BAB 3
ANALISIS DAN PERANCANGAN SISTEM

3. 1 Analisis Sistem

Pada bagian analisis sistem ini, akan dilakukan berbagai macam analisis yang berhubungan dengan *game* yang akan dibangun, mulai dari analisis masalah, *storyline, gameplay,* analisis algoritma, analisis kebutuhan non-fungsional dan analisis kebutuhan fungsional.

3.1. 1 Analisis Masalah

Game Alien Hunter merupakan *game first person shooter* 3D (tiga dimensi), yang mempunyai misi utama untuk mengalahkan *alien* yang ada dibumi. Di dalam *game* tersebut terdapat saran, diantaranya yaitu dari segi karakter NPC (*Non Player Character*) musuh yang kurang bervariasi sehingga membuat *game* menjadi kurang menantang untuk dimainkan.

Game 3D FPS *The Tracker* adalah *game first person shooter* yang dibuat menggunakan *Unity*. *Game* ini mempunyai solusi dari saran yang ada di *game Alien Hunter,* yaitu dengan menambahkan variasi pergerakan karakter NPC (*Non Player Character*) pada musuh. Misi utama pemain di *game* ini adalah untuk menyelamatkan seorang teman yang ditemuinya diperjalanan serta mencari pimpinan perampok dan mengalahkan semua pasukan perampok yang berada di sebuah desa sunyi.

3.1.1. 1 Analisis *game* sejenis pada *Alien Hunter*

Pada bagian analisis *game* sejenis, akan dilakukan berbagai macam analisis yang berhubungan dengan *game* yang akan dibangun, mulai dari dari *analisis* terhadap *game* sejenis serta *analisis* berbagai kebutuhan baik fungsional maupun non-fungsional.

3.1.1. 2 Pengenalan *game Alien Hunter*

Game Alien Hunter merupakan *game* ber*genre First Person Shooter (FPS),* yang dibuat oleh Refi Meisadri. *Game* ini dibuat menggunakan *Unity* yang

mempunyai desain 3D (3 dimensi). Beberapa cuplikan permainan dari *game Alien Hunter* dapat dilihat pada gambar 3.1.



Gambar 3. 1 Menu utama *game Alien Hunter*

Gambar 3.1 adalah tampilan menu utama dari *game Alien Hunter*, dimana terdapat pilihan menu *start game* untuk memulai permainan, *How to play* untuk info *controller*, *credit* untuk info pengembang, dan *exit game* untuk keluar dari *game.*



Gambar 3. 2 Layar Permainan *game Alien Hunter*

3.1.1. 3 *Gameplay Alien Hunter*

Tugas utama pemain dalam memainkan *game* ini adalah untuk menyelesaikan dua misi yang diberikan. Misi pertama yaitu pemain harus bisa

masuk kedalam gudang penyimpanan reaktor yang berada di tengah kota serta mendapatkan reaktor nuklir yang tersembunyi didalamnya. Misi kedua, pemain harus mampu masuk kedalam markas *alien* dan masuk kedalam ruang kendalinya untuk mengaktifkan reaktor nuklir yang telah didapatkan pada misi sebelumnya untuk menghancurkan markas komando *alien* tersebut.

3.1.1. 4 Elemen *Game*

Elemen permainan yang terdapat dalam *game Alien Hunter* diantaranya adalah sebagai berikut :

1. Arena Permainan

   Pada permainan *game Alien Hunter* arena permainanya adalah disebuah gudang yang berada ditengah kota. Disini pemain harus bisa masuk kedalam gudang penyimpanan reaktor.

2. Pemain

   Pemain harus bisa mendapatkan reaktor nuklir yang tersembunyi di dalam sebuah gudang yang berada ditengah kota dan harus mampu masuk kedalam markas *alien* dan masuk kedalam ruang kendalinya untuk mengaktifkan reaktor nuklir yang telah didapatkan pada misi sebelumnya untuk menghancurkan markas komando *alien* tersebut.

3. Musuh

   Musuh dalam *game Alien Hunter* terdiri dari beberapa jenis diantaranya *spider robot, white alien* dan *red alien* yang akan mengejar *player* dan akan menyerang *player*.

