ANALISIS DAN IMPLEMENTASI ALGORITMA TACTICAL PATHFINDING UNTUK NON-PLAYER CHARACTER DALAM PERMAINAN 3D

SKRIPSI

Untuk memenuhi sebagian persyaratan untuk mencapai gelar Sarjana Komputer

IQRA AHMADYA

NIM. 0810680041

KEMENTERIAN PENDIDIKAN DAN KEBUDAYAAN

UNIVERSITAS BRAWIJAYA

PROGRAM TEKNOLOGI INFORMASI DAN ILMU KOMPUTER

MALANG

2013

LEMBAR PERSETUJUAN

ANALISIS DAN IMPLEMENTASI ALGORITMA TACTICAL PATHFINDING UNTUK NON-PLAYER CHARACTER DALAM PERMAINAN 3D

Disusun oleh :

IQRA AHMADYA

NIM. 0810680041

Skripsi ini telah disetujui oleh dosen pembimbing pada tanggal 22 Mei 2013

Dosen Pembimbing I	**Dosen Pembimbing II**
Himawat Aryadita, ST., M.Sc.	**Eriq M. Adams J., ST., M.Kom.**
NIP. 19801018 200801 1 003	**NIK. 85041006110027**

LEMBAR PENGESAHAN

ANALISIS DAN IMPLEMENTASI ALGORITMA TACTICAL PATHFINDING UNTUK NON-PLAYER CHARACTER DALAM PERMAINAN 3D

Disusun oleh:

IQRA AHMADYA

NIM. 0810680041

Skripsi ini telah diuji dan dinyatakan lulus pada tanggal 7 Juni 2013

Dosen Penguji I	Dosen Penguji II
Dr. Eng Herman Tolle, ST., MT.	**Denny Sagita R., S.Kom., M. Kom.**
NIP. 19740823 200012 1 001	**NIK. 851124 06 1 1 0250**

Dosen Penguji III

Novanto Yudistira, S.Kom., M.Sc.
NIK. 831110 16 1 1 0425

Mengetahui,

Ketua Program Studi Teknik Informatika

Drs. Marji, M.T.
NIP. 19670801 199203 1 001

PERNYATAAN
ORISINALITAS SKRIPSI

Saya menyatakan dengan sebenar-benarnya bahwa sepanjang pengetahuan saya, di dalam naskah SKRIPSI ini tidak terdapat karya ilmiah yang pernah diajukan oleh orang lain untuk memperoleh gelar akademik di suatu perguruan tinggi dan tidak terdapat karya atau pendapat yang pernah ditulis atau diterbitkan oleh orang lain, kecuali yang secara tertulis dikutip dalam naskah ini dan disebutkan dalam sumber kutipan dan daftar pustaka.

Apabila ternyata didalam naskah SKRIPSI ini dapat dibuktikan terdapat unsur-unsur PLAGIASI, saya bersedia SKRIPSI ini digugurkan dan gelar akademik yang telah saya peroleh (SARJANA) dibatalkan, serta diproses sesuai dengan peraturan perundang-undangan yang berlaku. (UU No. 20 Tahun 2003, Pasal 25 ayat 2 dan Pasal 70)

Malang, Juli 2013

Mahasiswa,

Iqra Ahmadya
0810680041

KATA PENGANTAR

Puji syukur Alhamdulillaah penulis panjatkan kehadirat Allah SWT karena hanya dengan rahmat, kasih dan karunia-Nya, dan shalawat serta salam kepada Nabi Muhammad SAW beserta keluarganya, penulis telah menyelesaikan skripsi dengan judul "Analisis Dan Implementasi Algoritma Tactical Pathfinding Untuk Non-Player Character Dalam Permainan 3D" dengan baik. Skripsi ini diajukan untuk memenuhi persyaratan kelulusan dalam menyelesaikan pendidikan dan memperoleh gelar Sarjana Komputer di Program Studi Teknik Informatika Program Teknologi Informasi dan Ilmu Komputer Universitas Brawijaya Malang.

Dalam menyelesaikan skripsi ini, penulis banyak mendapat bantuan dan dukungan dari berbagai pihak. Oleh karena itu penulis ingin menyampaikan ucapan terima kasih yang sebesar-besarnya kepada :

1. Bapak Heru Susilo dan Ibu Sri Hartini selaku orang tua penulis beserta saudara-saudaraku yang telah memberikan dukungan dan bantuan baik lahir maupun batin, dan doa yang senantiasa teriring demi terselesaikannya skripsi ini.
2. Bapak Himawat Aryadita, ST., M.Sc. selaku Dosen Pembimbing I dan penasihat akademik dan Bapak Eriq M. Adams J., ST., M.Kom. selaku Dosen Pembimbing II yang telah memberikan banyak arahan, masukan, dan bimbingannya kepada penulis selama proses penyelesaian skripsi ini.
3. Bapak Ir. Sutrisno, MT. selaku Ketua Program PTIIK, Bapak Drs. Marji, MT. dan Bapak Issa Arwani, S.Kom, M.Sc. selaku Ketua dan Sekretaris Program Studi Teknik Informatika.
4. Seluruh Dosen Jurusan Teknik Informatika dan Civitas Akademika Program Teknologi Informasi dan Ilmu Komputer Universitas Brawijaya yang telah banyak memberi bantuan dan dukungan selama penulis menempuh masa studi di Teknik Informatika Universitas Brawijaya dan selama penyelesaian skripsi ini.
5. Febri Abdullah, M. Aminul Akbar, Andreas Tommy, Ayok Luthfi, Lintang G.P., yang telah memberikan banyak bantuan dalam proses pengerjaan skripsi ini.

6. Teman-teman angkatan 2008 Teknik Informatika yang telah memberikan bantuan dan pengalaman selama menjadi mahasiswa di Universitas Brawijaya.
7. Semua pihak yang tidak dapat penulis sebutkan satu per satu yang terlibat baik secara langsung maupun tidak langsung demi terselesaikannya skripsi ini, bersama segala sesuatu yang menyusul setelahnya.

Penulis menyadari bahwa skripsi ini masih banyak kekurangan dan jauh dari sempurna, karena keterbatasan materi dan pengetahuan yang dimiliki penulis. Maka, saran dan kritik yang membangun dari semua pihak sangat diharapkan demi penyempurnaan selanjutnya. Semoga skripsi ini dapat bermanfaat dan berguna bagi semua pihak, baik penulis maupun pembaca.

Malang, Juli 2013

Penulis

ABSTRAK

Iqra Ahmadya. 2013. Analisis Dan Implementasi Algoritma Tactical Pathfinding Untuk Non-Player Character Dalam Permainan 3D. Skripsi Program Studi Teknik Informatika, Program Teknologi Informasi dan Ilmu Komputer, Universitas Brawijaya.
Dosen Pembimbing : Himawat Aryadita ST., M.Sc. dan Eriq M. Adams J. ST., M.Kom.

Pencarian jalur merupakan salah satu implementasi kecerdasan buatan dalam permainan. Pencarian jalur terpendek merupakan hal yang mempengaruhi pergerakan dan pengambilan keputusan pada *non-player character*. Namun, jalur terpendek belum tentu dan tidak selalu menjadi jalur paling aman. Dalam permainan berbasis militer, karakter dituntut untuk bergerak secara taktis dalam menghadapi ancaman. Agen yang bergerak secara taktis dalam pencarian jalur tidak hanya mencari jalur terpendek, namun harus mempertimbangkan ancaman karena pertimbangan *hit points*, demi meningkatkan kesan nyata pada permainan.

Tactical Pathfinding merupakan salah satu algoritma pencarian jalur yang dapat melakukan pencarian jalur terpendek dengan perhitungan bobot ancaman. Implementasi algoritma *tactical pathfinding* dapat memberikan gerakan taktis pada *non-player character*. Algoritma *Tactical Pathfinding* dilakukan berdasarkan algoritma pencarian jalur berdasarkan A* ditambah perhitungan bobot.

Implementasi algoritma dilakukan dengan melakukan simulasi pada peta permainan 3D berbasis *navigation mesh*. Representasi peta permainan 3D menggunakan *navigation mesh* karena dalam beberapa tahun terakhir, *navigation mesh* menjadi pilihan utama. Implementasi algoritma dilakukan dengan melakukan pemberian bobot ancaman pada daerah *navigation mesh* sehingga mempengaruhi agen dalam memilih jalur menuju tujuan. Simulasi implementasi algoritma dibedakan menjadi 3 skenario. Simulasi dilakukan dengan *game engine* Unity versi 3.5.

Pengujian *frame rate* pada 20 uji coba dalam 3 skenario simulasi, menunjukkan bahwa algoritma *Tactical Pathfinding* ini menggunakan memori yang lebih banyak daripada pencarian jalur terpendek biasa walaupun tidak terlalu signifikan. Sedangkan untuk pengujian *hit points* pada 20 uji coba dalam 3 skenario simulasi, menunjukkan bahwa algoritma *Tactical Pathfinding* dapat mengurangi *damage* yang diterima oleh agen *non-player character* walaupun tidak terlalu signifikan.

Kata kunci : pencarian jalur, *tactical pathfinding*, *navigation mesh*, permainan komputer

ABSTRACT

Iqra Ahmadya. 2012. *Analysis and Implementation of Tactical Pathfinding Algorithm for Non-Player Character in 3D Game. Program of Informatics Technology and Computer Science, Brawijaya University.*
Advisors: Himawat Aryadita ST., M.Sc. *and* Eriq M. Adams J. ST., M.Kom.

Pathfinding is one of the implementation of artificial intelligence in games. Shortest pathfinding is a matter that affects non-player character movement and decision making. However, shortest path does not mean and does not always the safest path. In a military-based game, character be charged to move and making decision tactically. When agents move tactically, they are not only seek for the shortest path, but the agents must consider the threat since there any hit points consideration, for the sake of increasing the reality matter in game.

Tactical Pathfinding is a one of pathfinding algorithm that can implement shortest pathfinding with considering weights treat. The implementation of tactical pathfinding can provide non-player character with tactical movement. Tactical pathfinding is implemented based on A pathfinding with weights threat addition.*

The algorithm implemented with doing a simulation on 3D navigation mesh based game maps. Game maps will represent in 3D navigation mesh, because navigation mesh become the primary world representation in recent years. Algorithm is implemented by giving the navigation mesh region with threat weight, so that it change the agent pathfinding decision to find a goal. Algorithm is implemented in 3 different simulation scenarios. Simulation will be implemented in Unity game engine version 3.5.

Frame-rate testing of the 20 test in 3 simulation scenario shows that tactical pathfinding algorithm using more memory than traditional shortest pathfinding, although it is not very significant. Whereas, in health points testing in 15 simulation scenarios, shows that tactical pathfinding can reduces the damages received by the non-player character agent although it is not very significant.

Key words: pathfinding, tactical pathfinding, navigation mesh, computer game

DAFTAR ISI

DAFTAR TABEL

DAFTAR GAMBAR

DAFTAR LAMPIRAN

BAB I
PENDAHULUAN

1.1 Latar Belakang

Bermain merupakan pergerakan bebas dalam struktur yang lebih kaku [SCH-08:28]. Pergerakan bebas yang terstruktur tersebut menimbulkan sebuah konflik, yang disebut *game*. *Game* merupakan sistem di mana pemain terlibat dalam konflik buatan, ditentukan oleh aturan, dan menghasilkan hasil yang terukur [ZIM-03:11]. Perkembangan ilmu pengetahuan dan teknologi membuat permainan tersebut menjadi lebih menarik, menjadi sebuah *video game*, salah satunya adalah permainan 3D.

Video game merupakan perangkat lunak yang membutuhkan algoritma dalam penanaman kecerdasan buatan pada *non-player character*. Fungsi dari kecerdasan buatan adalah untuk membuat *non-player charater* dapat berjalan secara otomatis, serta mampu mengambil keputusan dan tindakan. Pencarian jarak terpendek merupakan konsep dalam pembuatan sistem kecerdasan buatan.

A* merupakan algoritma pencarian jarak terpendek yang bermaksud mengambil langkah terpendek dalam tiap pergerakannya dengan menggabungkan kelebihan dari algoritma Dijkstra dan *Best-First-Search* [NIU-08]. Kecerdasan buatan yang ditanamkan pada *video game* yang menggunakan peta berbasis 3D diharuskan memiliki algoritma yang lebih efektif. Halangan pada permainan 3D, tidak ditemui di peta berbasis 2D [NIU-08]. Situasi ini tampaknya tidak memungkinkan jika menggunakan solusi algoritma A* tradisional [NIU-08].

Selain halangan, faktor yang harus diperhatikan saat melakukan pencarian jalur dalam permainan 3D adalah faktor ancaman. Faktor ancaman sering diabaikan oleh sebagian permainan 3D sebagai pertimbangan dalam pencarian jalur. Pengabaian tersebut berakibat mengurangi nilai *hit points*

milik *non-player character*, dan merugikan pemain. Algoritma A* yang tradisional tidak memperhitungkan faktor ancaman tersebut. Terdapat berbagai macam penyempurnaan A* yang telah dirancang, salah satunya adalah *tactical pathfinding*.

Tactical pathfinding menambahkan pertimbangan faktor ancaman, jarak dan halangan. *Tactical pathfinding* yang mempertimbangkan situasi taktis dalam pencarian jalur, dibandingkan dengan A* yang hanya mencari jalur terpendek atau tercepat [MIL-09:507]. Oleh karena itu, implementasi algoritma *tactical pathfinding* akan membuat kecerdasan buatan *non-player character* menjadi lebih taktis dalam melakukan pencarian jalur.

1.2 Rumusan Masalah

Berdasarkan pada permasalahan yang telah dijelaskan pada bagian latar belakang, maka rumusan masalah dapat disusun sebagai berikut:

a. Mengimplementasikan *Tactical Pathfinding* dalam permainan 3D.
b. Mengetahui pengaruh implementasi *Tactical Pathfinding* terhadap rata-rata *hit points* yang diterima *non-player character* dalam permainan 3D.
c. Mengetahui pengaruh implementasi *Tactical Pathfinding* terhadap rata-rata *frame per second* dalam permainan 3D.

1.3 Batasan Masalah

Lingkup aspek kajian dalam penelitian ini sebagai berikut:

a. Representasi peta permainan berupa *navigation mesh* statis.
b. *Offline mesh generation.*
c. Peta permainan tidak berbasis *real-time*.
d. Halangan pada peta adalah halangan statis.

1.4 Tujuan

Berdasarkan pada rumusan masalah yang telah dijelaskan, maka tujuan yang ingin dicapai dari skripsi ini adalah merancang implementasi *tactical*

pathfinding dan mengetahui pengaruhnya terhadap *hit points* yang diterima *non-player character* dan rata-rata *frame per second* dalam permainan 3D.

1.5 Manfaat

Manfaat yang nantinya dapat diambil dari pengembangan perangkat lunak ini adalah:

a. Bagi Penulis

Dapat lebih memahami tentang pengembangan algoritma *tactical pathfinding* pada *non-player character* dalam permainan 3D.

b. Bagi Pengembang *Game*

Dapat memudahkan pengembang *game* apabila ingin menggunakan algoritma *tactical pathfinding* sebagai kecerdasan buatan untuk ditanamkan kepada *non-player character* pada permainan 3D.

1.6 Sistematika Penulisan

Sistematika penulisan laporan tugas akhir ini adalah sebagai berikut:

Bab I Pendahuluan

Memuat latar belakang, rumusan masalah, tujuan, batasan masalah, manfaat, metodologi pembahasan, dan sistematika penulisan.

Bab II Dasar Teori

Menguraikan teori dasar dan teori penunjang yang berkaitan dengan kecerdasan buatan pada *game*, algoritma *tactical pathfinding* berdasarkan algoritma A*, dan parameter pengujian.

Bab III Metode Penelitian

Membahas metode yang digunakan dalam penelitian yang terdiri dari pengumpulan dataset peta permainan, implementasi algoritma *tactical pathfinding*, pengujian algoritma serta pengambilan kesimpulan dan saran.

Bab IV Perancangan

Membahas tentang perancangan implementasi dari algoritma *tactical pathfinding*.

Bab V Implementasi

Membahas tentang implementasi dari algoritma *tactical pathfinding*.

Bab VI Pengujian dan Analisis

Memuat proses dan hasil pengujian terhadap algoritma yang telah direalisasikan dalam simulasi berdasarkan skenario.

Bab VII Penutup

Memuat kesimpulan serta saran yang diperoleh dari pembuatan dan pengujian sistem untuk pengembangan lebih lanjut.

BAB II
TINJAUAN PUSTAKA

Bab ini membahas dasar teori yang digunakan untuk menunjang penulisan skripsi mengenai *Tactical Pathfinding* untuk *Non-Player Character* dalam Permainan 3D. Beberapa dasar teori yang dimaksud adalah kecerdasan buatan pada *game*, *tactical pathfinding*, peta permainan dengan *navigation mesh* dan parameter pengujian.

2.1 Kecerdasan Buatan pada *Game*

Kecerdasan memiliki beragam jenis dan derajat yang dialami oleh manusia, sebagian besar binatang dan beberapa mesin [MCJ-07]. Kecerdasan buatan adalah membuat komputer dapat melakukan tugas berpikir seperti yang dapat dilakukan oleh hewan dan manusia [MIL-09:04]. Kecerdasan buatan merupakan sains dan teknik untuk membuat sebuah mesin yang cerdas, khusunya program komputer cerdas [MCJ-07]. Secara historis, terdapat beberapa pendekatan mengenai arti dari kecerdasan buatan seperti dijabarkan dalam Gambar 2.1 [RUS-10].

Kecerdasan buatan diaplikasikan ke berbagai macam disiplin ilmu, seperti klasifikasi *heuristic*, sistem pakar, *computer vision*, pengenalan suara, dan *game* [MCJ-07]. Kecerdasan buatan pada *game* memiliki sebuah model. Model tersebut terbagi menjadi tiga bagian: pergerakan, pengambilan keputusan dan strategi seperti yang digambarkan dalam Gambar 2.2 [MIL-09]. Kecerdasan buatan yang meliputi pergerakan (*movement*) dan pengambilan keputusan (*decision making*) menggunakan algoritma yang berkerja pada basis per-karakter, dan sisanya digunakan secara menyeluruh [MIL-09:09].

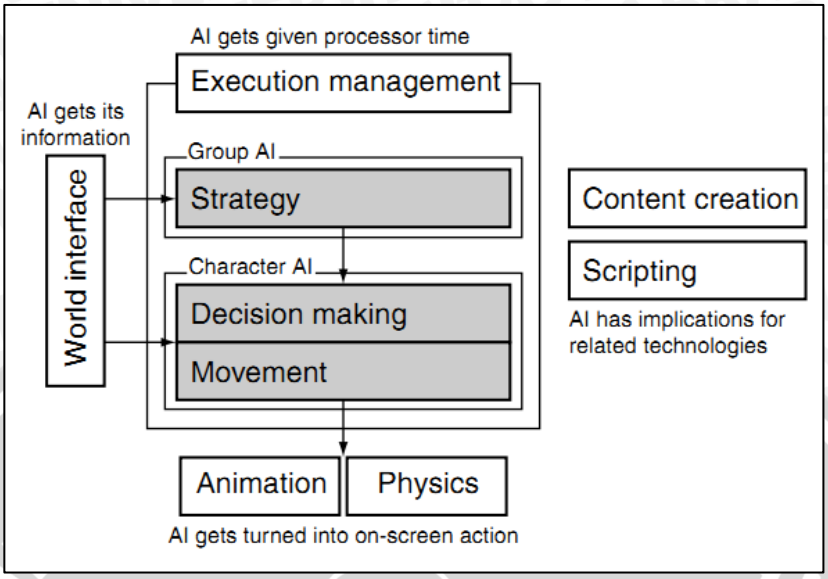
Kecerdasan buatan pada *game* untuk bertarung di semua area di peta, dibutuhkan pada *game* masa kini untuk mendukung pemain yang bertarung, pasukan yang diperintah oleh pemain, dan pertarungan *multiplayer* [DAL-09]. Tugas kecerdasan buatan tersebut harus dirancang semirip mungkin dengan manusia sehingga diharuskan memenuhi taktik dan hukum militer secara

otentik, meskipun hal tersebut komplek dan berada di lingkungan dinamis [DAL-09].

Berpikir Manusiawi	**Berpikir Rasional**
"Usaha yang menarik untuk membuat komputer berpikir . . . *mesin dengan pikiran,* dalam arti sebenarnya." (Haugeland, 1985) "[Otomatisasi dari] aktivitas yang berhubungan dengan pemikiran manusia, aktivitas seperti pengambilan keputusan, penyelesaian masalah, belajar . . ." (Hellman, 1978)	**"**Sebuah studi mengenai kecakapan mental melalui penggunaan model komputasi." (Charniak dan McDermott, 1985) "Studi mengenai komputasi yang membuat mungkin untuk melihat, memberikan alasan, dan bertindak." (Winston, 1992)
Bertindak Manusiawi	**Bertindak Rasional**
"Seni menciptakan yang melakukan fungsi intelegensi apabila dilakukan oleh manusia" (Kurzweil, 1990) "Studi tentang bagaimana membuat komputer melakukan sesuatu yang, saat ini, manusia melakukannya lebih baik." (Rich dan Knight, 1991)	**"***Computational Intelligence* merupakan studi untuk mendesain agen cerdas." (Poole, *et al,* 1998) "Kecerdasan buatan . . . berkaitan dengan perilaku cerdas dalam sebuah model." (Nilsson, 1998)

Gambar 2.1 Berbagai Penjelasan Kecerdasan Buatan, Dibedakan Menjadi 4 Jenis [RUS-10]

Implementasi dari kecerdasan buatan pada *game* umumnya diimplentasikan pada *non-player character*, dan salah satu dari displin yang diteliti adalah pencarian jalur terpendek [JIE-11]. Menurut Kuiper, pencarian jalur di dunia nyata, jenis algoritma apapun yang digunakan, algoritma akan terbebani oleh proses perangkat keras untuk sensor [CHE-09]. Secara relatif, dunia virtual secara keseluruhan telah dirancang dan disimpan oleh komputer yang tidak membutuhkan pendeteksian melalui sensor [CHE-09].



Gambar 2.2 Model Kecerdasan Buatan pada *Game* [MIL-09:09]

2.2 *Tactical Pathfinding*

Pencarian jalur merupakan proses penentuan sebuah set pergerakan suatu objek dari satu posisi ke posisi yang lain, tanpa bertabrakan dengan hambatan dalam jalurnya [CUI-12]. Pencarian jalur dibagi menjadi dua, statis atau dinamis, yaitu *global path searching* yang mengacu pada pencarian di dunia statis, dan *local path searching* pada lingkungan dinamis [CHE-09].

