org/10.1145/3555858.3555896>
- Saeroji, R. A. A., & Riska, S. Y. (2026): *Optimisasi Algoritma A* untuk Pencarian Rute Menggunakan Media Roblox*, Jurnal Informatika Polinema, 12(2), 279–286. <https://doi.org/10.33795/jip.v12i2.9232>
- Senoaji, A. H., Haryanto, H., & Novianto, S. (2023): *Implementasi Behaviour Tree untuk Mengatur Perilaku NPC Musuh pada Game 2D Platformer 'Cyberun'*, Techno Creative, 1(2), 88–95. <https://doi.org/10.62411/tcv.v1i2.2061>
- Setiawan, M. L., Arbansyah, & Suryawan, S. H. (2023): *Penerapan Algoritma A* dan Behaviour Trees untuk Perilaku Non-Player Character (NPC) pada Game "The Last Hope"*

- Berbasis Android Menggunakan Unity 2D*, Jurnal CoSciTech (Computer Science and Information Technology), 4(2), 451–460. <https://doi.org/10.37859/coscitech.v4i2.5418>
- Siswoko, T. M., Haryanto, H., Hastuti, K., & Kadiasti, R. (2023): *Non-Playable Character (NPC) Based on Behaviour Tree for Enhanced Immersive Experience in the Serious Game “Warik’s Adventure.”*, Journal of Applied Informatics and Computing, 7(2), 284–290. <https://doi.org/10.30871/jaic.v7i2.6753>
- Soefana, F., Jonemaro, E. M. A., & Akbar, M. A. (2021): *Penerapan Behavior Tree untuk Pengambilan Keputusan Non-Player Character pada Gim Balap*, Jurnal Pengembangan Teknologi Informasi dan Ilmu Komputer, 5(5), 1886–1890.
- Zulkarnain, I. A., Astuti, I. P., & Turuzi, R. F. L. (2024): *Rancang Bangun Non-Playable Character dalam Game Berbasis 3D Menggunakan Metode Algoritma A**, Jurnal Informatika Polinema, 11(1), 7–12. <https://doi.org/10.33795/jip.v11i1.5323>