3.1.1. 5 Objek yang terlibat dalam permainan *game Alien Hunter*

Beberapa objek yang terlibat dalam permainan *game Alien Hunter* adalah sebagai berikut :

Tabel 3. 1 Objek *game Alien Hunter*

No	Nama	Gambar	Penjelasan
1	Mesingun		Ini adalah senjata utama dalam game yang dikendalikan pemain melalui inputan dari *keyboard.*

2	*Rocket launcher*		Ini adalah senjata kedua dalam game yang dikendalikan pemain melalui inputan dari *keyboard.*
3	*Spider robot*		Ini adalah salah karakter musuh Musuh, yang akan meledakan diri jika pemain mendekati *robot spider.*
4	*White Alien*		White alien mempunyai daya serang damage 30 dan range shoot 15 dan hitspoint 100.
5	*Red Alien*		Red alien mempunyai daya serang damage 30 dan range shoot 20 dan hitspoint 100.
6	*Sentrygun*		*Sentrygun* mempunyai daya serang damage 5 dan range shoot 25 dan hitspoint 100

3.1.2 1 Analisis Perancangan *Game The Tracker*

Pada analisis ini, akan dilakukan analisis terhadap perancangan *game* yang akan dibangun. Tujuan dari analisis ini yaitu mendapatkan konsep permainan dari *game The Tracker*.

3.1.2 2 Pengenalan *Game The Tracker*

Game The Tracker merupakan *game* ber*genre first person shooter* (FPS) dengan sistem *single player*, *game The Tracker* yang akan dibangun menggunakan Unity 4.0 yang mempunyai desain 3D (tiga dimensi).

3.1.2 3 *Storyline The Tracker*

Game The Tracker bercerita tentang sebuah desa hilang yang berada jauh dari jantung kota. Masyarakat yang mengetahui desa tersebut menyebut sebagai desa sunyi, awalnya desa tersebut tentram dan damai, hingga pada akhirnya semua berubah ketika segerombolan perompok melakukan perampokan disebuah desa tersebut yang menyebabkan semua warga desa tersebut mati dibunuh. Desa

tersebut menjadi sebuah desa hilang yang dihuni oleh para perampok. Setelah beberapa bulan terjadinya perampokan tersebut, seorang polisi bernama Udie ditugaskan untuk menemukan dan mengalahkan perampok yang berada disebuah desa hilang. Untuk bisa menemukan desa sunyi dan mengalahkan perampok, seorang polisi harus melewati beberapa rintangan seperti hutan, disini seorang polisi dihadapkan dengan ancaman binatang liar seperti laba-laba, setelah melewati hutan seorang polisi harus menyelamatkan seorang teman yang ditemuinya diperjalanan dan menemukan sebuah desa sunyi untuk mengalahkan perampok dan pimpinan perampok.

3.1.2 4 *Gameplay The Tracker*

Tugas utama pemain (pengguna) dalam memainkan *game* ini yaitu harus melalui tiga *level*, dimana tiap *level* memiliki tingkat kesulitan yang berbeda baik dari segi NPC (*Non player character*) musuh maupun di dalam arena permainannya, semakin tinggi *level* maka tingkat kesulitan akan semakin sulit. Setiap *level* pemain harus menyelesaikan misi untuk melangkah ke *level* berikutnya. Musuh akan menghalangi pemain untuk menyelesaikan misi, akan tetapi, pemain dapat mengalahkan musuh dengan item-item yang telah disediakan berupa senjata api.

Pertama pemain akan memasuki *menu* utama yang berisi opsi-opsi seperti mulai, petunjuk, dan keluar. Dengan penjelasannya sebagai berikut.

1. *Menu* mulai berfungsi untuk memulai permainan.
2. *Menu* Petunjuk berfungsi untuk memberikan informasi tentang aturan-aturan yang diterapkan di dalam *game.*
3. *Menu* keluar untuk keluar dari permainan.

Di dalam *game* ini pemain dituntut untuk menyelesaikan misi yang ada dengan peralatan dan senjata yang telah diberikan. Didalam *game* juga tersedia menu *health bar s*ebagai informasi *health* (darah) *player, health bar* akan berkurang jika *player* terkena serangan musuh dan jika *health bar* habis, *player* akan mati dan otomatis misi akan gagal.