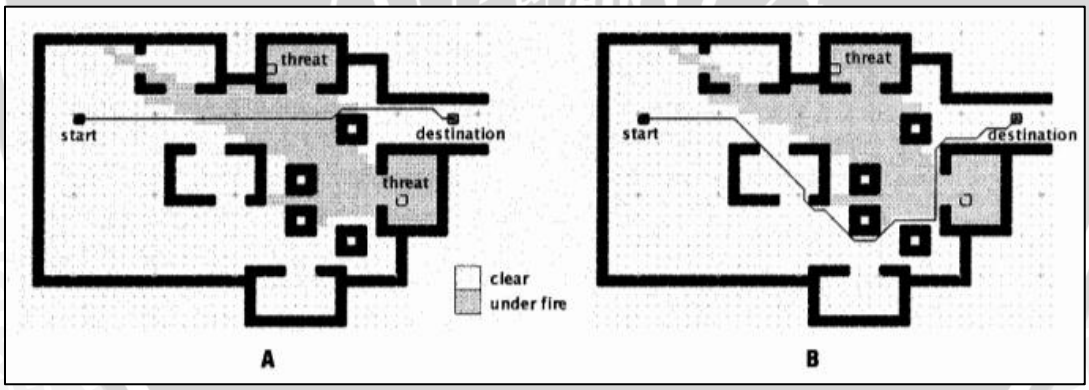
Pengerjaan praktek *level designer* pada permainan 3D pada awalnya menggunakan *waypoint* bagi *non-player character* untuk menjelajah lingkungan, dimana memakan waktu dan sering tidak akurat di beberapa kasus [DAL-09]. *Waypoint* bekerja dengan *script* yang mendefinisikan bagaimana kecerdasan buatan dapat merespon situasi yang khusus, untuk mendeskripsikan tanda saat 'berlindung' dan 'menyergap' [DAL-09]. Penempatan *waypoint* secara statis yang dikombinasikan dengan *script* secara praktis, tidak cukup untuk menyesuaikan dengan manuver taktis terhadap situasi dan medan sekaligus [DAL-09].

A* (*A-star*) kemudian muncul dan menjadi algoritma pencarian jalur terpendek yang paling banyak digunakan dengan metode *heuristic*-nya [WAN-09].

Halangan pada permainan 3D, tidak ditemui di peta berbasis 2D [NIU-08]. Situasi ini tampaknya tidak memungkinkan jika menggunakan solusi algoritma A* tradisional [NIU-08]. Solusi dari permasalahan tersebut adalah *tactical pathfinding* [DAL-09].

Tactical pathfinding tidak hanya menambahkan sentuhan realistis pada pergerakan kecerdasan buatan, tapi juga memastikan pemain yang berpengalaman tidak dapat memprediksi keputusan yang diambil oleh lawan [DAL-09]. Gambar 2.3 menunjukkan bahwa algoritma *tactical pathfinding* dapat memberikan perlindungan dan penyembunyian dibandingkan dengan algoritma pencarian jalur terpendek biasa [STR-02].

Fungsi evaluasi posisi merupakan faktor kunci untuk menjadikan sebuah pencarian jalur menjadi 'taktis' [DAL-09]. Pada tiap dan seluruh *node*, informasi statis dan dinamis mengenai *non-player character* dan dunia permainan akan diolah [DAL-09]. Informasi tersebut dimanipulasi menjadi standar bobot untuk menentukan seberapa baik *node* tersebut, untuk menuju ke titik tujuan yang dilakukan oleh pencari jalur [DAL-09].



Gambar 2.3 Perbedaan A* Tradisional (A) dengan A* *Tactical Pathfinding* (B) [STR-02]

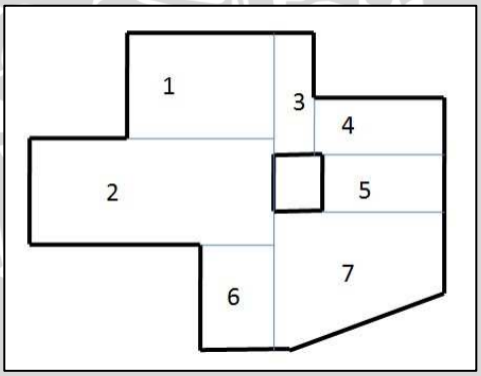
Sebagai media, sebagian besar *game* modern menggunakan *navigation mesh* sebagai media untuk pencarian jalur, disamping *Tile-based graphs*, *Dirichlet domains*, dan *waypoints* [MIL-09:246].

2.3 Peta Permainan dengan *Navigation Mesh*

Navigation Mesh (NavMesh) merupakan sebuah metode untuk mewakili dunia permainan dengan menggunakan poligon-poligon [CUI-12]. Sistem *navigation mesh* berusaha memanfaatkan seluruh keuntungan yang bisa diambil dari sistem peta *node*, tanpa harus membuat atau mempertahankan jaringan *node* [SCH-09:47].

Poligon-poligon pada *navigation mesh* harus bersifat konveks [CUI-12]. Pada sebagian besar kasus, jumlah sisi dari poligon sebuah *navigation mesh* bervariasi antara 3 sampai 6, dan pada prakteknya apabila melebihi 6 akan memberikan dampak peningkatan penggunaan memori yang signifikan [CUI-12].

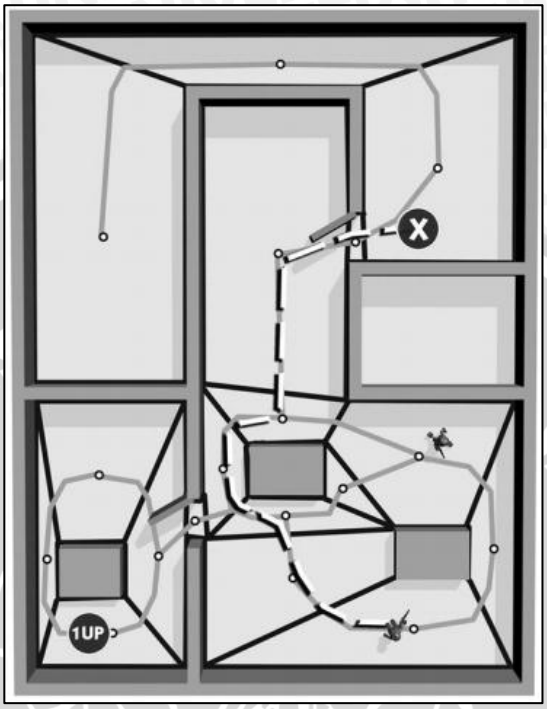
Representasi dari *navigation mesh* merupakan representasi yang mencakup seluruh permukaan dunia virtual yang dapat dilalui dengan poligon konveks, ditunjukkan dalam Gambar 2.4 [LEI-10:20]. Sistem secara algoritmis memanfaatkan poligon yang digunakan dalam pembuatan peta, untuk membentuk jaringan *node* yang digunakan oleh kecerdasan buatan, seperti dalam Gambar 2.5 [SCH-09:47].



Gambar 2.4 Representasi *Navigation Mesh* [LEI-10]

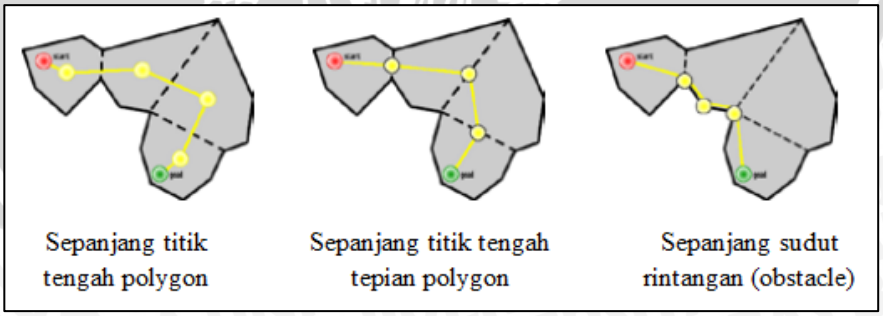
Navigation mesh juga menjadi pilihan utama sebagai representasi dunia untuk agen pada dunia virtual dalam beberapa tahun terakhir [HAL-10]. *Navigation mesh* bekerja dengan baik pada agen dengan memungkinkan algoritma pencarian jalur terpendek menentukan daerah awal dan tujuan [HAL-10].

Algoritma kemudian melakukan pencarian pada jalur konektivitas dan memilih daerah yang akan dilalui [HAL-10].



Gambar 2.5 Sistem Navigation Mesh [SCH-09]

Metode pencarian jalur pada *navigation mesh* dibedakan menjadi tiga jenis seperti dijabarkan dalam Gambar 2.6 [CUI-12]. Tidak peduli metode yang dipilih, tidak ada satupun yang menjamin bahwa itu adalah jalur yang optimal, sehingga harus dilakukan proses lebih lanjut [CUI-12].



Gambar 2.6 Tiga Cara Berbeda Untuk Menemukan Jalur [CUI-12]

2.4 Parameter Pengujian

2.4.1 *Frame per Second (FPS)*

Frame per second (FPS) pada tampilan simulasi adalah jumlah gambar fragmen, gambar-gambar berbeda ditampilkan tiap detik untuk mengelabui mata penonton menjadi objek yang bergerak [ALD-09]. FPS biasa disebut juga *frame rate*, fps merupakan sebuah metrik fluiditas [ALD-09]. Acara pada TV NTSC disiarkan pada 30 fps, sedangkan PAL adalah 25 fps, dan film pada bioskop ditampilkan pada 24 fps [ALD-09].

Kapasitas 20 fps dianggap oleh sebagian besar pihak sebagai batas minimum untuk mengelabui mata [ALD-09]. *Frame rate* yang lebih tinggi (berkisar antara 60 fps) menghasilkan animasi yang lebih memanjakan, dan beberapa konsol permainan mengorbankan elemen lain untuk menjaga level ini [ALD-09].

FPS yang lebih rendah disebabkan oleh keterbatasan sistem, pemrograman yang buruk, dan aplikasi lain yang menghabiskan sumber daya sistem (seperti kecerdasan buatan) [ALD-09]. Detail dari gambar, seperti resolusi layar, kompleksitas objek, dan pencahayaan juga menurunkan jumlah FPS [ALD-09]. FPS dapat dihitung dengan formula, jumlah *frame* yang digambar dibagi satuan detik.

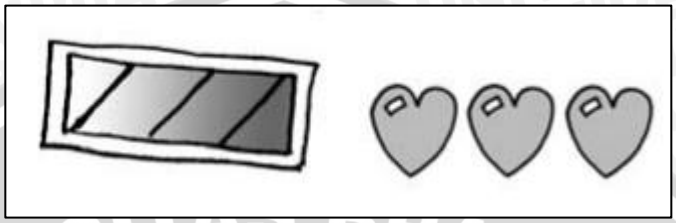
$$fps = \frac{frame\ yang\ digambar}{detik}$$

2.4.2 *Hit Points*

Hit points dari sebuah karakter menentukan seberapa banyak *damage* yang dapat ditanggung sebelum karakter tersebut tidak sadarkan diri atau mati [KIT-10]. Karakter yang menerima *damage* akan menandai jumlah *hit points* yang biasa digunakan pada *health bar* [KIT-10].

Hal ini berfungsi untuk mengindikasikan seberapa banyak *damage* yang diterima karakter [KIT-10]. *Hit points* biasa diwakili

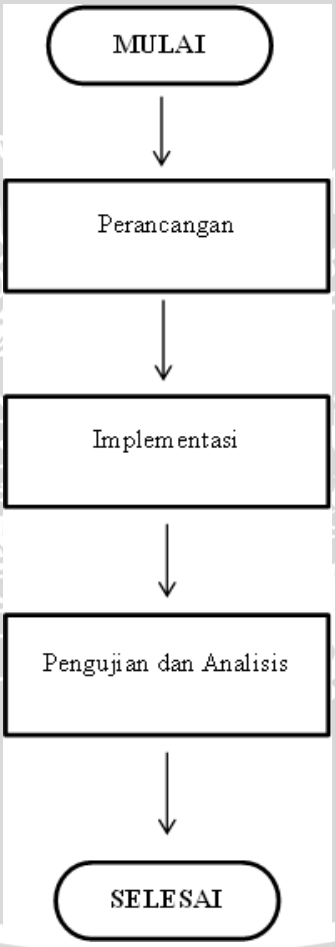
oleh *health bar*, contoh *health bar* dijabarkan dalam Gambar 2.7. Pada *health bar* saat karakter menerima *damage*, maka persentase akan berkurang atau pengosongan warna pada ikon, jika *bar* sudah kosong, karakter akan keluar dari permainan [ROG-10].



Gambar 2.7 Contoh Representasi *Health Bar* [ROG-10]

BAB III
METODOLOGI PENELITIAN

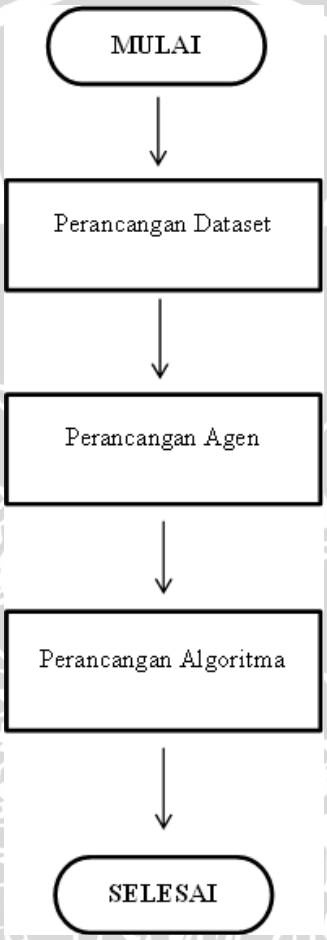
Bab ini menjelaskan langkah-langkah metodologi penelitian. Penelitian ini akan dilakukan dalam beberapa tahap, dijabarkan pada diagram alir dalam Gambar 3.1 yaitu pengumpulan dataset peta permainan, perancangan implementasi, implementasi algoritma dalam bentuk *library. Library* ditanamkan ke objek 3D dalam permainan 3D. Kesimpulan dan saran disertakan sebagai catatan atas aplikasi dan kemungkinan arah pengembangan aplikasi selanjutnya.



Gambar 3.1 Diagram Alir Runtutan Metode Penelitian
Sumber: [Metode Penelitian]

3.1 Perancangan

Bab perancangan terdiri dari tiga tahap, perancangan dataset, perancangan agen, dan perancangan algoritma. Tahap-tahap perancangan dijabarkan dalam diagram alir dalam Gambar 3.2.



Gambar 3.2 Diagram Alir Perancangan Implementasi
Sumber: [Metode Penelitian]

3.1.1 Perancangan Dataset

Dataset merupakan simulasi permainan 3D yang digunakan untuk uji coba. Dataset berupa kesatuan objek 3D agen dan halangan yang terlibat dalam simulasi dalam sebuah area permainan. Dataset dari simulasi permainan dirancang, membuat daftar kebutuhan mengenai apa saja yang dibutuhkan dalam pembuatan dataset. Sketsa

penggambaran dari dataset juga dirancang untuk mempermudah implementasi.

3.1.2 Perancangan Agen

Berupa perancangan agen yang akan menjalankan simulasi, yaitu AI Turret dan AI Agen. AI Agen merupakan agen kecerdasan buatan yang ada pada pihak pemain (*non-player character)*. AI Agen berupa objek 3D dan memiliki tugas menuju ke titik tujuan. AI Turret merupakan agen kecerdasan buatan yang berada di pihak yang bertentangan dengan pemain. AI Turret berupa objek 3D yang memiliki tugas untuk menghadang AI Agen dan memberikan bobot ancaman.

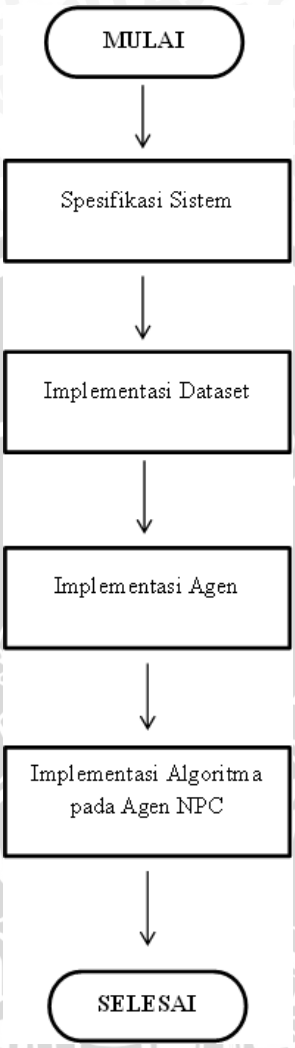
Kemampuan dari tiap-tiap agen akan didata, untuk merancang kemampuan yang dibutuhkan oleh tiap-tiap agen dalam menjalankan simulasi. Sketsa penggambaran dari AI Turret dan AI Agen juga dirancang untuk mempermudah proses implementasi.

3.1.3 Perancangan Algoritma

Perancangan algoritma yang akan dipakai dalam simulasi. Tahapan dari algoritma akan dirancang untuk memberikan rencana yang baik dalam melakukan implementasi algoritma. Metode-metode yang akan digunakan dalam implementasi, juga akan dijabarkan dan dirancang dalam subbab ini.

3.2 Implementasi

Bab implementasi terdiri dari empat tahap, yaitu spesifikasi sistem, implementasi dataset, implementasi agen yang akan menjalankan simulasi, yaitu AI Turret dan AI Agen, dan implementasi algoritma dan penanaman algoritma pada agen *non-player character*. Tahap-tahap implementasi dijabarkan dalam diagram alir dalam Gambar 3.3.



Gambar 3.3 Diagram alir Implementasi
Sumber: [Metode Penelitian]

3.2.1 Spesifikasi Sistem

Subbab ini menjabarkan spesifikasi dari lingkungan sistem tempat simulasi dijalankan. Spesifikasi sistem yang dimaksud adalah lingkungan perangkat keras dan perangkat lunak yang digunakan dalam menjalankan simulasi. Subbab ini bertujuan untuk mengetahui dalam spesifikasi tersebut, seberapa baik simulasi dijalankan untuk pengolahan data selanjutnya.

3.2.2 Implementasi Agen

Berupa implementasi dari agen yang akan menjalankan simulasi, yaitu AI Turret dan AI Agen. Kemampuan dari tiap-tiap agen akan diimplementasikan. Fungsi-fungsi yang menjalankan AI Agen dan AI Turret juga akan dijabarkan dalam subbab ini.

3.2.3 Implementasi Algoritma

Subbab yang berisi penerapan dari algoritma yang telah dirancang dan akan dipakai dalam simulasi. Metode-metode yang sudah dirancang dalam bab perancangan juga akan dijabarkan dalam subbab ini.

3.3 Pengujian dan Analisis

Pengujian yang dilakukan berdasarkan implementasi yang telah dibuat berdasarkan metriks pengujian yang telah ditentukan, yaitu FPS dan *hit points.*

3.3.1 Pengujian Skenario dalam Simulasi

Skenario pengujian algoritma dilakukan dengan beberapa alternatif, yang pertama adalah mengubah jumlah AI Turret dan peta permainan, kedua mengubah jumlah AI Agen dan peta permainan, ketiga mengubah peta permainan dengan perbandingan jumlah AI Agen dan AI Turret adalah sama.

3.3.2 Pencatatan Hasil FPS dan *Hit Points* per Simulasi

Setiap simulasi akan menghasilkan *file* XML yang mencatat FPS per detik selama simulasi berlangsung, yaitu dari titik awal AI Agen mulai bergerak, hingga simulasi berakhir yaitu saat semua AI Agen sudah mencapai titik tujuan. AI Agen juga menghasilkan *file* XML yang mencatat *hit points* tiap-tiap agen di akhir simulasi.

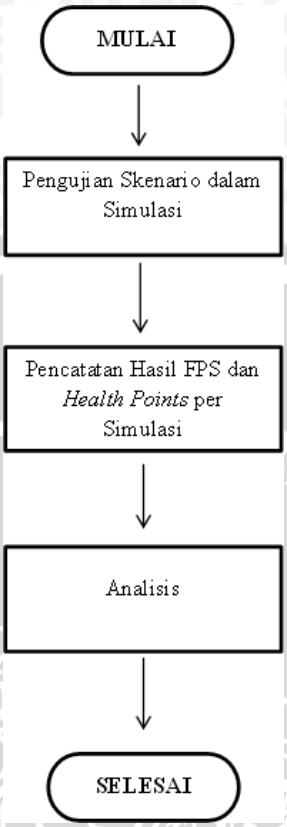
Kedua *file* XML tersebut akan dicatat dan dibandingkan antara simulasi dengan *tactical pathfinding* dengan simulasi tanpa *tactical pathfinding.*

3.3.3 Analisis

Hasil pengujian tiap uji coba per skenario akan dibedakan berdasarkan metriks pengujian FPS, dan seberapa besar rata-rata *hit points* yang dimiliki AI Agen di akhir simulasi. Hasil dari tiap-tiap skenario akan dianalisis. Analisis dibutuhkan untuk menjawab pertanyaan pada subbab rumusan masalah, yaitu menganalisis pengaruh implementasi *tactical pathfinding* terhadap rata-rata *hit points* yang diterima *non-player character* dalam permainan 3D dan pengaruh pengaruh implementasi *tactical pathfinding* terhadap rata-rata *frame per second* dalam permainan 3D.

Hasil rata-rata FPS dan *hit points* dari skenario dengan *tactical pathfinding* akan dibandingkan dengan hasil dari skenario tanpa *tactical pathfinding* dan dilakukan pengujian hipotesis dengan uji t untuk mengetahui signifikansi perbedaan diantara keduanya.

Gambar 3.4 menjabarkan agen *non-player character* dengan algoritma *tactical pathfinding* diuji pada dataset peta permainan berbasis *navigation mesh* dengan parameter pengujian yang telah ditentukan.



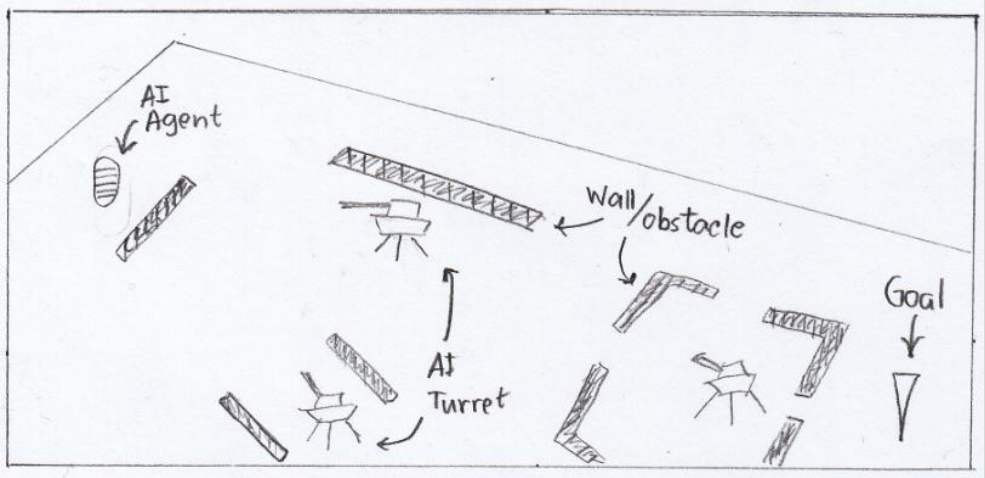
Gambar 3.4 Diagram alir Pengujian dan Analisis
Sumber: [Metode Penelitian]

BAB IV
PERANCANGAN

Bab ini membahas perancangan yang digunakan untuk implementasi *Tactical Pathfinding* untuk *Non-Player Character* dalam Permainan 3D. Perancangan ini meliputi perancangan dataset, perancangan agen dan perancangan algoritma.