3.1.2 5 Elemen *game The Tracker*

Elemen permainan yang terdapat dalam permainan *game The Tracker* diantaranya adalah sebagai berikut :

1. Arena Permaianan
    1) *Level* satu

        Pemain berada di tengah hutan yang dikelilingi oleh binatang seperti Laba-laba. Pemain harus bisa menghindari atau mengalahkan binatang yang menyerang pemain, atau menghindari binatang yang ditemuinya. Pemain harus bisa sampai ke lokasi tujuan misi untuk lanjut ke *level* berikutnya.
    2) *Level* kedua

        Pemain harus masuk kearea markas musuh dan mengalahkan semua musuh berupa manusia dan tower penembak yang ada disekitar desa sunyi serta menyelamatkan seorang teman berupa manusia untuk bisa lanjut ke*level* berikutnya.
    3) *Level* ketiga

        Untuk menyelesaikan misi terakhir Pemain harus mencari dan mengalahkan pemimpin musuh yang berada di area desa sunyi. Disini pemain dihadapkan dengan ancaman pasukan perampok, binatang berupa laba-laba dan tower penembak.
2. Pemain

    Pemain bernama Udie merupakan seorang polisi yang ditugaskan untuk mengalahkan kelompok perampok. Untuk bisa mengalahkan perampok, pemain harus melewati beberapa rintangan seperti pemain harus bisa keluar dari hutan yang dihadapkan dengan ancaman binatang liar, pemain harus bisa memasuki area musuh disini pemain harus bisa mengalahkan musuh, dan yang terakhir pemain harus mengalahkan Pemimpin perampok yang berada di area desa sunyi.
3. Musuh

    Pada *game The Tracker* musuh terdiri dari tiga jenis yaitu hewan berupa laba-laba, tower penembak, dan manusia berupa perampok. Musuh

hewan akan keluar dari hutan untuk mengejar pemain, pemain dapat menghindari dan mengalahkan laba-laba dengan menembaknya menggunakan senjata yang telah disediakan, musuh berupa manusia akan keluar disekitar area desa sunyi.

3.1.2 6 Objek yang terlibat dalam permainan *Game The Tracker*

Karakter utama dalam *game* ini bernama Udie seorang polisi yang dirancang untuk melawan para pasukan perampok. Udie dilengkapi dengan persenjataan berupa dua senjata berupa laras panjang. Karena *game* ini merupakan *game* yang mempunyai *genre first person shooter* yang mempunyai sudut pandang orang pertama, maka karakter Udie seperti tubuh, kepala, kaki tidak diperlihatkan tetapi hanya memperlihatkan bagian tangan dan moncong senjatanya saja.

Tabel 3. 2 *Properties* karakter utama

Nama	Udie
Darah	100
Lives	3
Kemampuan	Berjalan, berlari, melompat, menembak
Senjata	ScarNew dan sniper

Beberapa Objek yang berada dalam permainan *game The Tracker* adalah sebagai berikut :

Tabel 3. 3 Objek *game The Tracker*

No	Nama	Gambar	Penjelasan
1	*ScarNew*		Ini adalah senjata utama dalam game yang dikendalikan pemain melalui inputan dari *keyboard.*
2	*Sniper*		Ini adalah senjata kedua dalam game yang dikendalikan pemain melalui inputan dari *keyboard.*
3	*Sentrygun*		Ini adalah tower senjataotomatis musuh mempunyai daya serang damage 5 dan range shoot 25.

4	Laba-laba hitam		Ini adalah karakter musuh hewan yang mempunyai damage 7-15 dan Hitspoint 40.
5	Laba-laba merah		karakter musuh hewan yang mempunyai damage 15-30 dan Hitspoint 80.
6	perampok merah		Ini adalah karakter musuh manusia yang mempunyai senjata pistol dengan damage 10, Hitspoint 80 dan kecepatan menyerang 3.
7	perampok hitam		Ini adalah karakter musuh manusia (Pimpinan Perampok) yang mempunyai senjata rocket dengan damage 30, Hitspoint 100 dan kecepatan menyerang 4.
8	Karakter teman		Ini adalah karakter teman berupa manusia yang harus diselamatkan.