4.1 Perancangan Dataset

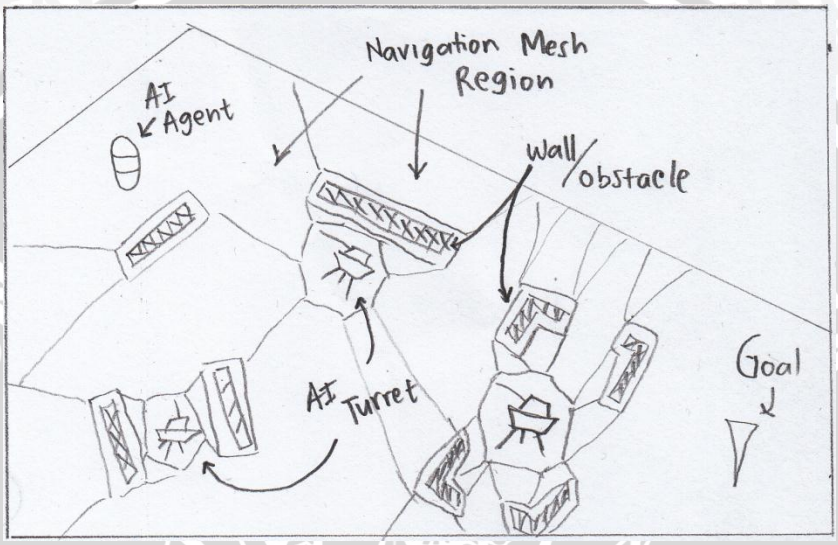
Rancangan untuk tampilan simulasi pada peta permainan ditunjukkan dalam Gambar 4.1. Simulasi pada peta permainan terdiri dari AI Agen, AI Turret, halangan, dan titik tujuan. AI Agen berjalan menuju titik tujuan melalui halangan dan rintangan ancaman dari AI Turret.



Gambar 4.1 Rancangan Dataset (*Scene*) Simulasi Permainan
Sumber: [Perancangan]

Peta terbuat dari kumpulan bidang 3D sebagai tempat berpijak agen yang disebut *plane*. *Scene* merupakan dunia dalam *game engine* tempat semua objek 3D berada, dan merupakan format dataset yang digunakan dalam melakukan simulasi. Dataset scene memuat semua agen seperti AI Agen, AI Turret, halangan-halangan berupa objek 3D berbentuk dinding dan *plane*.

Navigation mesh dari peta akan dihasilkan oleh *game engine*. *Plane* yang memuat segala halangan berupa dinding maupun AI Turret akan menghasilkan daerah *navigation mesh.* Rancangan dari peta permainan dengan *navigation mesh* digambarkan dalam Gambar 4.2. Halangan yang digunakan dalam dataset adalah halangan statis, halangan berupa *plane,* AI Turret, dan dinding penghalang.

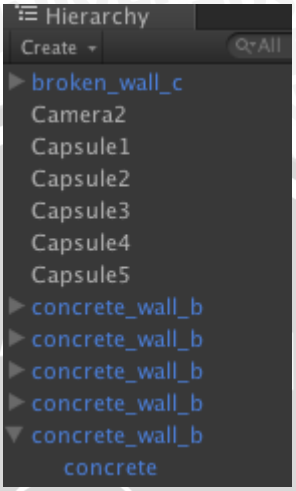


Gambar 4.2 Rancangan Dataset Simulasi Permainan dengan *Navigation Mesh*
Sumber: [Perancangan]

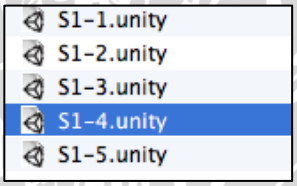
Plane merupakan area simulasi, maka halangan yang bisa dibentuk adalah bentuk objek *plane* yang diatur sehingga membentuk halangan alam berupa lekuk medan dan ketinggian. AI Turret sebagai halangan yang dapat menyerang AI Agen, merupakan objek 3D statis, karena AI Turret tidak dapat berpindah posisi dan hanya moncong senjata yang dapat berputar pada poros objek AI Turret. Dinding penghalang merupakan penghalang statis berupa objek 3D berbentuk dinding. Semua penghalang statis akan menghasilkan perubahan bentuk pada daerah *navigation mesh* yang dihasilkan diatas *plane.*

Isi dari format dataset berupa *scene* dijabarkan dalam Gambar 4.3, daftar tersebut merupakan isi dari *scene* yang berupa objek-objek 3D yang

digunakan dalam *scene* dalam *game engine* Unity. Format dari dataset adalah .unity dijabarkan dalam Gambar 4.4.



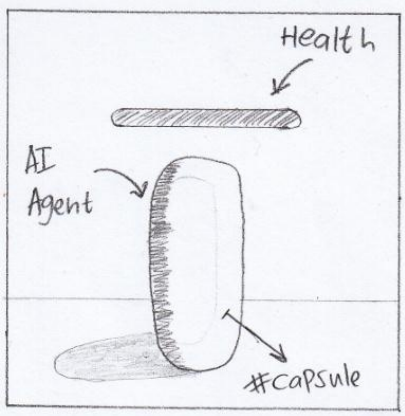
Gambar 4.3 Isi dari Dataset Permainan dalam Daftar *Hierarchy*
Sumber: [Perancangan]

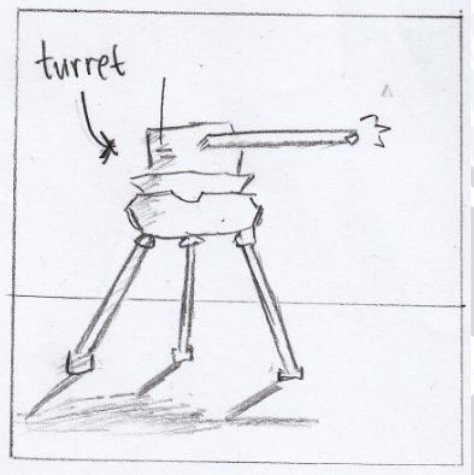


Gambar 4.4 Contoh Tampilan Format dari Dataset
Sumber: [Perancangan]

4.2 Perancangan Agen

Tabel 4.1 Perancangan Agen

No.	Nama	Rancangan Tampilan	Kemampuan
1.	AI Agen		1. Berjalan ke semua arah 360°, namun tidak dapat melompat 2. Percepatan bernilai nol, dan kecepatan diatur oleh *user* 3. Memiliki *health*

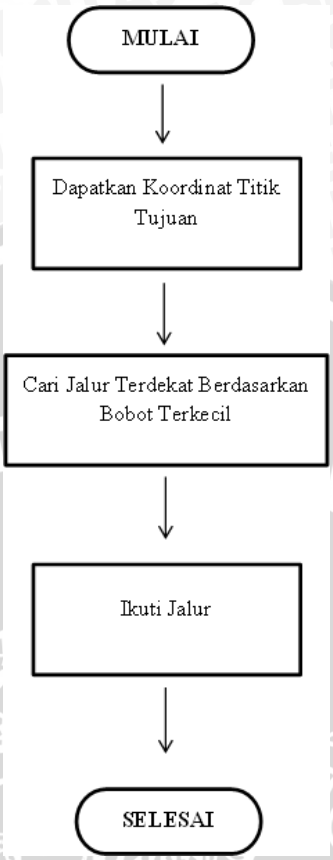
			bar yang dicatat di akhir simulasi.
2.	AI Turret		1. Moncong senjata dapat berputar 360° mengikuti posisi AI Agen 2. Menembakkan peluru dengan rasio yang ditentukan. 3. Menembak AI Agen terdekat dan mengubah target apabila AI Agen tersebut sudah diluar jangkauan

Sumber: [Perancangan]

4.3 Perancangan Algoritma

Algoritma *tactical pathfinding* merupakan algoritma pencarian jarak terpendek A* ditambah dengan pertimbangan perhitungan bobot. Bobot yang dimaksud adalah bobot ancaman yang mengubah pencarian jalur terpendek tradisional menjadi pencarian jalur secara taktis. Algoritma *tactical pathfinding* bertujuan untuk mengubah algoritma pencarian jalur terpendek A*, menjadi algoritma pencarian jalur taktis dengan perhitungan bobot.

Algoritma *tactical pathfinding* memiliki tiga tahapan, mendapatkan koordinat titik tujuan, mencari jalur terdekat berdasarkan bobot terkecil dan membuat jalurnya, dan mengikuti jalur yang sudah tercipta. Perancangan algoritma *tactical pathfinding* dijabarkan pada diagram alir dalam Gambar 4.5.



Gambar 4.5 Diagram Alir *Tactical Pathfinding*
Sumber: [Perancangan]

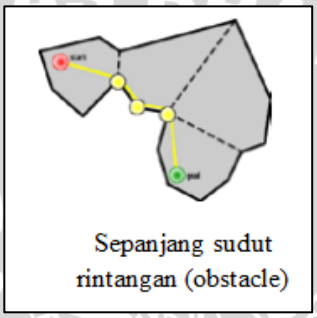
4.3.1 Mendapatkan Koordinat Titik Tujuan

Algoritma *tactical pathfinding* diawali dengan menentukan letak titik awal dan titik tujuan yang berada pada dunia peta permainan. Secara relatif, dunia virtual secara keseluruhan telah dirancang dan disimpan oleh komputer yang tidak membutuhkan pendeteksian melalui sensor [CHE-09]. *Node-node* dan koordinat titik awal dan titik tujuan sudah dihasilkan saat awal simulasi.

Pada peta permainan *navigation mesh*, poligon yang terbentuk ditransformasikan menjadi *node*. Tepi yang digunakan untuk berbagi bersama antar dua poligon akan dipetakan menjadi penghubung antar dua *node*. Metode yang dipakai dalam pencarian jalur pada *navigation*

mesh adalah metode pembuatan jalur atau *node* sepanjang sudut rintangan, yang dijabarkan dalam Gambar 4.7.

Dalam *game engine* Unity, representasi *node* adalah daerah *navigation mesh*, diawali dan diakhiri oleh *corner*, sesuai metode *node* sepanjang sudut rintangan. Untuk koordinat x,y,z titik awal dan titik tujuan didapatkan dari proses pemrograman untuk mengambil koordinat dari objek 3D yang menjadi acuan titik awal, yaitu AI Agen, dan objek yang menjadi acuan titik tujuan, yaitu *Goal*.



Gambar 4.6 Metode Menemukan Jalur Sepanjang Sudut Rintangan [CUI-12]

4.3.2 Mencari Jalur Terdekat Berdasarkan Bobot Terkecil

Titik awal dan titik tujuan dihubungkan oleh *node-node*, bermula dari titik awal, *node-node* tetangga dari titik awal akan dibandingkan bobotnya dan diambil yang paling kecil. Dalam pencarian jarak berdasarkan A*, *g(n)* melambangkan nilai pasti dari titik awal ke tiap titik *n*, *h(n)* melambangkan nilai perkiraan dari titik *n* menuju titik akhir [CUI-11].

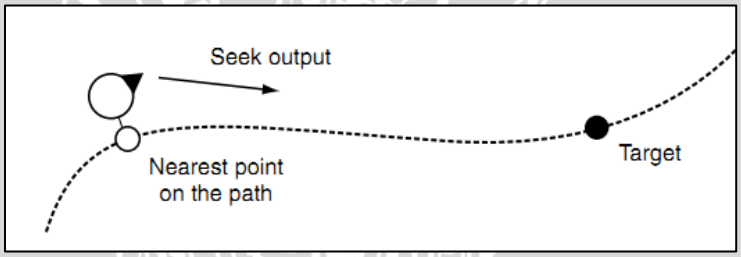
Bobot taktis tiap titik *n* terhadap ancaman dilambangkan *t(n)*, dan *f(n)=g(n)+h(n)+t(n)*, nilai *f* diurutkan dari yang terkecil sebagai pembuatan jalur taktis terdekat. *Node* diawali dan diakhiri oleh pojok dari titik pertemuan antar daerah pada *navigation mesh* yang disebut *corner*. Perulangan proses membandingkan akan dilakukan hingga tidak tersisa, atau mencapai titik tujuan dan membentuk sebuah hubungan *node-node*.

4.3.3 Mengikuti Jalur

Jalur berupa kumpulan hubungan *node-node* yang sudah tercipta dari pencarian berdasarkan bobot terkecil, akan diikuti oleh AI Agen untuk menuju ke titik tujuan dari titik awal. Pencarian jalur merupakan sebuah perilaku yang didelegasikan [MIL-09:76]. Pencarian jalur akan menghitung posisi target berdasarkan lokasi karakter secara aktual dan bentuk dari jalur tersebut [MIL-09].

Berdasarkan penghitungan tersebut, kemudian AI Agen akan mencari tiap-tiap ujung *node*, yang disebut *corner*, yang sudah terbentuk. Proses pencarian *corner* oleh AI Agen disebut *seek*, dimana proses *seek* berusaha mengarahkan agen menuju titik tujuan [REY-13]. Titik tujuan yang dicapai bukan merupakan titik tujuan akhir, melainkan *corner* atau ujung dari *node*.

AI Agen akan berjalan menuju ke arah *corner* dan akan merubah tujuan ke *corner* berikutnya hingga mencapai titik tujuan akhir. Agen yang mengikuti jalur dijelaskan seperti dalam Gambar 4.7.



Gambar 4.7 Agen Mencari Target Tujuan dalam Jalur yang Telah Diperhitungkan [MIL-09]

BAB V
IMPLEMENTASI

Bab ini membahas mengenai implementasi algoritma berdasarkan hasil yang telah didapatkan dari analisis kebutuhan dan proses perancangan. Pembahasan terdiri dari penjelasan tentang spesifikasi sistem, batasan-batasan dalam implementasi, implementasi rancangan algoritma.

5.1 Spesifikasi Sistem

Hasil analisis kebutuhan dan perancangan perangkat lunak yang telah dijelaskan pada bab 4 menjadi acuan untuk melakukan implementasi algoritma *tactical pathfinding* untuk *non-player character* dalam permainan 3D yang dapat berfungsi sesuai dengan kebutuhan. Spesifikasi sistem untuk implementasi algoritma dijelaskan pada spesifikasi perangkat keras dan perangkat lunak.

5.1.1 Spesifikasi Perangkat Keras

Spesifikasi perangkat keras yang dipakai dalam proses pengembangan dijelaskan dalam Tabel 5.1.

Tabel 5.1 Spesifikasi Perangkat Keras Komputer

Nama Komponen	Spesifikasi
Prosesor	2.4 GHz Intel Core 2 Duo
Memori	4 GB DDR3
Hard Drive	250 GB
Kartu Grafis	NVIDIA GeForce 320M 256 MB
Motherboard	Apple Inc. Mac-F222BEC8

Sumber: [Implementasi]

5.1.2 Spesifikasi Perangkat Lunak

Spesifikasi perangkat lunak yang dipakai dalam proses pengembangan dijelaskan dalam Tabel 5.2.

Tabel 5.2 Spesifikasi Perangkat Lunak Komputer

Sistem Operasi	Macintosh OSX Lion 10.7.2
Game Engine	Unity 3D 3.5.0 for Mac

Sumber: [Implementasi]

5.2 Implementasi Dataset

Implementasi dari peta dibuat beberapa skenario untuk membuktikan sistem berjalan dengan baik, serta sebagai penguji performa algoritma di berbagai jenis peta permainan. Peta permainan adalah peta 3D berbasis *navigation mesh* statis karena Unity 3.5 sebagai *game engine* membatasi *user* dalam mengakses *navigation mesh* yang telah dihasilkan.

Tiap simulasi akan dihasilkan satu *file* XML yang berisi *frame per second* tiap detik. *File* ini dipergunakan untuk pengujian performa dari sebuah simulasi, contoh keluaran berupa *file* XML ditunjukkan dalam Gambar 5.1. Gambar 5.1 menjabarkan keluaran *file* XML yang mencatat FPS selama simulasi berlangsung, baris 3 hingga 11 adalah nilai dari FPS per detik, karena terdiri dari 8 baris, maka simulasi tersebut berlangsung selama 8 detik.

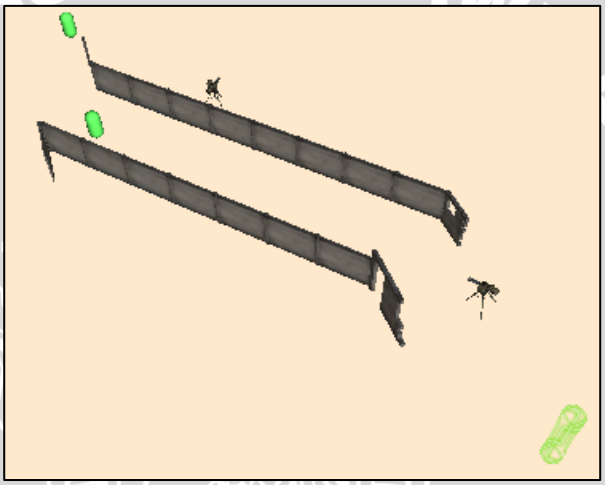
Tampilan simulasi pada peta permainan ditunjukkan dalam Gambar 5.2. Dalam Gambar 5.2. objek 3D berbentuk kapsul hijau adalah AI Agen, objek 3D berbentuk tembok adalah halangan statis, objek 3D berbentuk senapan adalah AI Turret, dan objek 3D berbentuk tabung hijau semi-transparan adalah titik tujuan dari AI Agen. Tampilan simulasi dengan *navigation mesh* dalam Gambar 5.3, warna-warna yang berbeda dan garis-garis pada objek 3D *plane* tempat berpijak, merupakan daerah-daerah poligon *navigation mesh* yang dihasilkan secara otomatis oleh *game engine*. Daerah-daerah tersebut berfungsi sebagai *node* AI Agen dalam melakukan pencarian jalur.

Simulasi pada peta permainan terdiri dari AI Agen, AI Turret, halangan, dan titik tujuan. AI Agen berjalan menuju titik tujuan melalui halangan dan rintangan ancaman dari AI Turret.

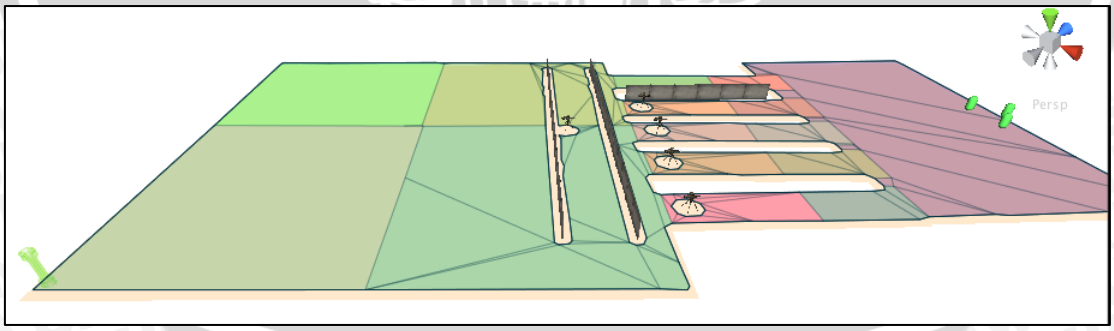
myFPS.xml

```
<?xml version="1.0" encoding="utf-8"?>
<fps>
  <frame detik="82.98 FPS" />
  <frame detik="118.70 FPS" />
  <frame detik="124.44 FPS" />
  <frame detik="116.72 FPS" />
  <frame detik="111.91 FPS" />
  <frame detik="123.14 FPS" />
  <frame detik="123.48 FPS" />
  <frame detik="123.80 FPS" />
  <frame detik="123.09 FPS" />
</fps>
```

Gambar 5.1 myFPS.xml sebagai Keluaran Pencatatan FPS dalam Simulasi
Sumber: [Implementasi]



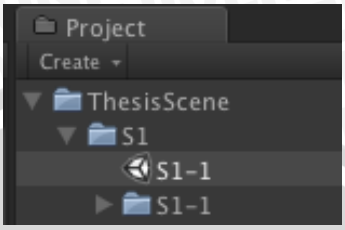
Gambar 5.2 Dataset Simulasi Permainan
Sumber: [Implementasi]



Gambar 5.3 Dataset Simulasi Permainan dalam *Navigation Mesh*
Sumber: [Implementasi]

Tiap simulasi akan disimpan dalam format *scene*, ekstensi .unity sebagai *unity document*, *file* ini hanya dapat dibuka oleh *game engine* Unity, tampilan *file* scene pada *game engine* tampak dalam Gambar 5.4.

Scene berisi semua objek dari permainan, dalam *scene*, pengembang *game* dapat menempatkan *environment*, halangan, dekorasi dan objek-objek lain untuk membuat *game*.



Gambar 5.4 *File scene* pada *Game Engine*
Sumber: [Implementasi]

5.3 Implementasi Agen

5.3.1 Implementasi AI Agen

Tampilan AI Agen ditunjukkan dalam Gambar 5.5. AI Agen berupa objek 3D berbentuk kapsul, dengan *health bar* diatasnya yang berguna untuk memonitor *hit points* tiap agen. Agen dapat berjalan melewati peta dengan kecepatan yang dapat diatur, namun dengan kecepatan konstan, dan agen tidak dapat melompat. Pencarian jalur dilakukan oleh *library* NavMeshAgent yang disediakan oleh *game engine*, dan kode untuk mencari koordinat titik tujuan dijabarkan dalam Gambar 5.6. Komponen dari tujuan yang dibutuhkan NavMeshAgent yang berisi koordinat titik tujuan diambil dari koordinat `m_player.position`.

Gambar 5.5 Tampilan AI Agen
Sumber: [Implementasi]

```
GetComponent<NavMeshAgent>().destination = m_Player.position;
```

Gambar 5.6 Kode Mendapatkan Posisi Titik Tujuan
Sumber: [Implementasi]

AI Agen hanya memiliki tugas untuk berjalan menuju titik tujuan, dengan algoritma yang sudah tertanam, agen dengan sendirinya akan menghindari halangan. *File* XML yang berisi *hit points* akhir saat mencapai titik tujuan tiap-tiap agen akan dihasilkan di akhir simulasi, contoh dari *file* tersebut dijelaskan dalam Gambar 5.7.

myHealth.xml

```
<?xml version="1.0" encoding="utf-8"?>
<health>
  <final health="100" />
  <final health="100" />
  <final health="100" />
  <final health="99" />
  <final health="99" />
</health>
```

Gambar 5.7 myHealth.xml sebagai Keluaran Pencatatan *Hit Points* dalam Simulasi
Sumber: [Implementasi]

5.3.2 Implementasi AI Turret

Tampilan AI Turret ditunjukkan dalam Gambar 5.8. AI Turret merupakan objek 3D statis, dengan satu moncong senjata yang bisa berputar 360 derajat, bertujuan untuk mempermudah membidik, yaitu ke arah AI Agen, pemrograman dijabarkan dalam Gambar 5.8.

Gambar 5.8 Tampilan AI Turret
Sumber: [Implementasi]

```
1:var targetPoint = target.position;
2:var    targetRotation    =    Quaternion.LookRotation    (targetPoint    -
3:transform.position, Vector3.up);
4:transform.rotation = Quaternion.Slerp(transform.rotation, targetRotation,
5:Time.deltaTime * 2.5);
```

Gambar 5.9 Pemrograman untuk Membidik Target
Sumber: [Implementasi]

Penjelasan dari fungsi membidik target dalam Gambar 5.9 :

i. Baris 1 mendeklarasikan bahwa target adalah koordinat dari target. Target secara otomatis ditentukan oleh fungsi pencarian target, koordinat terikat pada target kemanapun target pergi.

ii. Baris 2 dan 3 menjabarkan bahwa moncong AI Turret akan berputar melihat ke posisi target (`targetPoint`) terlebih dahulu dengan fungsi `Quaternion.LookRotation`

iii. Baris 4 dan 5 menjabarkan bagaimana kerja berputar dari titik awal moncong senjata ke arah titik tujuan dengan kecepatan deltaTime * 2,5.

AI Turret hanya memiliki satu tugas, yaitu membidik dan menembak agen yang terdekat yang ditentukan secara otomatis dan mengubah target apabila sudah di luar jangkauan. AI Turret merupakan objek statis, sehingga dalam peta permainan 3D berbasis *navigation mesh* akan dideteksi sebagai halangan. Kelas yang ditanamkan pada AI Turret dijabarkan dalam Tabel 5.3 dan 5.4.

Fungsi *TacticalRay* dikeluarkan oleh AI Turret untuk mengeluarkan *ray* yang berfungsi untuk memberikan bobot pada *plane* peta *navigation mesh.* Fungsi TacticalRay dijabarkan dalam Gambar 5.10.

Fungsi TacticalRay

```
1:  Ray frontray = transform position Vector3 (x,y-0.03,z)
2:  RaycastHit () hit
3:
4:   if Physic.Raycast is hit
5:    SetLayerCost to NavMesh = 10
```

Gambar 5.10 Fungsi TacticalRay
Sumber: [Implementasi]

Penjelasan dari fungsi TacticalPathfinding dalam Gambar 5.10 :

1. Baris 1 adalah deklarasi fungsi *ray* dengan nama *frontray*, untuk mengeluarkan sebuah garis dari asal ke arah tertentu. *Frontray* dikeluarkan ke arah berdasarkan Vector3 : `x,y-0.03,z`
2. Baris 2 adalah deklarasi struct *Raycast* bernama hit, berfungsi untuk mendapatkan informasi yang didapat dari *ray* yang dikeluarkan.
3. Baris 4 dan 5 adalah kondisi apabila *raycast* berpapasan dengan *collider* pada *mesh* peta maka melakukan SetLayerCost, memberikan bobot pada layer sebesar 10.

Tabel 5.3 Kelas MachineGun

MachineGun Class

`void Fire ()`
Tujuan: rasio waktu dari peluru ditembakkan, dan memanggil fungsi `FireOneShot()`
 	
`void FireOneShot()`
Tujuan:menembakkan satu peluru, dan memberikan dorongan pada objek target, dan memanggil fungsi `Reload()` apabila peluru dalam satu *clip* telah habis
 	
`void Reload ()`
Tujuan: memberi jeda waktu untuk mengisi ulang, mengurangi jumlah *clip* dan mengisi ulang peluru untuk jatah menembak
 	
`void GetBulletsLeft ()`
Tujuan: mengeset nilai peluru yang tersisa

Sumber: [Implementasi]

Tabel 5.4 Kelas SentryGun

SentryGun Class

`void Start ()`
Tujuan: mengeset nilai awal *targetting* adalah *null*, supaya pada awal, tidak langsung menentukan target
 	
`void Update()`
Tujuan: mencari dan membidik target dengan *tag* `"Player"` apabila target berada dalam jangkauan, yaitu apabila fungsi `CanSeeTarget()` bernilai `true`
 	
`void CanSeeTarget ()`
Tujuan: mengembalikan nilai *boolean* `true` atau `false` dalam menentukan target, `true` apabila target terkena *raycast* dan dalam jangkauan, `false` apabila target berada di luar jangkauan tembak

Sumber: [Implementasi]

5.4 Implementasi Rancangan Algoritma

Algoritma *Tactical Pathfinding* digunakan untuk mencari jalur terpendek berdasarkan pertimbangan bobot ancaman yang dikeluarkan oleh AI Turret. Implementasi *pseudocode* dari algoritma dijabarkan dalam Tabel 5.5.

Tabel 5.5 *Pseudocode* Algoritma *Tactical Pathfinding*

Pseudocode *TacticalPathfinding*
Deklarasi currentnode, g(jarak dari startpoint ke tiap node), h(jarak node ke goalpoint),t(bobot tactical), f(g+h+t), openlist, closedlist, distance, path **Deskripsi** Input: koordinat start point Proses: 1. Menentukan koordinat goal point 2. Memasukkan startpoint sebagai currentnode dan mengecek tiap open list

```
3. Membandingkan nilai f tiap node pada openlist,
   lalu dipilih yang terkecil
Output: Jalur terdekat dengan pertimbangan bobot
```

Sumber: [Implementasi]

Node-node dan koordinat titik awal dan titik tujuan sudah dihasilkan saat awal simulasi. Dunia virtual secara keseluruhan telah dirancang dan disimpan oleh komputer yang tidak membutuhkan pendeteksian melalui sensor [CHE-09].

Pendeklarasian *node* awal, dan *node* akhir yaitu titik awal dan titik tujuan. Pembuatan dua buah *list*, closedList untuk yang sudah dievaluasi, nilai awal adalah *null* dan openList untuk *node* yang akan dievaluasi, nilai awal adalah *node* start. Pendeklarasian *list* parent, yaitu *list* dari *node* yang sudah dilewati.

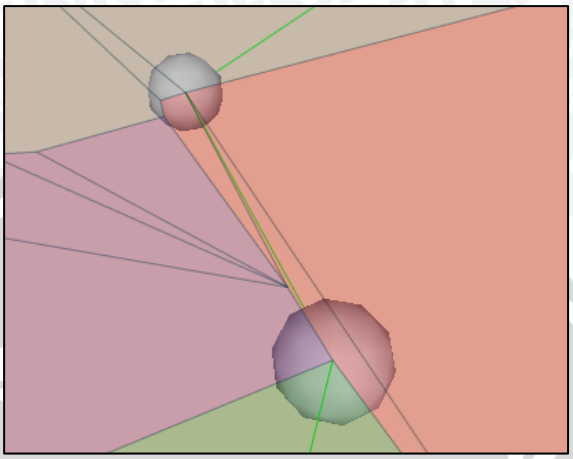
Dalam pencarian jarak berdasarkan A*, *g(n)* melambangkan nilai pasti dari titik awal ke tiap titik *n*, *h(n)* melambangkan nilai perkiraan dari titik *n* menuju titik akhir [CUI-11]. Bobot taktis tiap titik *n* terhadap ancaman dilambangkan *t(n)*, dan *f(n)=g(n)+h(n)+t(n)*, nilai *f* diurutkan dari yang terkecil sebagai pembuatan jalur taktis terdekat.

Proses pencarian titik goal akan berhenti apabila openList sudah kosong, atau sudah mencapai titik goal. Langkah pertama adalah menentukan *node* yang kini dievaluasi, *current*, dipilih berdasarkan *node* dengan nilai *f_score* paling kecil pada openList.

Jika *current* adalah goal, maka pencarian berakhir dengan perunutan jalur, sebaliknya jika ternyata belum mencapai goal, memindahkan *node current* dari openList ke closedList. Selanjutnya, dilakukan perulangan pengecekan *node* tetangga dari current *node* dengan dipilih yang memiliki nilai *f* terkecil hingga openList kosong atau mencapai titik akhir. Jalur yang terdiri dari *node-node* tersebut digunakan oleh AI Agen untuk menuju ke titik tujuan secara taktis.

Implementasi dari *node* dan *corner* dari daerah *navigation mesh* yang terbentuk berdasarkan metode pembuatan jalur sepanjang sudut rintangan terdapat dalam Gambar 5.11. Dalam Gambar 5.11 tampak garis

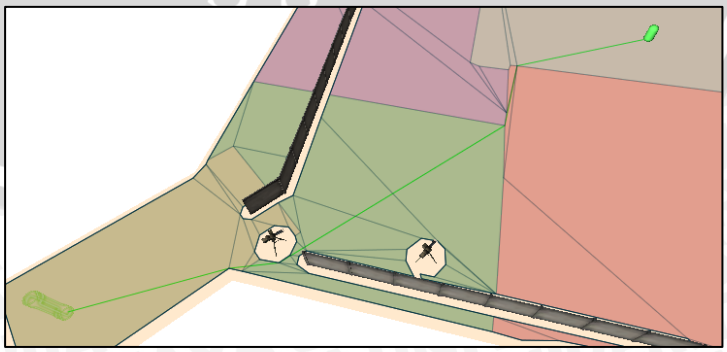
hijau adalah *node*, diawali dan diakhiri oleh *corner* yang diwakili dengan bangun bola semi-transparan 3D berwarna biru



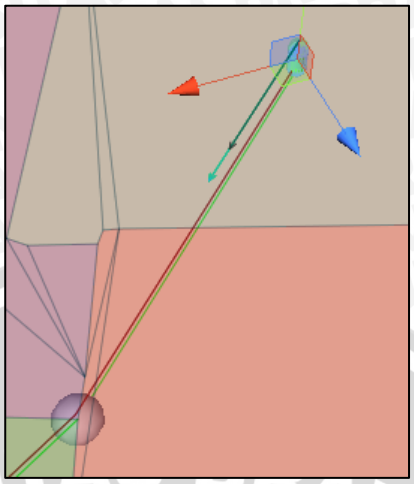
Gambar 5.11 Representasi *Node* dan *Corner* dari Daerah *Navigation Mesh*
Sumber: [Implementasi]

Kumpulan dari *node-node* yang sudah terbentuk berdasarkan perhitungan dari titik awal ke titik tujuan akan membentuk sebuah jalur seperti ditunjukkan dalam Gambar 5.12. Dalam Gambar 5.12 *node* berawal dari koordinat dimana objek 3D AI Agen berada dan berakhir di koordinat objek 3D titik tujuan.

Setelah jalur terbentuk, maka AI Agen akan mengikuti jalur, dengan mencari dan mendatangi (*seek)* satu persatu ujung *node* atau *corner* dari jalur hingga mencapai titik tujuan akhir. Representasi dari AI Agen yang berjalan mengikuti jalur ditunjukkan dalam Gambar 5.13. Dalam Gambar 5.13 arah AI Agen ditunjukkan pada panah



Gambar 5.12 *Node-Node* yang Terhubung dari Titik Awal ke Titik Tujuan
Sumber: [Implementasi]



Gambar 5.13 AI Agen Mencari dan Mendatangi Tiap Ujung *Node* (*Corner*)
Sumber: [Implementasi]

BAB VI
PENGUJIAN DAN ANALISIS

Bab ini dilakukan proses pengujian dan analisis terhadap implementasi *Tactical Pathfinding* untuk *Non-Player Character* dalam Permainan 3D. Pengujian dilakukan dengan menguji tiap skenario dan pencatatan hasil berdasarkan parameter pengujian. Parameter pengujian performa berupa jumlah rata-rata *frame per second* pada *game engine* dan rata-rata *hit points non-player character* di akhir simulasi.

6.1 Uji Coba dan Hasil Kinerja

Pengujian dilakukan untuk mengetahui performa dari sistem dalam menjalankan algoritma *tactical pathfinding.* Performa dari sistem dapat ditinjau dari rata-rata *frame per second* per simulasi. Jumlah *hit points* yang diterima oleh agen *non-player character* digunakan sebagai sebagai bukti bahwa agen dapat menuju jalur terpendek secara taktis.

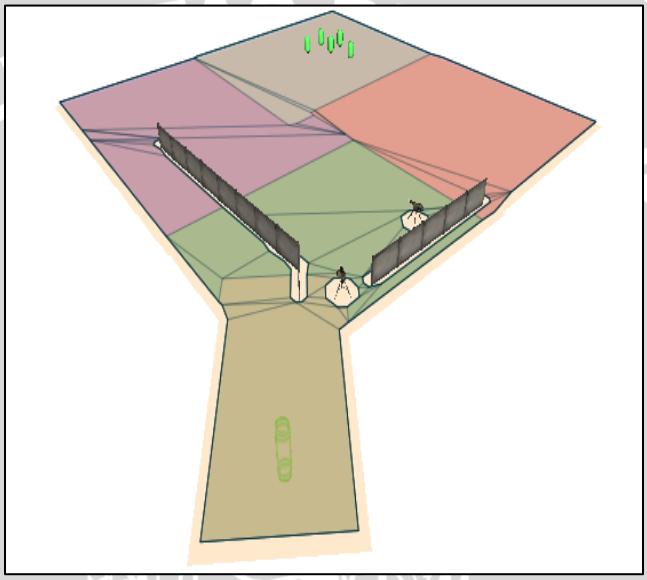
Uji coba dibagi menjadi 3 skenario:

1. Uji coba skenario 1, simulasi dilakukan tanpa merubah jumlah AI Agen yaitu 5, dengan merubah jenis peta dan jumlah AI Turret.
2. Uji coba skenario 2, simulasi dilakukan tanpa merubah jumlah AI Turret yaitu 5, dengan merubah jenis peta dan jumlah AI Agen.
3. Uji coba skenario 3, simulasi dilakukan dengan perbandingan jumlah AI Agen dan AI Turret yang sama pada peta permainan yang berbeda.

6.1.1 Skenario 1

Uji coba skenario 1 adalah simulasi tanpa mengubah jumlah AI Agen yaitu 5 agen untuk menuju ke titik tujuan dengan merubah peta permainan dan jumlah AI Turret yang menghalangi AI Agen. Simulasi skenario 1 dimaksudkan untuk menguji fungsionalitas AI Agen dalam melewati rintangan AI Turret. Contoh simulasi skenario 1 dijabarkan dalam Gambar 6.1.

Hasil uji coba FPS skenario 1 terdapat dalam Tabel 6.1, dijelaskan dalam grafik dalam Gambar 6.2. Dalam Tabel 6.1 rata-rata dari FPS simulasi permainan 3D dengan *tactical pathfinding* adalah 126,91 dan rata-rata FPS simulasi permainan 3D tanpa *tactical pathfinding* adalah 133,46. Dalam Gambar 6.2, tampak dalam grafik bahwa 2 dari 7 uji coba FPS dengan *tactical pathfinding* lebih rendah dan berbanding lurus dengan FPS tanpa *tactical pathfinding.*

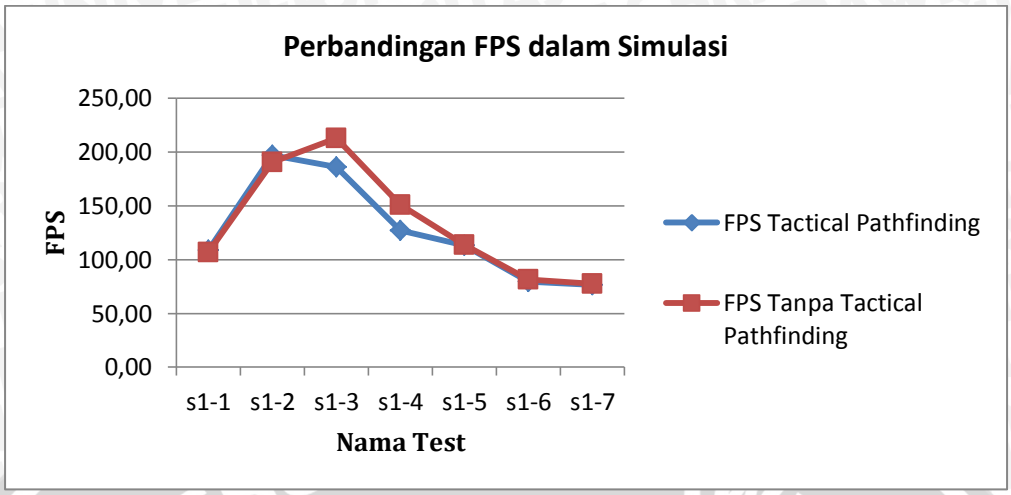


Gambar 6.1 Contoh Skenario 1
Sumber : [Pengujian dan Analisis]

Tabel 6.1 Tabel Hasil Pengujian FPS Skenario 1

No. Test	Jumlah AI Agen	Jumlah AI Turret	Tactical Pathfinding (FPS)	Tanpa Tactical Pathfinding (FPS)
1	5	2	108,64	106,61
2	5	3	197,14	190,46
3	5	4	186,17	213,04
4	5	5	127,03	151,02
5	5	6	113,33	113,97
6	5	20	79,30	81,52
7	5	40	76,75	77,58
AVG			**126,91**	**133,46**

Sumber : [Pengujian dan Analisis]



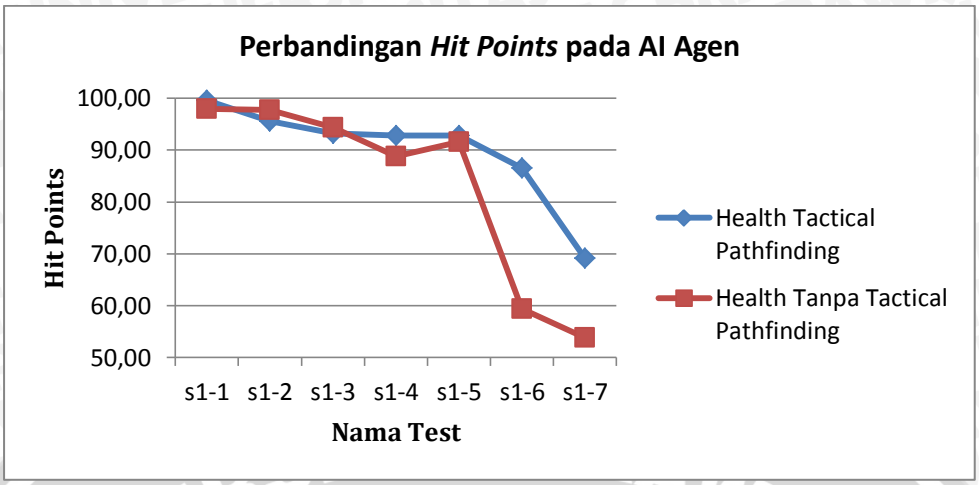
Gambar 6.2 Grafik Hasil Pengujian FPS Skenario 1
Sumber : [Pengujian dan Analisis]

Hasil uji coba *hit points* skenario 1 terdapat dalam Tabel 6.2, dijelaskan dalam grafik dalam Gambar 6.3. Dalam Tabel 6.2 rata-rata *hit points* dengan *tactical pathfinding* adalah 89,97 dan rata-rata FPS simulasi permainan 3D tanpa *tactical pathfinding* adalah 83,40. Dalam Gambar 6.2, tampak dalam grafik bahwa 2 dari 7 uji coba *hit points* dengan *tactical pathfinding* lebih tinggi dan berbanding lurus dengan *hit points* tanpa *tactical pathfinding.*

Tabel 6.2 Tabel Hasil Pengujian *Hit Points* Skenario 1

No. Test	Jumlah AI Agen	Jumlah AI Turret	Tactical Pathfinding (Poin)	Tanpa Tactical Pathfinding (Poin)
1	5	2	99,60	98,00
2	5	3	95,60	97,80
3	5	4	93,20	94,40
4	5	5	92,80	88,80
5	5	6	92,80	91,60
6	5	20	86,6	59,4
7	5	40	69,2	53,8
AVG			**89,97**	**83,40**

Sumber : [Pengujian dan Analisis]

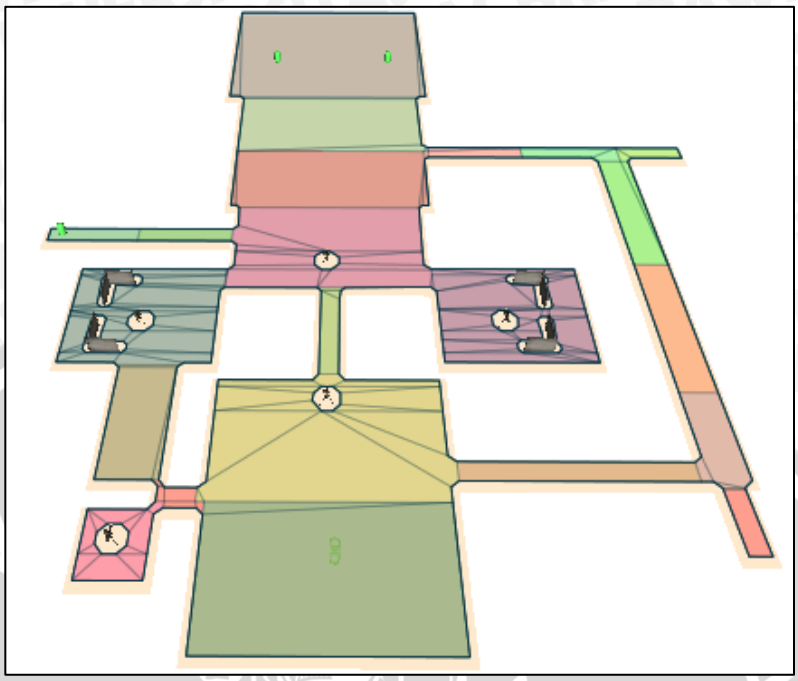


Gambar 6.3 Grafik Hasil Pengujian *Hit Points* Skenario 1
Sumber : [Pengujian dan Analisis]

6.1.2 Skenario 2

Uji coba skenario 2 adalah simulasi tanpa mengubah jumlah AI Turret yaitu 5 agen untuk menghalangi AI Agen dengan merubah peta permainan dan jumlah AI Agen yang akan menuju ke titik tujuan. Simulasi skenario 2 dimaksudkan untuk menguji fungsionalitas AI Turret dalam menghadang AI Agen. Contoh simulasi skenario 2 dijabarkan dalam Gambar 6.4.