3.1.2 7 *Story Board*

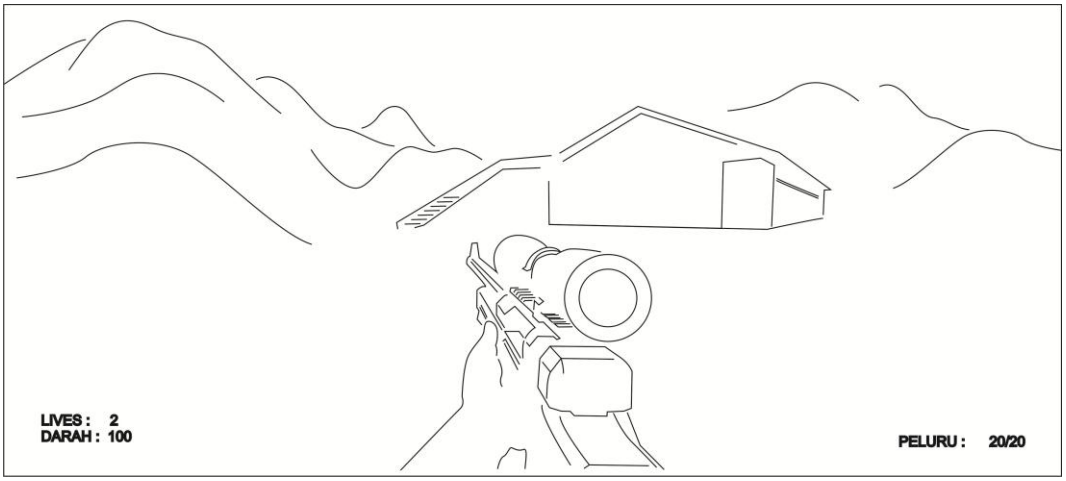
Storyboard adalah sketsa gambar yang disusun berurutan sesuai dengan naskah, dengan menggunakan *storyboard* dapat menyampaikan ide cerita kepada orang lain dengan lebih mudah karena dapat dengan *storyboard* dapat mengimajinasikan khayalan yang sesuai dengan gambar-gambar yang ada sehingga dapat menghasilkan persepsi yang sama pada ide cerita yang dibuat.

Berikut beberapa *storyboard* dari *game first person shooter* 3D *The Tracker*.



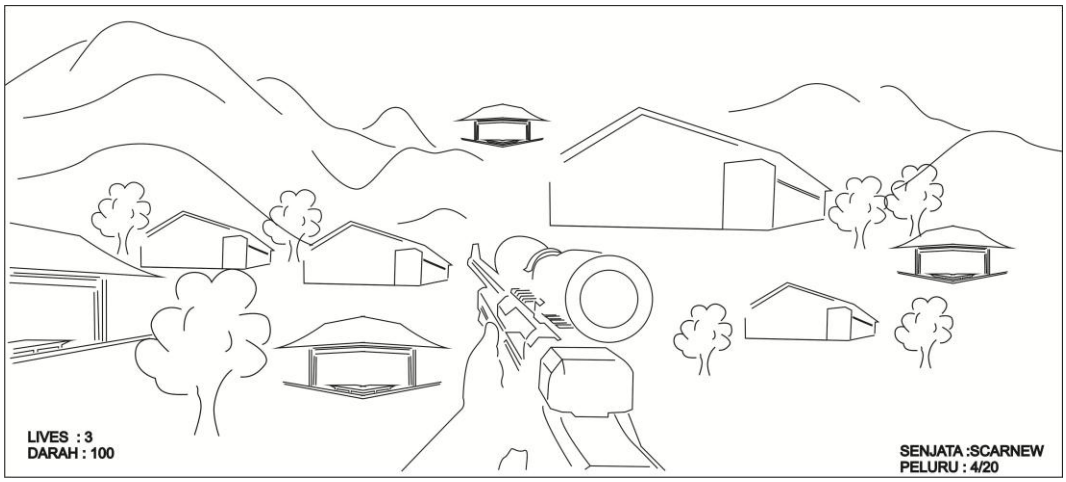
Gambar 3. 3 *Story Board level* 1

Gambar ini memperlihatkan kondisi tempat *player* memainkan *level* pertama permainan yang berada disebuah hutan. Pada *level* pertama ini pemain harus mencari jalan keluar untuk bisa melanjutkan ke *level* berikutnya.