Hasil uji coba FPS skenario 2 terdapat dalam Tabel 6.3, dijelaskan dalam grafik dalam Gambar 6.5. Dalam Tabel 6.3 rata-rata dari FPS simulasi permainan 3D dengan *tactical pathfinding* adalah 101,64 dan rata-rata FPS simulasi permainan 3D tanpa *tactical pathfinding* adalah 113,94. Dalam Gambar 6.5, tampak dalam grafik bahwa 1 dari 7 uji coba FPS dengan *tactical pathfinding* lebih rendah daripada FPS tanpa *tactical pathfinding.*

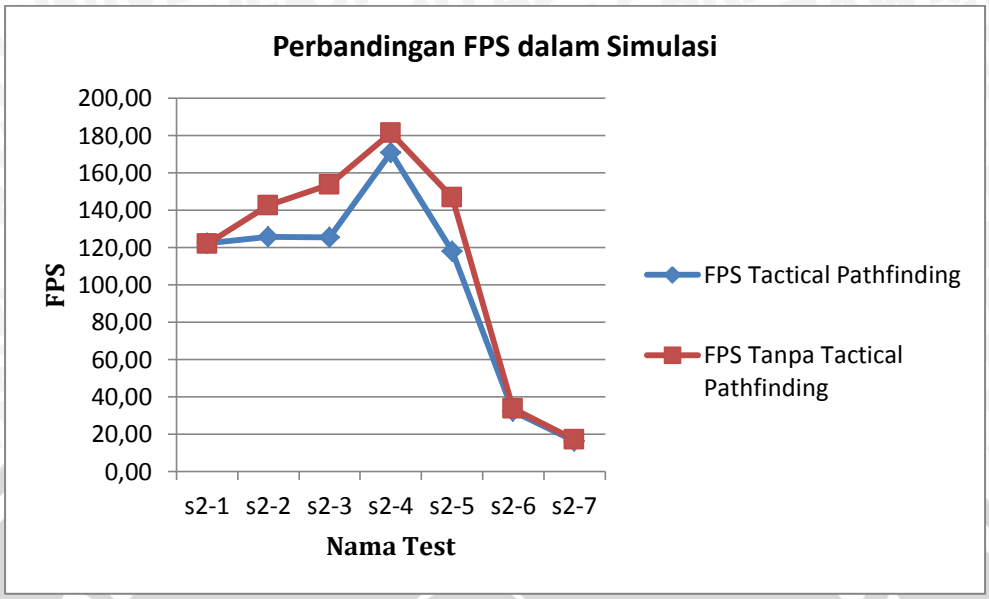


Gambar 6.4 Contoh Skenario 2
Sumber : [Pengujian dan Analisis]

Tabel 6.3 Tabel Hasil Pengujian FPS Skenario 2

No. Test	Jumlah AI Agen	Jumlah AI Turret	Tactical Pathfinding (FPS)	Tanpa Tactical Pathfinding (FPS)
1	4	5	122,48	121,92
2	3	5	125,86	142,60
3	12	5	125,43	153,83
4	6	5	170,94	181,41
5	3	5	118,00	146,85
6	28	5	32,30	33,68
7	100	5	16,45	17,29
AVG			**101,64**	**113,94**

Sumber : [Pengujian dan Analisis]



Gambar 6.5 Grafik Hasil Pengujian FPS Skenario 2
Sumber : [Pengujian dan Analisis]

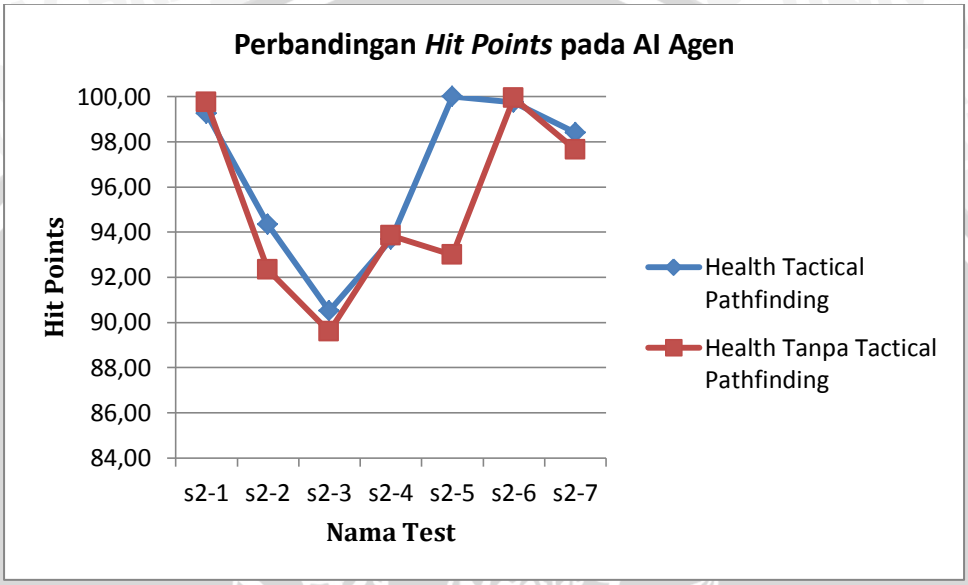
Hasil uji coba *hit points* skenario 2 untuk terdapat dalam Tabel 6.4, dijelaskan dalam grafik dalam Gambar 6.6. Hasil uji coba *hit points* skenario 2 terdapat dalam Tabel 6.4, dijelaskan dalam grafik dalam Gambar 6.6. Dalam Tabel 6.4 rata-rata *hit points* dengan *tactical pathfinding* adalah 96,56 dan rata-rata FPS simulasi permainan 3D tanpa *tactical pathfinding* adalah 95,15. Dalam Gambar 6.6, tampak dalam grafik bahwa 3 dari 7 uji coba *hit points* dengan *tactical pathfinding* lebih tinggi dibandingkan *hit points* tanpa *tactical pathfinding* dan grafik tidak berbanding lurus karena saat garis dengan *tactical pathfinding* menurun terdapat garis tanpa *tactical pathfinding* yang justru naik dan juga sebaliknya.

Tabel 6.4 Tabel Hasil Pengujian *Hit Points* Skenario 2

No. Test	Jumlah AI Agen	Jumlah AI Turret	Tactical Pathfinding (Poin)	Tanpa Tactical Pathfinding (Poin)
1	4	5	99,25	99,75
2	3	5	94,33	92,33
3	12	5	90,50	89,58
4	6	5	93,67	93,83

5	3	5	100,00	93,00
6	28	5	99,75	99,93
7	100	5	98,41	97,65
AVG			**96,56**	**95,15**

Sumber : [Pengujian dan Analisis]



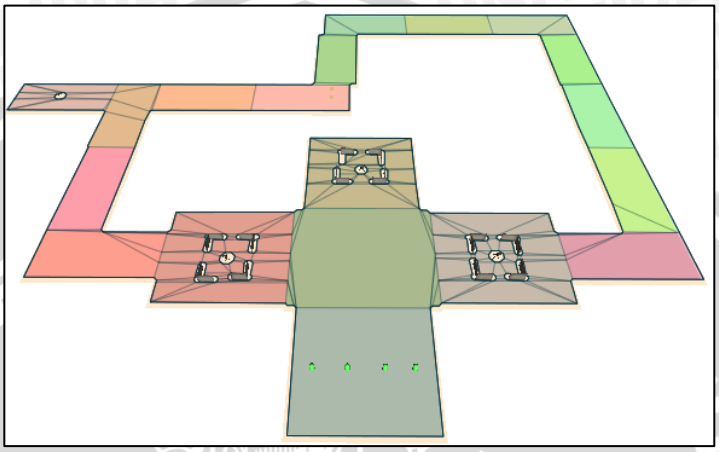
Gambar 6.6 Grafik Hasil Pengujian *Hit Points* Skenario 2
Sumber : [Pengujian dan Analisis]

6.1.3 Skenario 3

Uji coba skenario 3 adalah simulasi dengan perbandingan jumlah AI Agen dan AI Turret yang sama pada peta permainan yang berbeda. Simulasi skenario 3 dimaksudkan untuk menguji fungsionalitas AI Agen dan AI Turret dalam situasi yang seimbang. Contoh simulasi skenario 3 dijabarkan dalam Gambar 6.7.

Hasil uji coba FPS skenario 3 untuk terdapat dalam Tabel 6.5, dijelaskan dalam grafik dalam Gambar 6.8. Hasil uji coba FPS skenario 2 terdapat dalam Tabel 6.3, dijelaskan dalam grafik dalam Gambar 6.5. Dalam Tabel 6.5 rata-rata dari FPS simulasi permainan 3D dengan *tactical pathfinding* adalah 141,68 dan rata-rata FPS simulasi permainan 3D tanpa *tactical pathfinding* adalah 153,71.

Dalam Gambar 6.8, tampak dalam grafik bahwa 1 dari 7 uji coba FPS dengan *tactical pathfinding* lebih rendah dan berbanding lurus dengan FPS tanpa *tactical pathfinding.*

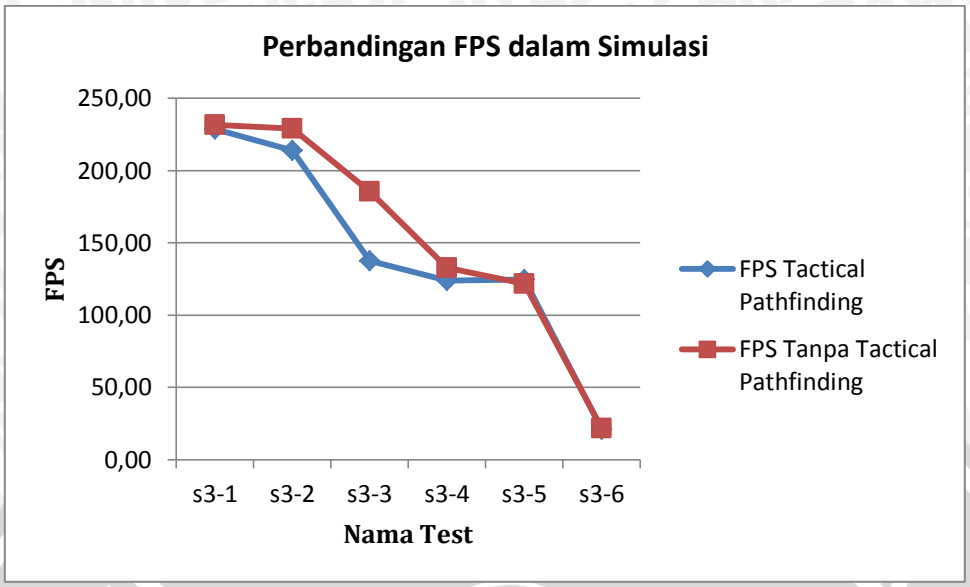


Gambar 6.7 Contoh Skenario 3
Sumber : [Pengujian dan Analisis]

Tabel 6.5 Tabel Hasil Pengujian FPS Skenario 3

No. Test	Jumlah AI Agen	Jumlah AI Turret	Tactical Pathfinding (FPS)	Tanpa Tactical Pathfinding (FPS)
1	1	1	228,65	231,79
2	2	2	214,01	228,88
3	3	3	137,68	185,68
4	4	4	124,04	132,58
5	5	5	124,65	121,77
6	50	50	21,07	21,56
AVG			**141,68**	**153,71**

Sumber : [Pengujian dan Analisis]



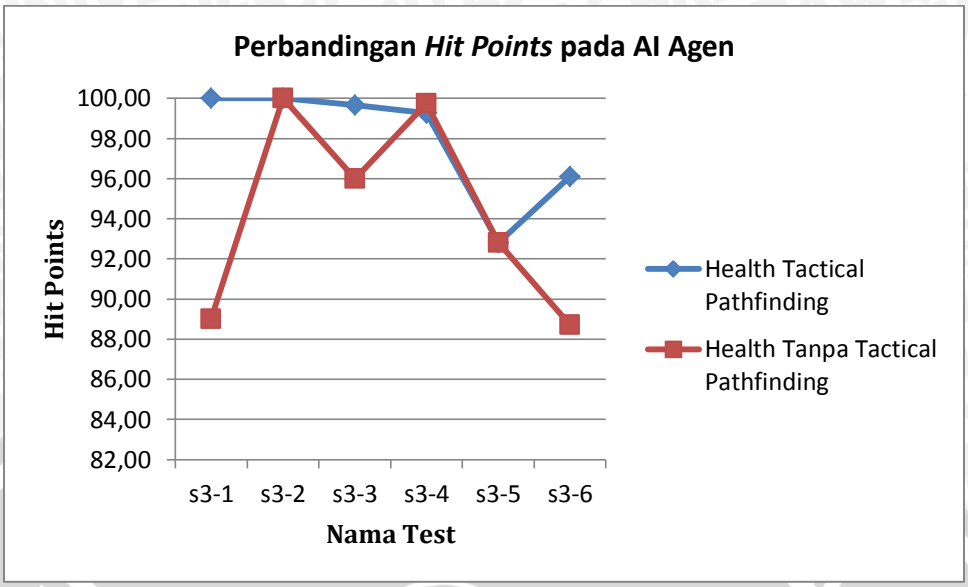
Gambar 6.8 Grafik Hasil Pengujian FPS Skenario 3
Sumber : [Pengujian dan Analisis]

Hasil uji coba *hit points* skenario 3 untuk terdapat dalam Tabel 6.6, dijelaskan dalam grafik dalam Gambar 6.9. Dalam Tabel 6.6 rata-rata *hit points* dengan *tactical pathfinding* adalah 97,97 dan rata-rata FPS simulasi permainan 3D tanpa *tactical pathfinding* adalah 94,38. Dalam Gambar 6.9, tampak dalam grafik bahwa 3 dari 6 uji coba *hit points* dengan *tactical pathfinding* tidak lebih tinggi dibandingkan *hit points* tanpa *tactical pathfinding* dan grafik tidak berbanding lurus karena saat garis dengan *tactical pathfinding* menurun terdapat garis tanpa *tactical pathfinding* yang justru naik.

Tabel 6.6 Tabel Hasil Pengujian *Hit Points* Skenario 3

No. Test	Jumlah AI Agen	Jumlah AI Turret	Tactical Pathfinding (Poin)	Tanpa Tactical Pathfinding (Poin)
1	1	1	100,00	89,00
2	2	2	100,00	100,00
3	3	3	99,67	96,00
4	4	4	99,25	99,75
5	5	5	92,80	92,80
6	50	50	96,1	88,72
AVG			**97,97**	**94,38**

Sumber : [Pengujian dan Analisis]



Gambar 6.9 Grafik Hasil Pengujian *Hit Points* Skenario 3
Sumber : [Pengujian dan Analisis]

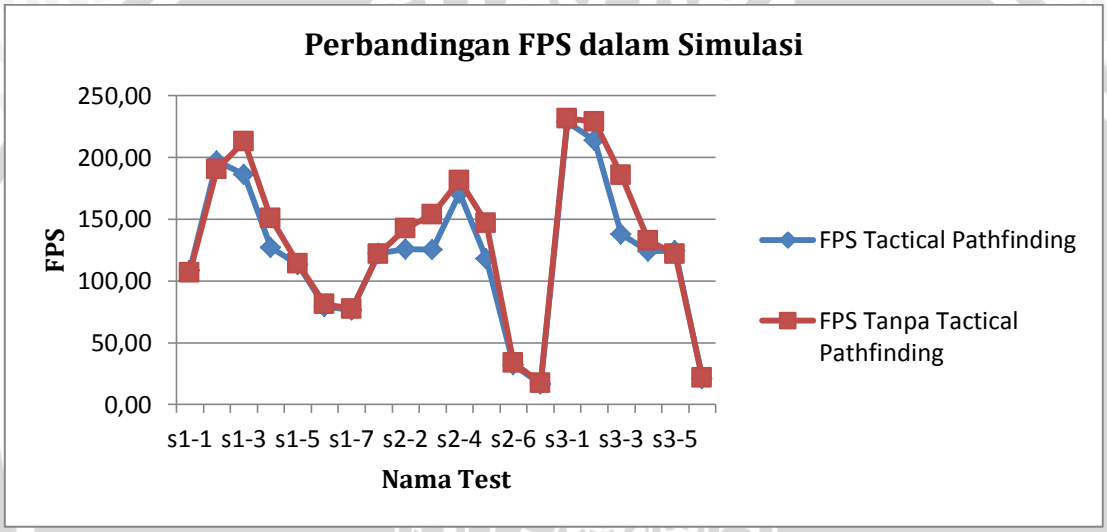
Data dari hasil pengujian FPS dari ketiga skenario dijelaskan dalam Tabel 6.7 dan gambar 6.10, dan data dari hasil pengujian *hit points* dari ketiga skenario dijelaskan dalam Tabel 6.8 dan gambar 6.11. Kedua tabel tersebut bertujuan menunjukkan rata-rata dari hasil pengujian pada seluruh simulasi untuk mempermudah analisis hasil uji.

Tabel 6.7 Tabel Hasil Pengujian FPS

Nama Test	Tactical Pathfinding (FPS)	Tanpa Tactical Pathfinding (FPS)
s1-1	108,64	106,61
s1-2	197,14	190,46
s1-3	186,17	213,04
s1-4	127,03	151,02
s1-5	113,33	113,97
s1-6	79,30	81,52
s1-7	76,75	77,58
s2-1	122,48	121,92
s2-2	125,86	142,60
s2-3	125,43	153,83
s2-4	170,94	181,41
s2-5	118,00	146,85

s2-6	32,30	33,68
s2-7	16,45	17,29
s3-1	228,65	231,79
s3-2	214,01	228,88
s3-3	137,68	185,68
s3-4	124,04	132,58
s3-5	124,65	121,77
s3-6	21,07	21,56
AVG	**122,50**	**132,70**

Sumber : [Pengujian dan Analisis]



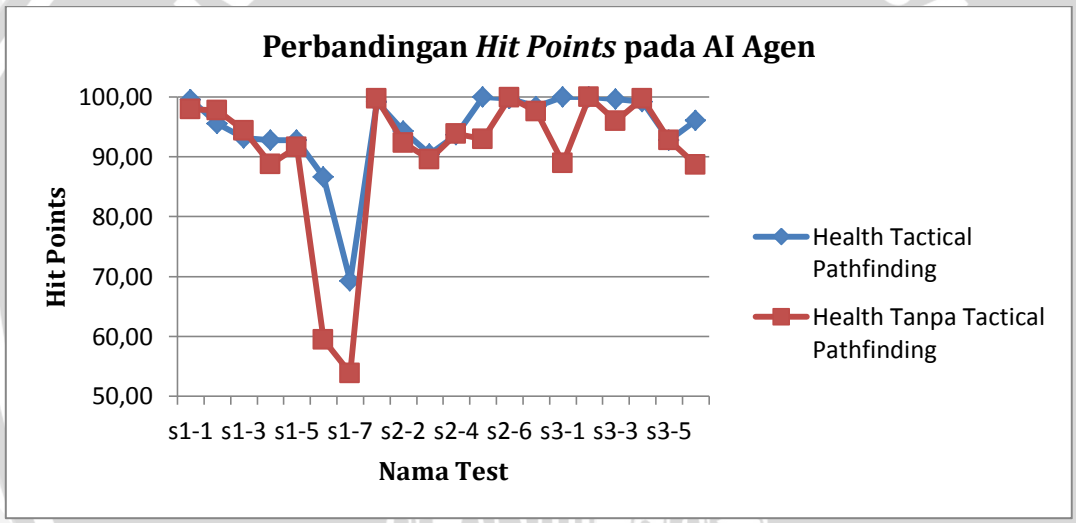
Gambar 6.10 Grafik Hasil Pengujian FPS
Sumber : [Pengujian dan Analisis]

Tabel 6.8 Tabel Hasil Pengujian *Hit Points*

Nama Test	Tactical Pathfinding (Points)	Tanpa Tactical Pathfinding (Points)
s1-1	99,60	98,00
s1-2	95,60	97,80
s1-3	93,20	94,40
s1-4	92,80	88,80
s1-5	92,80	91,60
s1-6	86,6	59,4
s1-7	69,2	53,8
s2-1	99,25	99,75
s2-2	94,33	92,33
s2-3	90,50	89,58

s2-4	93,67	93,83
s2-5	100,00	93,00
s2-6	99,75	99,93
s2-7	98,41	97,65
s3-1	100,00	89,00
s3-2	100,00	100,00
s3-3	99,67	96,00
s3-4	99,25	99,75
s3-5	92,80	92,80
s3-6	96,1	88,72
AVG	**94,68**	**90,81**

Sumber : [Pengujian dan Analisis]



Gambar 6.11 Grafik Pengujian *Hit Points*
Sumber : [Pengujian dan Analisis]

6.2 Analisis Uji Coba

Analisis bertujuan untuk mendapatkan kesimpulan dari hasil pengujian implementasi *tactical pathfinding* pada permainan 3D yang telah dilakukan. Proses analisis terhadap hasil pengujian peforma dilakukan dengan melihat hubungan antara *frame per second* yang dihasilkan dan rata-rata *hit points* akhir tiap Agent AI.

Pada uji coba skenario 1, yaitu simulasi dengan skenario tanpa mengubah jumlah AI Agen yaitu 5 agen untuk menuju ke titik tujuan dengan

merubah peta permainan dan jumlah AI Turret yang menghalangi AI Agen. Tabel 6.1 menunjukkan bahwa rata-rata FPS pada uji coba tanpa *tactical pathfinding* lebih baik daripada dengan *tactical pathfinding* dengan nilai rata-rata 133,46 untuk tanpa *tactical pathfinding* dan 126,91 dengan *tactical pathfinding.* Namun, pada uji coba 1 dan 2, ditemukan hasil yang berlawanan. Hal ini disebabkan karena jumlah AI Turret yang lebih sedikit dibandingkan dengan uji coba yang lain, sehingga penggunaan memori untuk perhitungan bobot dan komputasi algoritma tidak terlalu berpengaruh pada laju FPS.

Dalam Tabel 6.2 untuk hasil pengujian *hit points* pada skenario 1, menunjukkan bahwa rata-rata *hit points* pada uji coba dengan *tactical pathfinding* lebih baik daripada tanpa *tactical pathfinding* dengan nilai rata-rata 83,40 untuk tanpa *tactical pathfinding* dan 89,87 dengan *tactical pathfinding.*

Namun, pada uji coba 2 dan 3, ditemukan hasil yang berlawanan. Hal ini disebabkan karena pada uji coba 2 dan 3, AI Turret dan AI Agen tidak diletakkan pada posisi yang menyebar. Saat dilakukan uji coba tanpa *tactical pathfinding,* AI Agen bergerak berkelompok dan menempuh jarak yang sangat dekat dengan titik tujuan dibandingkan dengan uji coba dengan *tactical pathfinding,* sehingga AI Turret hanya memiliki waktu yang lebih sedikit untuk memberikan *damage* pada seluruh AI Agen.

Pada uji coba skenario 2, yaitu simulasi tanpa mengubah jumlah AI Turret yaitu 5 agen untuk menghalangi AI Agen dengan merubah peta permainan dan jumlah AI Agen yang akan menuju ke titik tujuan. Tabel 6.3 menunjukkan bahwa rata-rata FPS pada uji coba tanpa *tactical pathfinding* lebih baik daripada dengan *tactical pathfinding.*

Nilai rata-rata 113,94 untuk tanpa *tactical pathfinding* dan 101,64 dengan *tactical pathfinding.* Namun, hanya pada uji coba 1, ditemukan hasil yang berlawanan. Hal ini disebabkan karena peta permainan yang memungkinkan AI Agen untuk menjauh dari ancaman AI Turret, sehingga AI Turret hampir tidak pernah melakukan tembakan. Hal tersebut mengakibatkan

penggunaan memori yang lebih sedikit dan berdampak pada meningkatnya laju FPS.

Dalam Tabel 6.4 untuk hasil pengujian *hit points* pada skenario 2, menunjukkan bahwa rata-rata *hit points* pada uji coba dengan *tactical pathfinding* lebih baik daripada tanpa *tactical pathfinding* dengan nilai rata-rata 95,15 untuk tanpa *tactical pathfinding* dan 96,56 dengan *tactical pathfinding.*

Namun, hanya pada uji coba 1, 4, dan 6 ditemukan hasil yang berlawanan dengan selisih yang tipis yaitu 0,5 poin dan 0,2 poin. Hal ini disebabkan karena pada uji coba 1, 4 dan 6, AI Agen bergerak berkelompok, sehingga AI Turret hanya memiliki waktu yang lebih sedikit untuk memberikan *damage* pada seluruh AI Agen.

Pada uji coba skenario 3, yaitu dengan perbandingan jumlah AI Agen dan AI Turret yang sama pada peta permainan yang berbeda. Tabel 6.5 menunjukkan bahwa rata-rata FPS pada uji coba tanpa *tactical pathfinding* lebih baik daripada dengan *tactical pathfinding.*

Nilai rata-rata 153,71 untuk tanpa *tactical pathfinding* dan 141,68 dengan *tactical pathfinding.* Namun, hanya pada uji coba 5, ditemukan hasil yang berlawanan. Hal ini disebabkan karena peta permainan yang sederhana sehingga hasil FPS tidak terlalu jauh berbeda antara dengan atau tanpa *tactical pathfinding* dengan beda hanya 0,12 poin.

Dalam Tabel 6.6 untuk hasil pengujian *hit points* pada skenario 3, menunjukkan bahwa rata-rata *hit points* pada uji coba dengan *tactical pathfinding* lebih baik daripada tanpa *tactical pathfinding* dengan nilai rata-rata 94,38 untuk tanpa *tactical pathfinding* dan 94,38 dengan *tactical pathfinding.*

Namun, hanya pada uji coba 4, ditemukan hasil yang berlawanan dengan selisih yang tipis yaitu 0,5 poin. Hal ini disebabkan karena pada uji coba 4, AI Agen bergerak berkelompok, sehingga AI Turret hanya memiliki waktu yang lebih sedikit untuk memberikan *damage* pada seluruh AI Agen.

Berdasarkan hasil pengujian performa yang telah dilakukan pada dua puluh kasus uji coba yang berbeda, nilai *frame per second* yang didapatkan memiliki rata-rata 122,50 FPS untuk simulasi dengan *tactical pathfinding* dan 132,70 FPS untuk simulasi tanpa *tactical pathfinding*. Data dalam Tabel 6.7 membuktikan bahwa *tactical pathfinding* memakai memori lebih banyak sehingga nilai FPS menjadi lebih sedikit. Hal ini disebabkan karena *tactical pathfinding* membutuhkan memori lebih untuk melakukan perhitungan dan pemberian bobot pada daerah *navigation mesh.*

Hasil pengujian simulasi ditinjau dari *hit points* yang ditunjukkan oleh data dalam tabel 6.8, didapatkan rata-rata 94,68 poin untuk simulasi dengan *tactical pathfinding* dan 90,81 poin untuk simulasi tanpa *tactical pathfinding*. Data tersebut membuktikan bahwa *tactical pathfinding* mampu mengurangi *damage* yang diterima oleh AI Agen. Hal ini disebabkan karena dengan *tactical pathfinding,* agen dapat menuju ke titik tujuan melalui jalur yang lebih aman, meskipun pada sebagian kecil simulasi membuktikan sebaliknya. Seperti ditunjukkan pada tes skenario 1 nomor 2 dan pada tes skenario 1 nomor 3. Hal ini dikarenakan pada dua kasus tersebut pada jalur taktis, salah satu agen lebih lama terlihat oleh AI Turret, dibandingkan pada jalur yang taktis, sehingga menurunkan rata-rata *hit points* per simulasi.

6.2.1 Uji t (*Paired Sample t-test*)

Uji t (*Paired Sample t-test*) digunakan untuk membandingkan rata-rata dua grup yang saling berpasangan. Sampel berpasangan dapat diartikan sebagai sebuah sampel dengan subjek yang sama namun mengalami 2 perlakuan atau pengukuran yang berbeda, yaitu pengukuran sebelum dan sesudah dilakukan sebuah perlakuan.

Untuk pengujian uji t (*Paired Sample t-test*) pada sampel FPS dilakukan dengan hipotesis sebagai berikut:

- H0, artinya tidak ada pengaruh signifikan *tactical pathfinding* terhadap rata-rata FPS pada permainan 3D.

- H1, artinya ada pengaruh signifikan *tactical pathfinding* terhadap rata-rata FPS pada permainan 3D.

Pengujian uji t (*Paired Sample t-test*) dilakukan dengan cara membandingkan antara nilai t hitung dengan t tabel dengan *Level of significant* (α) dengan nilai α = 0,05 sehingga dapat diketahui suatu hipotesis diterima atau tidak jika:

- H0 diterima apabila –t tabel ≤ t hitung ≤ t tabel artinya tidak ada pengaruh signifikan *tactical pathfinding* terhadap rata-rata FPS pada permainan 3D.
- H0 ditolak apabila t hitung > t tabel atau –t hitung < -t tabel, artinya ada pengaruh signifikan *tactical pathfinding* terhadap rata-rata FPS pada permainan 3D.

Tabel 6.9 Hasil Uji t (*Paired Sample t-test*) pada FPS

t-Test: Two-Sample Assuming Equal Variances		
	Tactical Pathfinding	*Tanpa Tactical Pathfinding*
Mean	122,4960997	132,702265
Variance	3464,164842	4080,09638
Observations	20	20
Pooled Variance	3772,130611	
Hypothesized Mean Difference	0	
df	38	
t Stat	-0,525495784	
P(T<=t) one-tail	0,301145875	
t Critical one-tail	1,68595446	
P(T<=t) two-tail	0,602291751	
t Critical two-tail	2,024394164	

Sumber : [Pengujian dan Analisis]

Berdasarkan hasil pengujian uji t (*Paired Sample t-test*) dalam Tabel 6.9 menunjukkan bahwa: t tabel (-2,024) t hitung ≤ (-0,525) ≤ t tabel (2,024), yang berarti Ho tidak ditolak , artinya tidak ada pengaruh signifikan *tactical pathfinding* terhadap rata-rata FPS pada permainan 3D.

Untuk pengujian uji t (*Paired Sample t-test*) pada sampel *hit points* dilakukan dengan hipotesis sebagai berikut:

- H0, artinya tidak ada pengaruh signifikan *tactical pathfinding* terhadap rata-rata *hit points* pada *non player character* yaitu AI Agen.
- H1, artinya ada pengaruh signifikan *tactical pathfinding* terhadap rata-rata *hit points* pada *non player character* yaitu AI Agen.

Pengujian uji t (*Paired Sample t-test*) dilakukan dengan cara membandingkan antara nilai t hitung dengan t tabel dengan *Level of significant* (α) dengan nilai α = 0,05 sehingga dapat diketahui suatu hipotesis diterima atau tidak jika:

- H0 diterima apabila –t tabel ≤ t hitung ≤ t tabel artinya tidak ada pengaruh signifikan *tactical pathfinding* terhadap rata-rata *hit points* pada *non player character* yaitu AI Agen.
- H0 ditolak apabila t hitung > t tabel atau –t hitung < -t tabel, artinya ada pengaruh signifikan *tactical pathfinding* terhadap rata-rata *hit points* pada *non player character* yaitu AI Agen.

Tabel 6.10 Hasil Uji t (*Paired Sample t-test*) pada *Hit Points*

t-Test: Two-Sample Assuming Equal Variances		
	Tactical Pathfinding	*Tanpa Tactical Pathfinding*
Mean	94,67633333	90,80742857
Variance	50,74528409	152,8934796
Observations	20	20
Pooled Variance	101,8193819	
Hypothesized Mean Difference	0	
df	38	
t Stat	1,212475051	
P(T<=t) one-tail	0,116407539	
t Critical one-tail	1,68595446	
P(T<=t) two-tail	0,232815079	
t Critical two-tail	2,024394164	

Sumber : [Pengujian dan Analisis]

Berdasarkan hasil pengujian uji t (*Paired Sample t-test*) dalam Tabel 6.10 menunjukkan bahwa: t tabel (-2,024) t hitung ≤ (1,212) ≤ t tabel (2,024), yang berarti H0 tidak ditolak , artinya tidak ada pengaruh signifikan *tactical pathfinding* terhadap rata-rata *hit points* pada *non player character* yaitu AI Agen.

BAB VII
PENUTUP

7.1 Kesimpulan

Berdasarkan hasil pengujian dan analisis yang dilakukan terhadap implementasi algoritma, dapat diambil kesimpulan sebagai berikut:

1. Implementasi algoritma *tactical pathfinding* dalam permainan 3D dilakukan berdasarkan algoritma pencarian jalur berdasarkan A* ditambah perhitungan bobot pada peta berbasis *navigation mesh.*
2. Implementasi *tactical pathfinding* terbukti mampu mengurangi *damage* yang diterima oleh *non-player character* AI Agen, dibandingkan dengan pencarian jalur secara tradisional. Terbukti dari rata-rata *hit points* dari semula 90,81 meningkat menjadi 94,68. Berdasarkan uji t yang telah dilakukan, perbedaannya tidak signifikan, implementasi algoritma dapat digunakan pada *non-player character* untuk mengurangi *damage* yang diterima walaupun tidak signifikan.
3. Implementasi *tactical pathfinding* terbukti membutuhkan komputasi algoritma dan memori yang lebih banyak dibandingkan pencarian jalur tradisional, dilihat dari jumlah *frame per second* yang lebih sedikit dibandingkan dengan pencarian jalur secara tradisional. Terbukti dari rata-rata FPS dari 132,70 menjadi 122,50. Berdasarkan uji t yang telah dilakukan, perbedaannya tidak signifikan, implementasi algoritma cocok untuk permainan 3D.

7.2 Saran

Untuk meningkatkan hasil yang telah dicapai dari penelitian ini dapat dilakukan beberapa perbaikan sebagai berikut:

1. Diharapkan pada penelitian selanjutnya, peta permainan bersifat *real-time* sehingga dapat memantapkan implementasi algoritma *tactical pathfinding* pada permainan 3D secara umum.

2. Menambahkan algoritma pencarian jalur heuristik *Fast Subgoaling* [HER-11], pada algoritma *tactical pathfinding* supaya dapat melakukan pencarian jalur taktis dan *real-time*.

DAFTAR PUSTAKA

[ALD-09] Aldrich, Clark. 2009. *The Complete Guide to Simulations and Serious Games: How the Most Valuable Content Will be Created in the Age Beyond Gutenberg to Google*. John Wiley and Sons. Hoboken, New Jersey 2009.

[CHE-09] Chen, Siyuan, et al. 2009. *Fast Path Searching in Real Time 3D Game*. 2009 Global Congress on Intelligent Systems.

[CUI-11] Cui, Xiao. Shi, Hao. 2011. *A*-based Pathfinding in Modern Computer Games.* IJCSNS International Journal of Computer Science and Network Security, VOL.11 No.1, Januari 2011.

[CUI-12] Cui, Xiao. Shi, Hao. 2012. *An Overview of Pathfinding in Navigation Mesh.* IJCSNS International Journal of Computer Science and Network Security, VOL.12 No.12, Desember 2012.

[DAL-09] Daly, Kyle. 2009. *Design, Implement and Evaluate the Effectiveness and Performance of a Dynamic Tactical Game Agent Position Evaluation System.* University of Bolton Computer Games Software Development Dissertation .

[HER-11] Hernández, Carlos, et al. 2011. *Fast Subgoaling for Pathfinding via Real-Time Search.* Proceedings of the Twenty-First International Conference on Automated Planning and Scheduling, Freiburg, Juni 2011.

[JIE-11] Jie, Jiang, et al. 2011. *The Application of AI for the Non Player Character in Computer Games* . 2011 International Conference on Computational and Information Sciences.

[KIT-10] Kittrel, Brian. 2010. *d10 Core Roleplaying System.* CreateSpace, California 2010.

[LEI-10] Lei, Mao. 2010. *Navigation Mesh Construction and Path_Finding for Architecture Environment.* Department Of Computer Science National University Of Singapore Thesis, Singapore 2010.

[HAL-10] Hale, D. Hunter, et al. 2010. *Using Intelligent Agents to Build Navigation Meshes.* Proceedings of the Twenty-Third International Florida Artificial Intelligence Research Society Conference (FLAIRS 2010)

[MCJ-07] McCarthy, John. 2007. *What is Artificial Intelligence?* . Stanford University Computer Science Department. http://www-formal.stanford.edu/jmc/whatisai/whatisai.html

[MIL-09] Millington, Ian. Funge, John. *Artificial Intelligence for Games Second Edition.* Morgan Kaufmann. Burlington, Massachusetts 2009.

[NIU-08] Niu, Lei, et al. 2008. *An Improved Real 3D A* Algorithm For Difficult Path Finding Situation.* The International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences. Vol. XXXVII. Part B4. Beijing 2008.

[ROG-10] Roger, Scott. 2010. *Level Up! The Guide to Great Video Game Design*. John Wiley & Sons Ltd, Chichester 2010.

[REY-13] Reynold, Craig. 2004. *Seek and Flee steering behaviors.* http://www.red3d.com/cwr/steer/SeekFlee.html. Diakses tanggal 4 April 2013

[RUS-10] Russel, Stuart. Norvig, Peter. 2010. *Artificial Intelligence. A Modern Approach. Third Edition.* Prentice Hall, New Jersey 2010.

[SCH-08] Schell, Jesse. 2008. *The Art of Game Design: A book of lenses*. CRC Press, Florida 2008.

[SCH-09] Schwab, Brian. 2009. *AI Game Engine Programming*. Course Technology, Boston 2009.

[STR-02] Sterren, William van der. 2002. *Game Programming Gems 3*. Charles River Media, Massachusetts 2002.

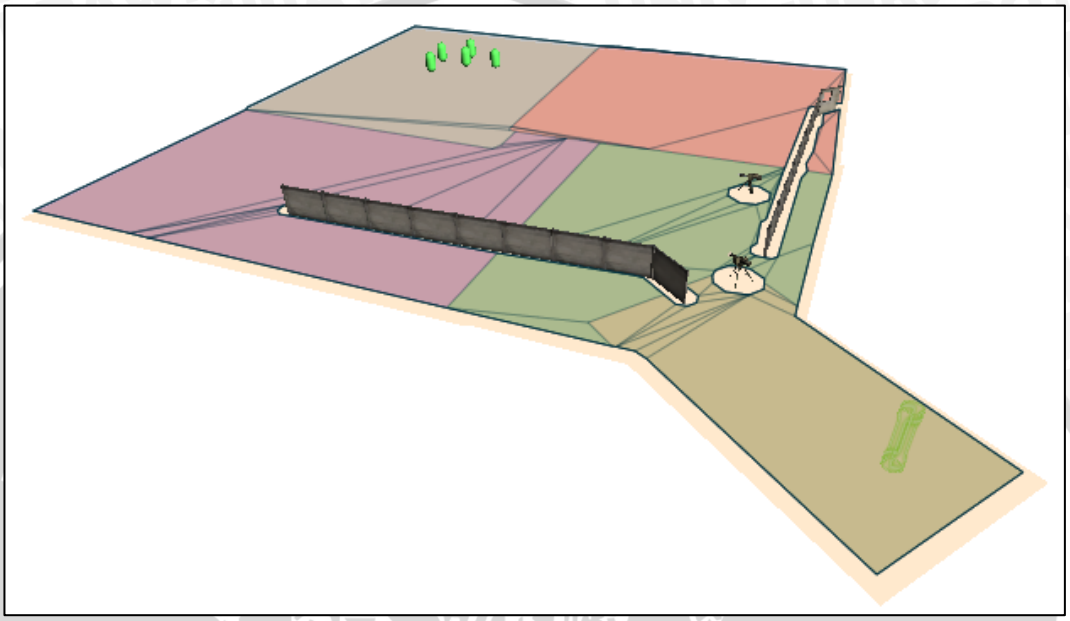
[WAN-09] Wan, Haifeng, et al. 2009. *The M2M Pathfinding Algorithm Based on the Idea of Granular Computing*. 2009 IEEE/WIC/ACM International Joint Conference on Web Intelligence and Intelligent Agent Technology.

[ZIM-03] Zimmerman, Eric. Salen, Katie. 2003. *Rules of Play - Game Design Fundamentals*. The MIT Press Cambridge, Massachusetts London, England 2004.

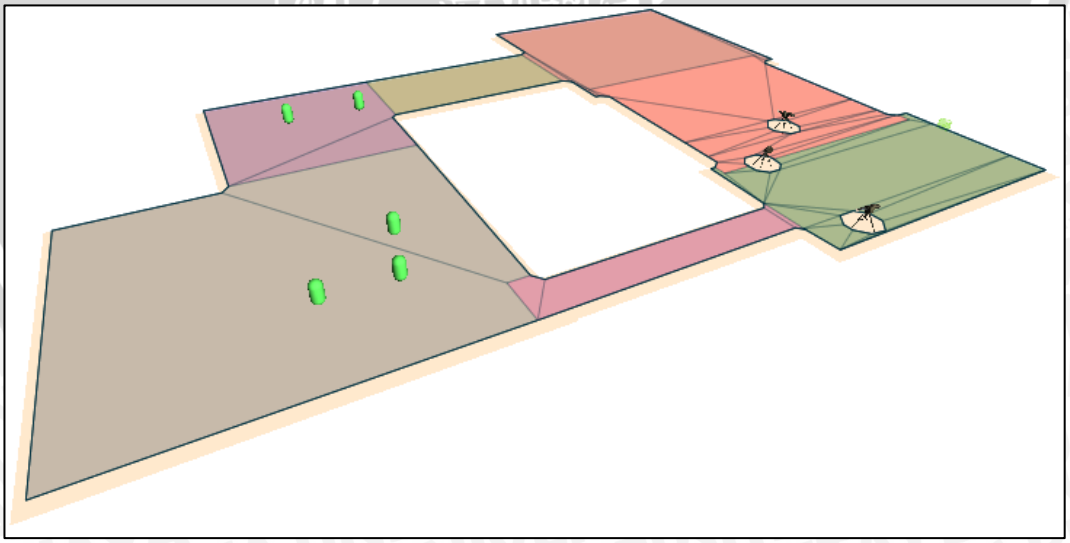
LAMPIRAN I

DATASET UJI COBA SKENARIO 1

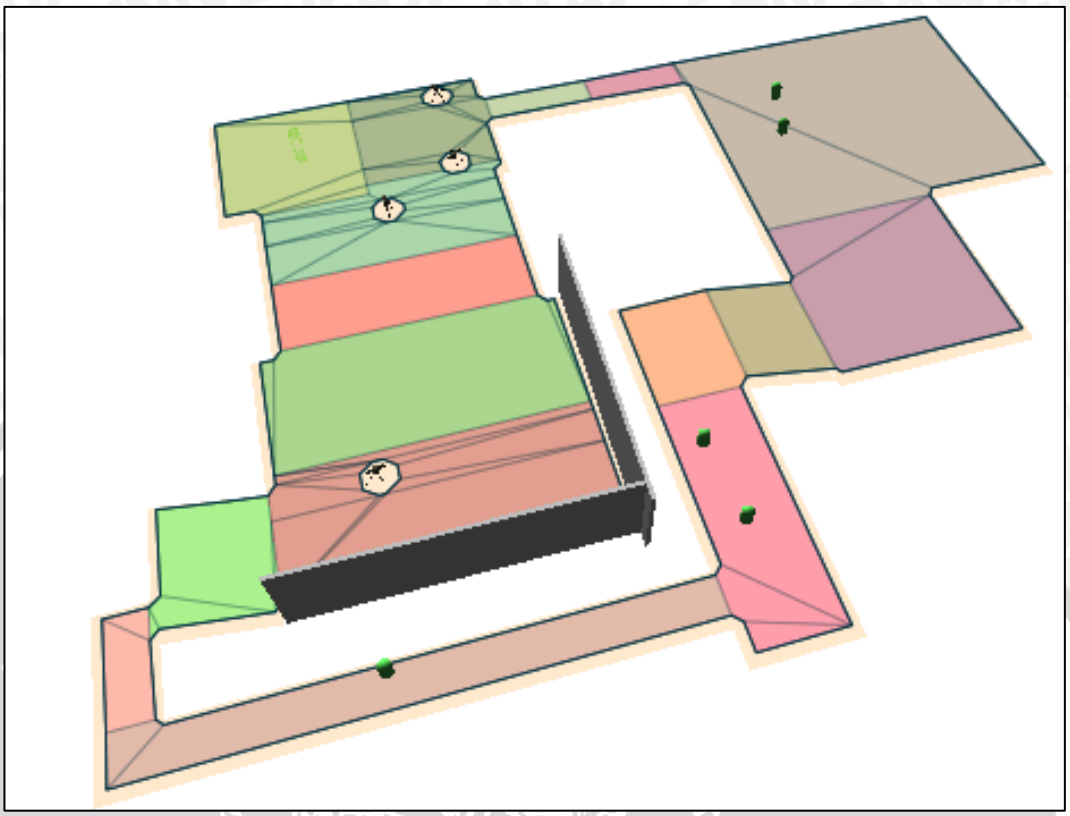
1. S1-1.scene



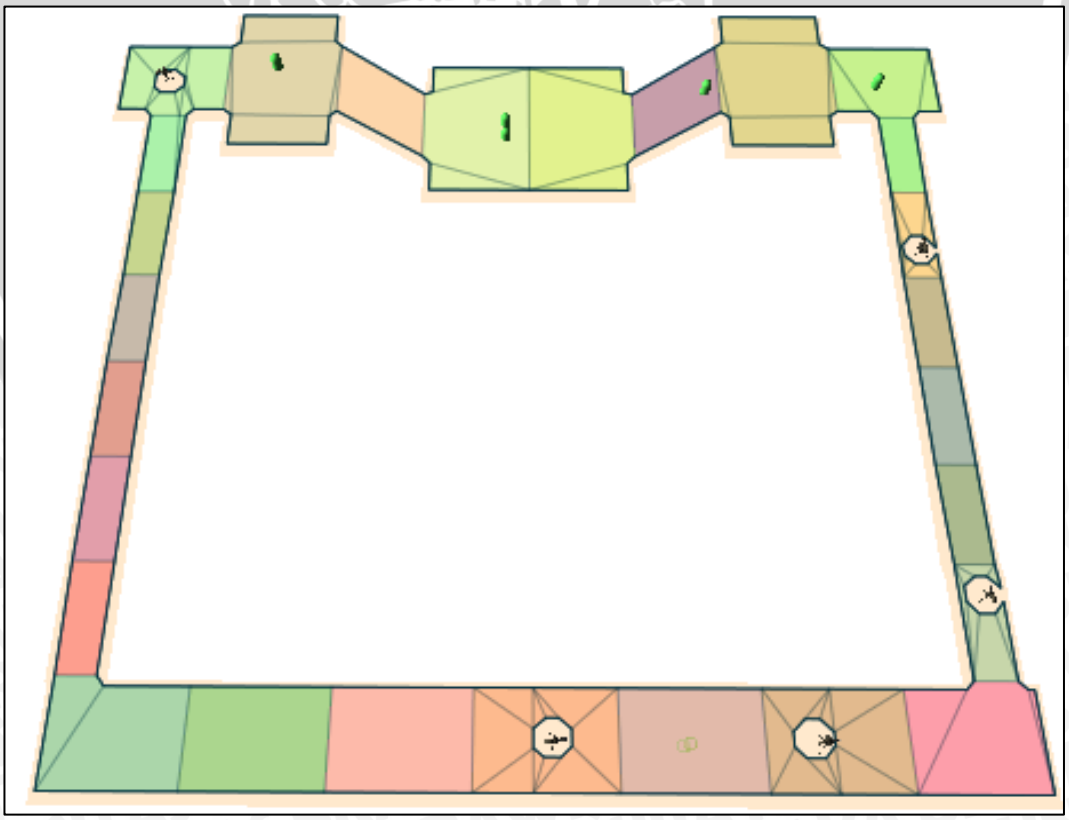
2. S1-2.scene



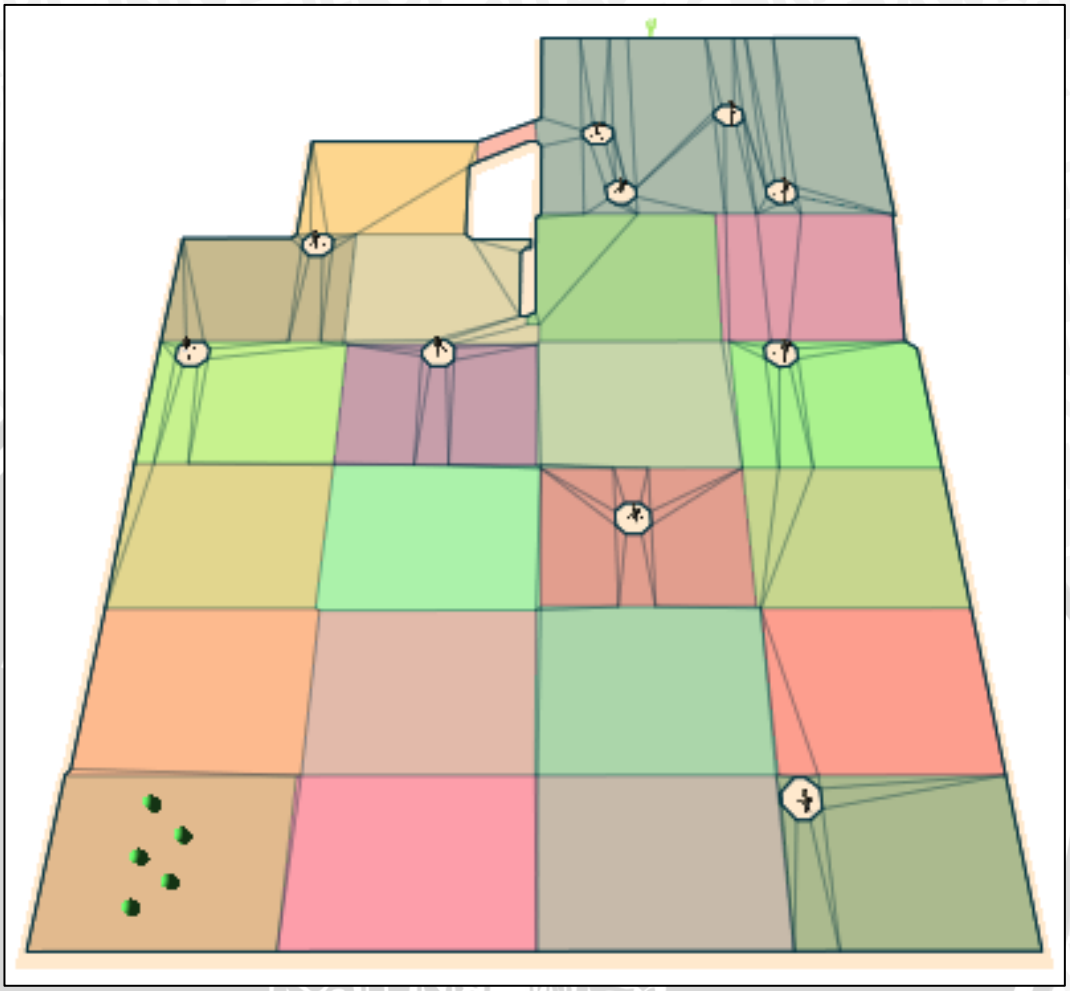
3. S1-3.scene



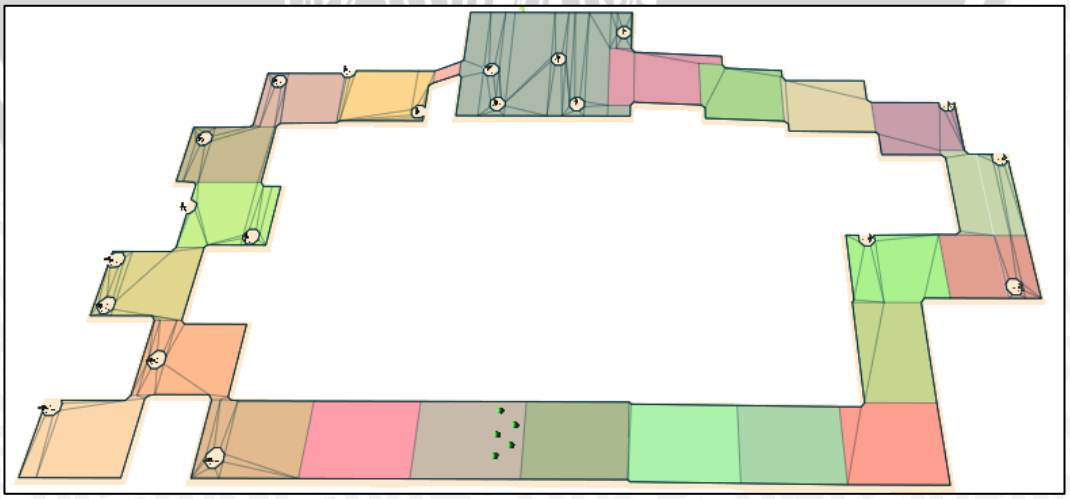
4. S1-4.scene



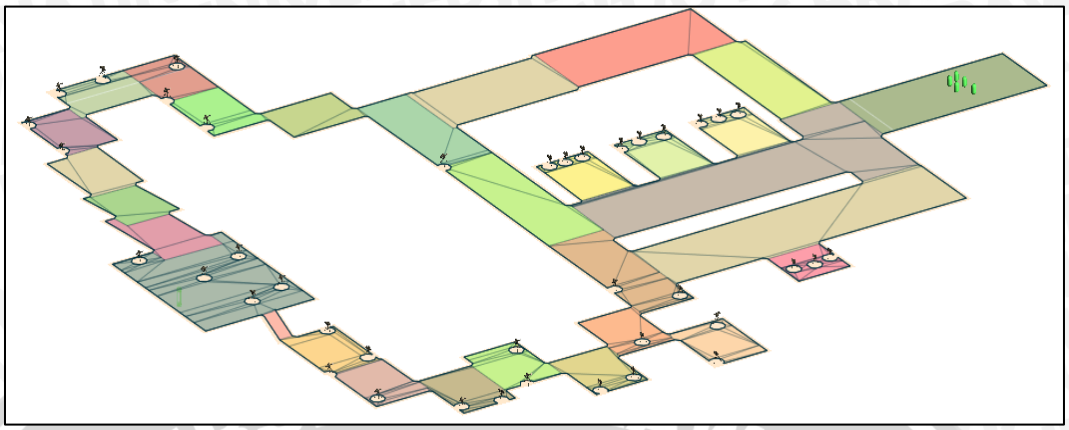
5. S1-5.scene



6. S1-6.scene

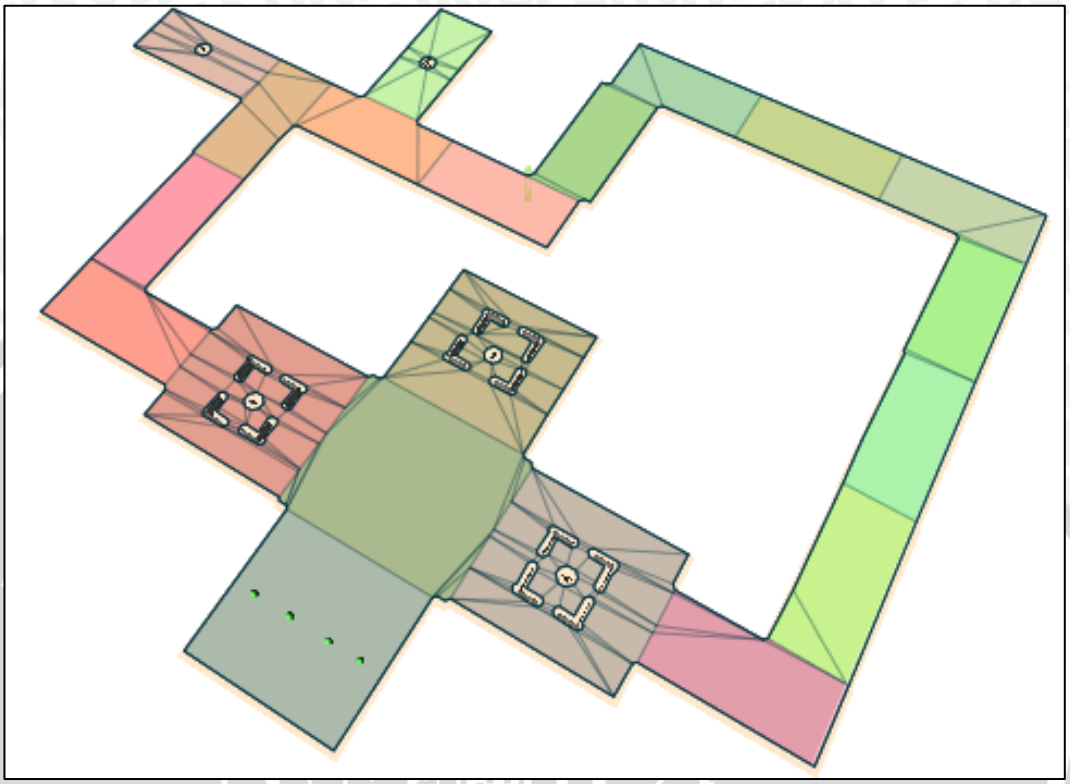


7. S1-7.scene

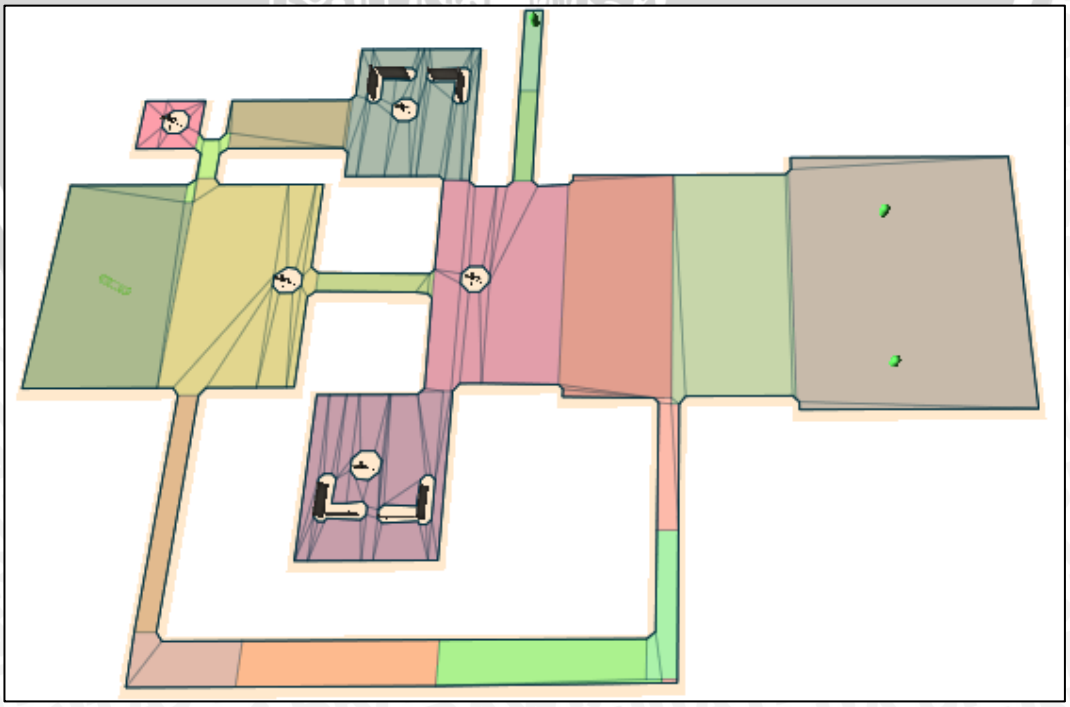


DATASET UJI COBA SKENARIO 2

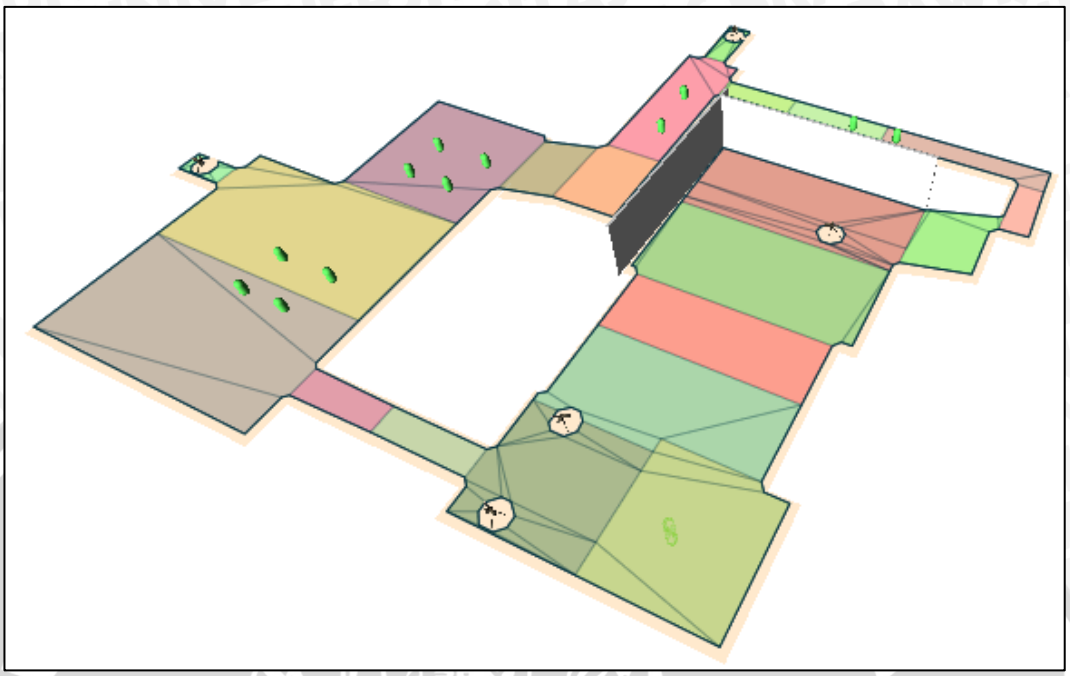
1. S2-1.scene



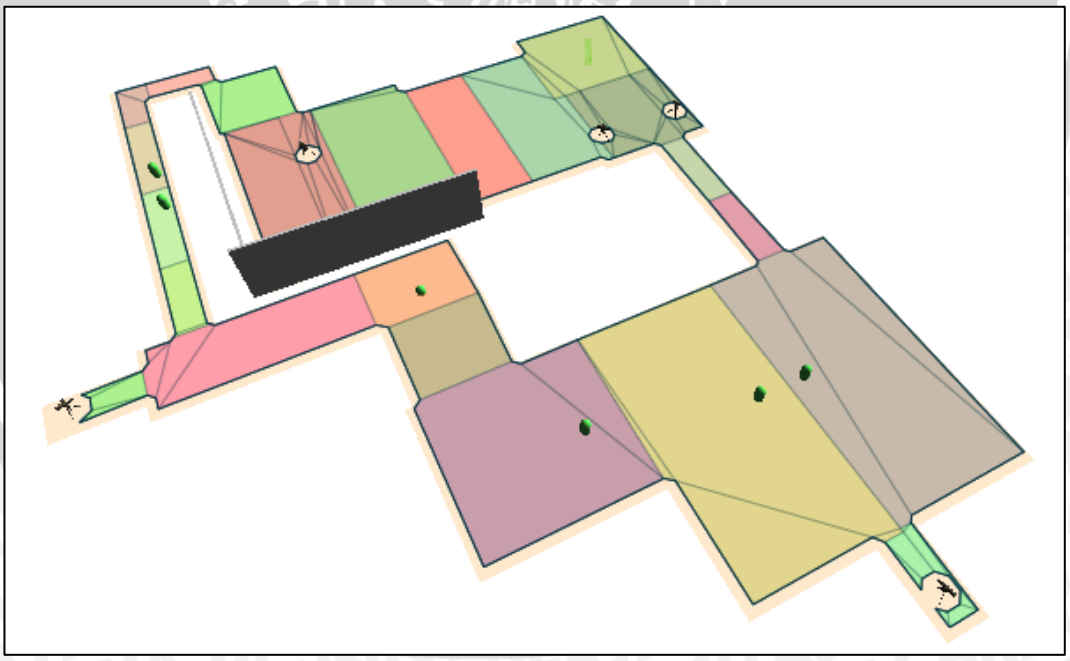
2. S2-2.scene



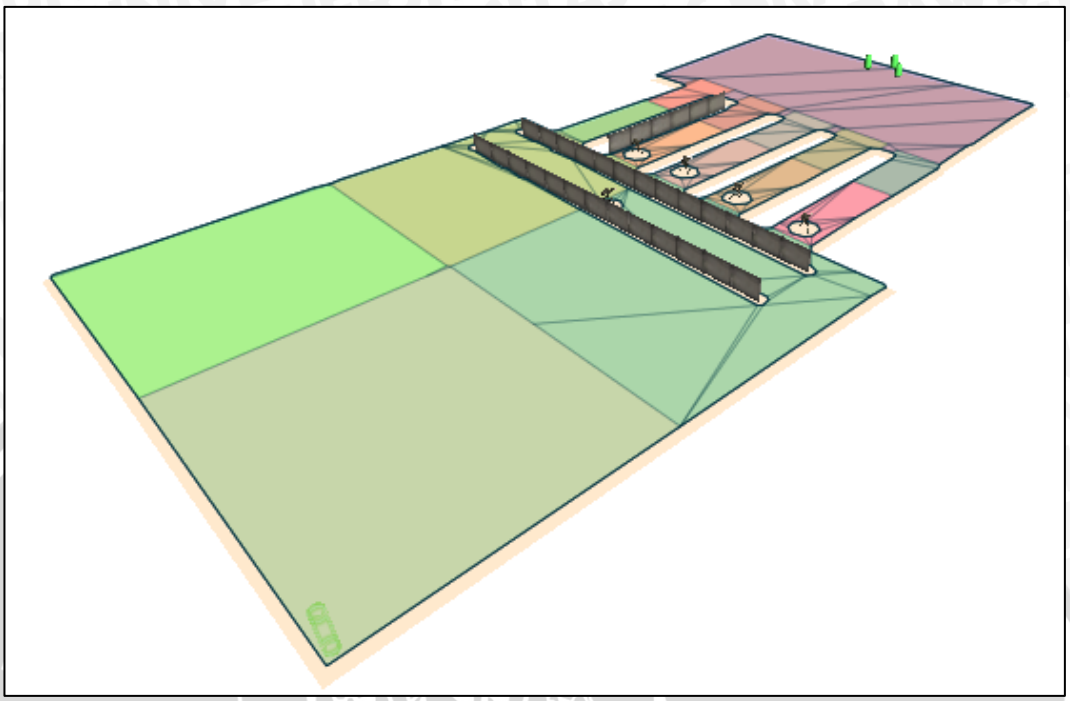
3. S2-3.scene



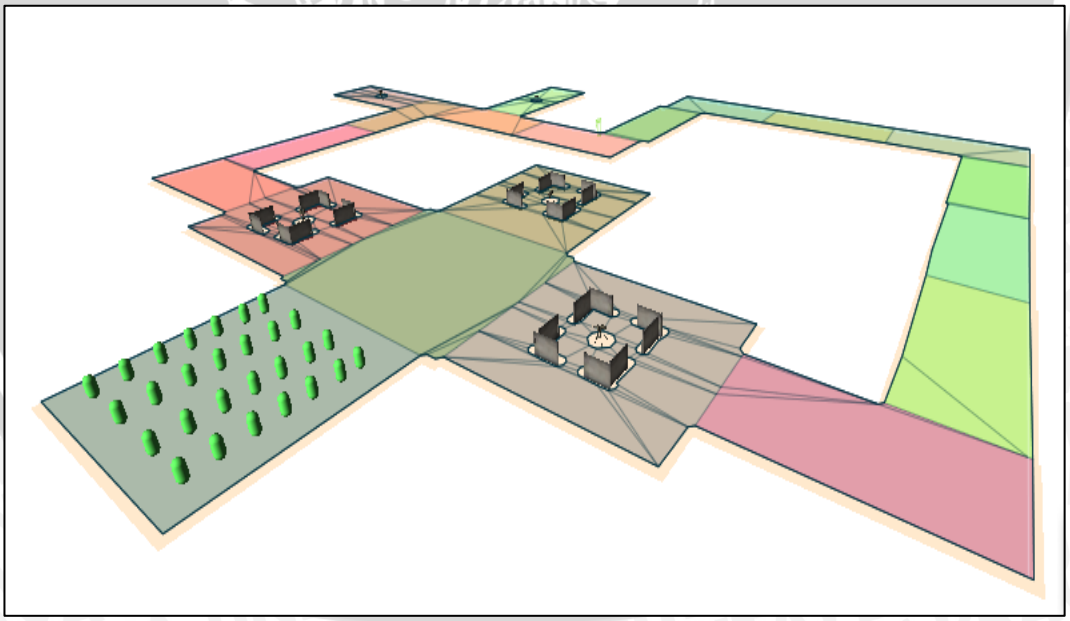
4. S2-4.scene



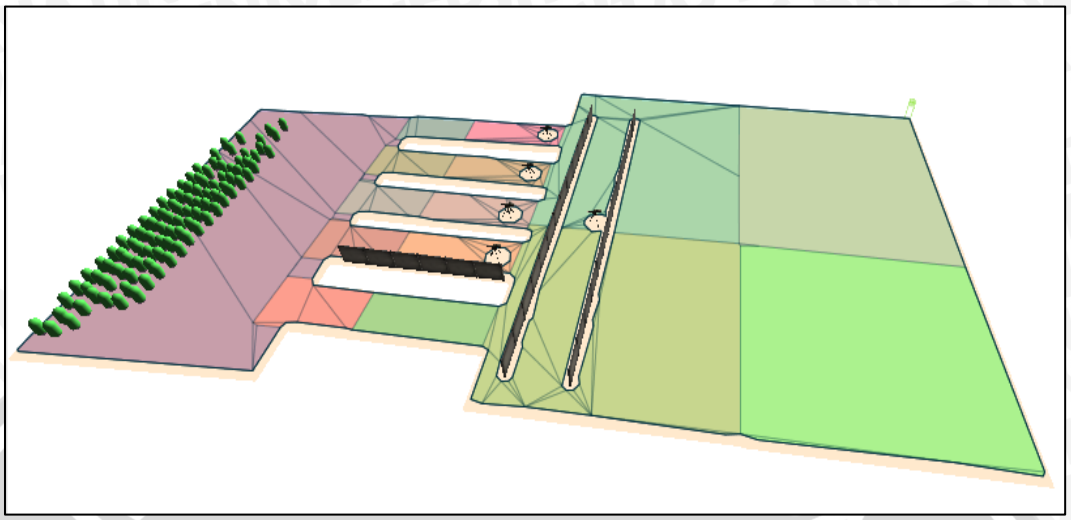
5. S2-5.scene



6. S2-6.scene

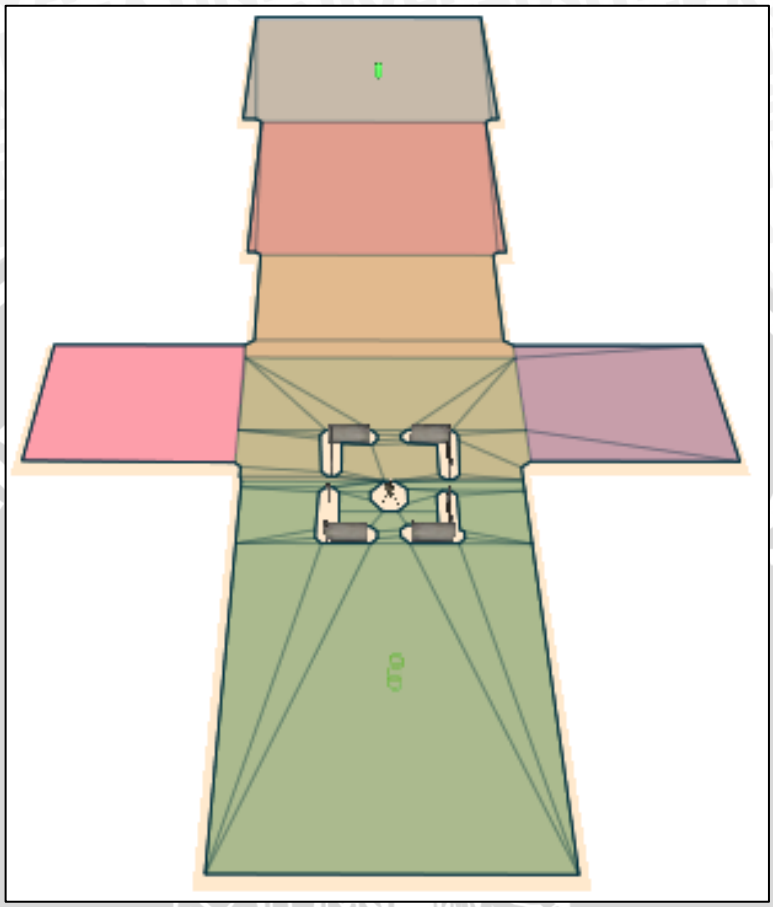


7. S2-7.scene

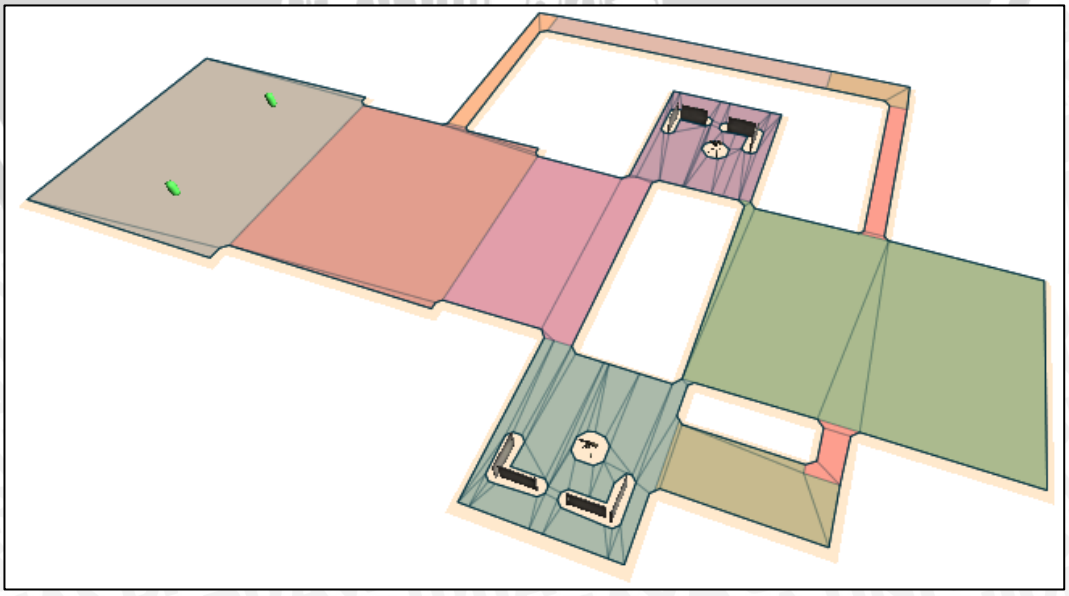


DATASET UJI COBA SKENARIO 3

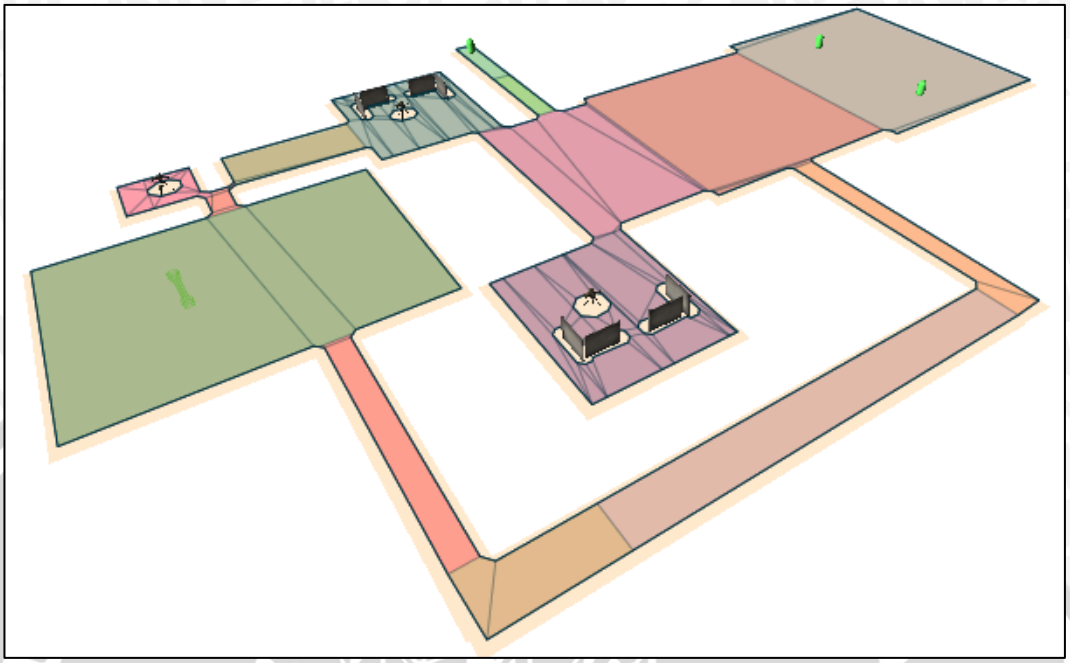
1. s3-1.scene



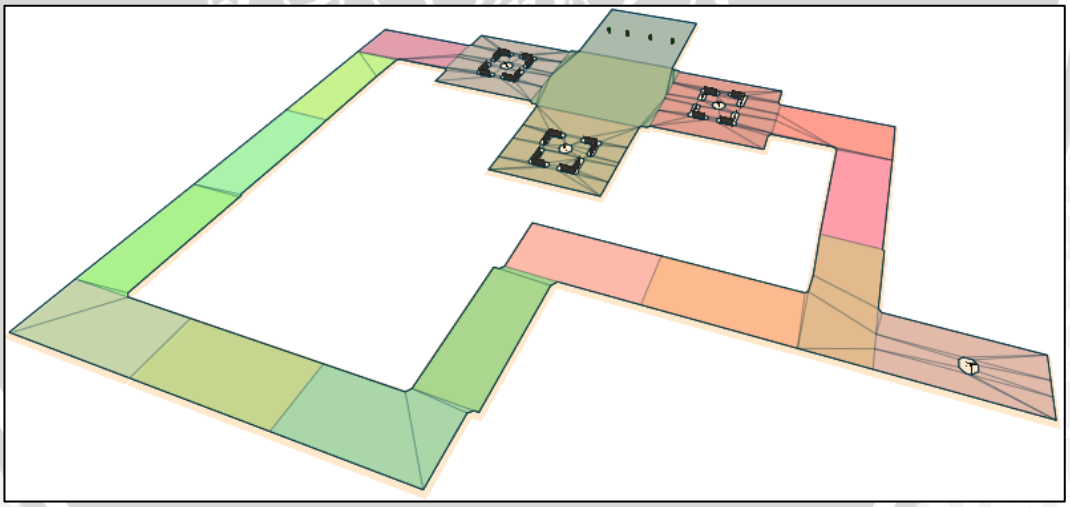
2. s3-2.scene



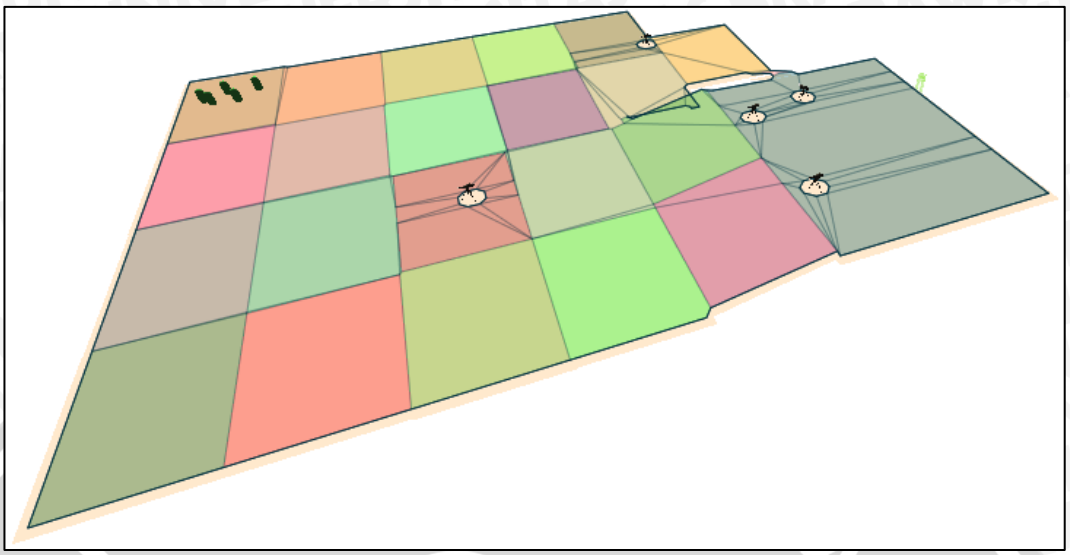
3. s3-3.scene



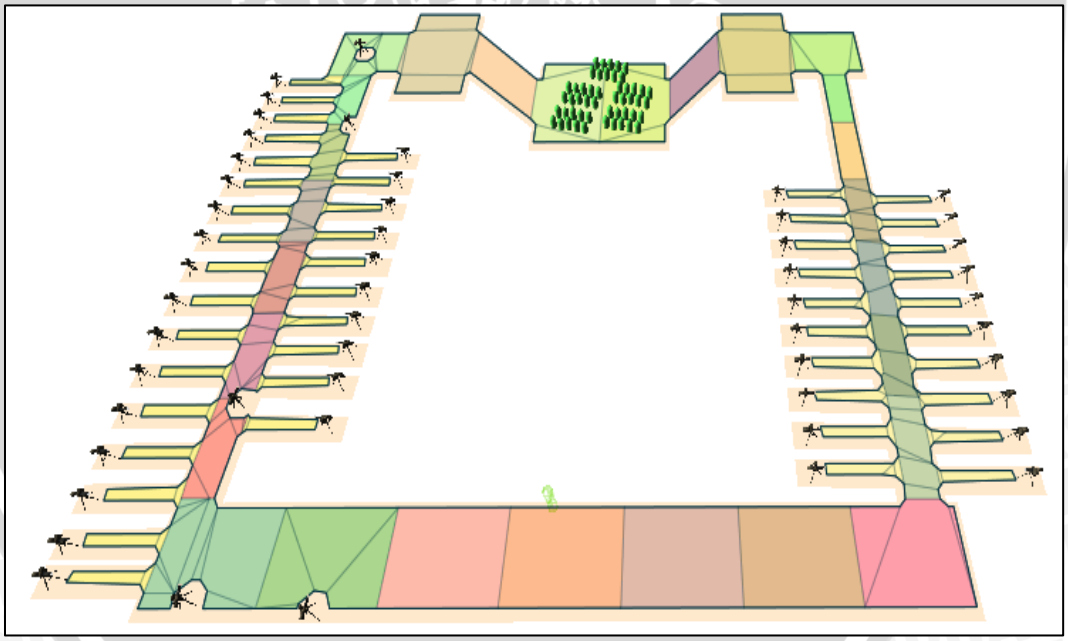
4. s3-4.scene



5. s3-5.scene



6. s3-6.scene

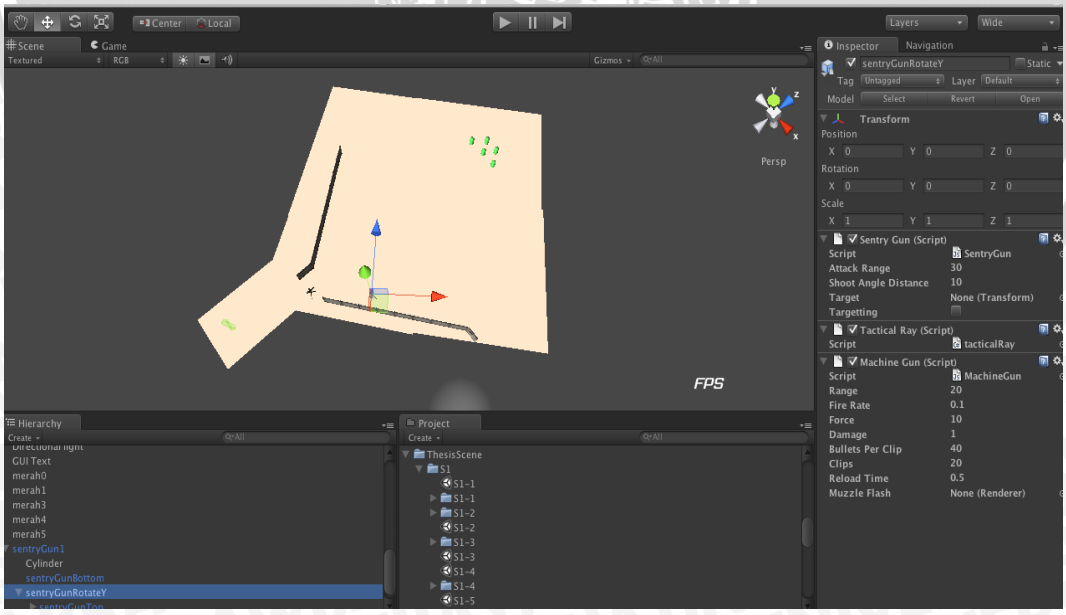


LAMPIRAN II
MANUAL PROGRAM

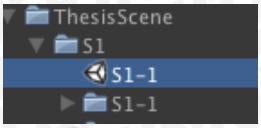
1. *Minimum Requirement* untuk menjalankan simulasi:
    - Intel Dual Core Processor
    - 2GB RAM
    - Windows: XP SP2, Mac OS X Snow Leopard 10.6
    - Kartu Grafis dengan DirectX level 9
    - Unity 3D 3.5

2. Petunjuk menjalankan simulasi:
    - Buka aplikasi *game engine* Unity

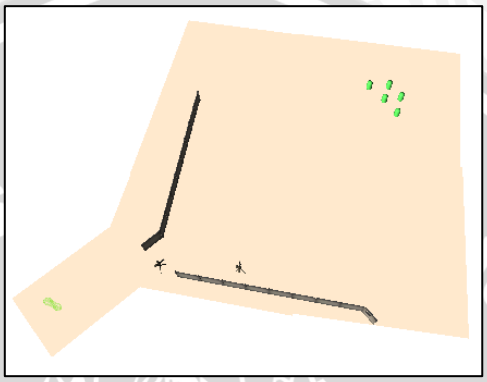
    - Maka muncul UI dari Unity



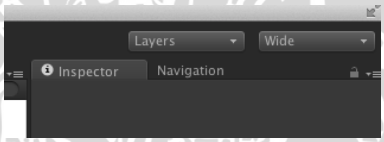
- Pilih *scene* yang ingin disimulasikan



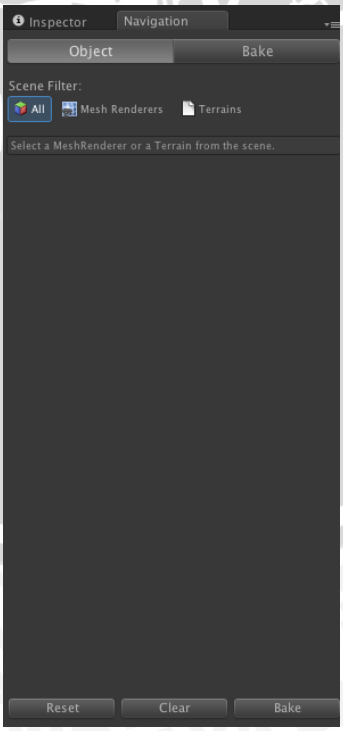
- Maka di tab *scene* akan muncul objek-objek yang ada pada *scene* S1-1



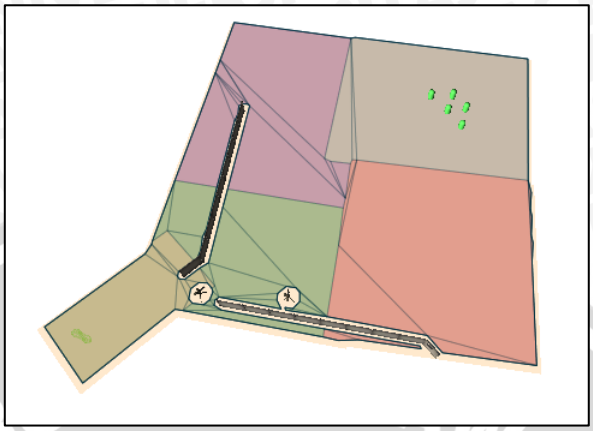
- Untuk *default*, *navigation mesh* belum digenerate, pilih tab *navigation*



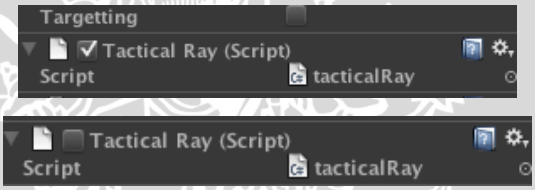
- Maka tab akan berpindah ke tab navigation, kemudian klik ‘bake’



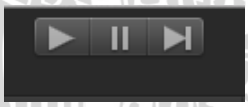
- Sistem akan menggeneralisasi *navigation mesh* pada peta permainan



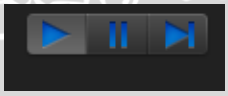
- Untuk *default,* AI Turret akan mengeluarkan *ray* untuk memberikan bobot pada peta, untuk membuat non-aktif, klik pada centang dalam tab '*inspector*'



- Untuk memulai simulasi, klik tombol dengan simbol '*play*' yang masih berwarna putih



Untuk mengakhiri simulasi, klik tombol dengan simbol '*play*' yang berwarna biru



- Setelah simulasi berakhir, maka di folder *project* akan muncul myFPS.xml sebagai hasil pencatatan FPS saat simulasi berakhir dan myHealth.xml sebagai hasil pencatatan *hit points* saat simulasi berakhir