Gambar 3. 4 *Story Board level* 2

Gambar ini memperlihatkan kondisi tempat *player* memainkan *level* kedua permainan, yang berada di sebuah desa sunyi.



Gambar 3. 5 *Story Board level* 3

Gambar ini memperlihatkan kondisi tempat *player* memainkan *level* ketiga permainan yang berada di area desa sunyi.

3.1.2.8 Analisis *Scoring*

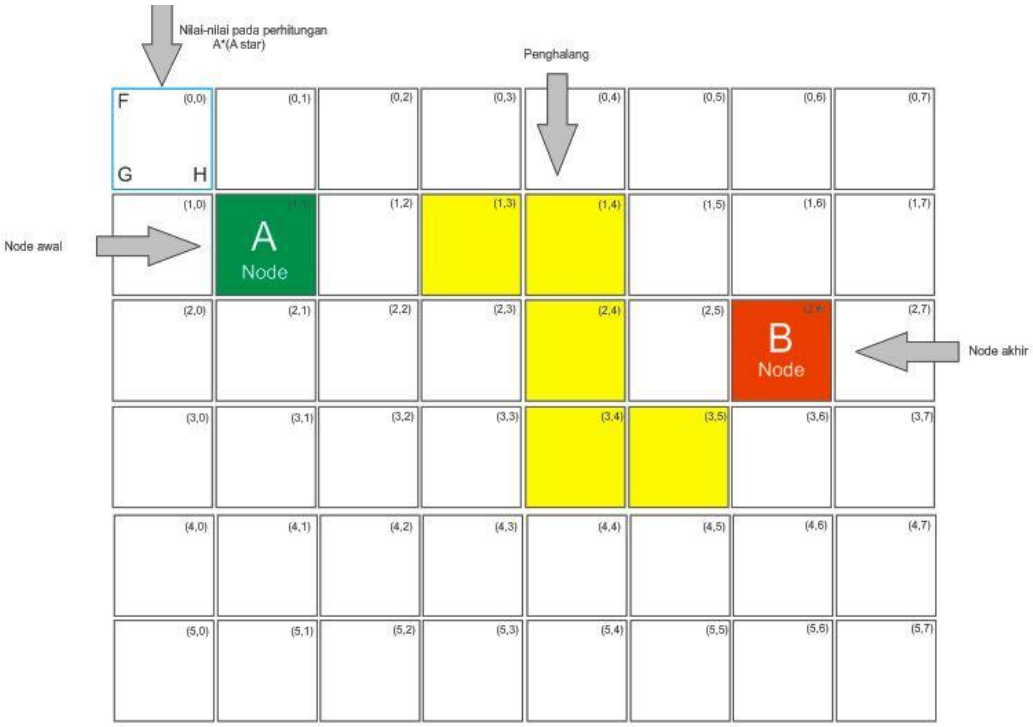
Pada *game the tracker* ini tidak memiliki *score*, satu-satunya penilaian yang ada adalah sukses atau tidaknya misi dalam setiap *level*.

3.1. 3 Analisis Algoritma

Pada Analisis Algoritma akan dijelaskan tentang algoritma apa saja yang dipergunakan dalam *game*.

3.1.3. 1 Analisis Algoritma A* (A *Star*)

Algoritma A* di dalam *game* akan diterapkan pada NPC (*Non player Character*) sebagai penentuan jalur terpendek atau *pathfinding* menuju *target.* Cara kerja algoritma A* di dalam *game the tracker* lebih jelasnya sebagai berikut.



Gambar 3. 6 Ilustrasi A*

Pada Gambar 3.6 terdapat beberapa kotak yang memiliki koordinat dari (0,1) sampai (5,7). Kotak atau *node* yang masing-masing telah memiliki koordinat merupakan *node* yang digunakan untuk melakukan pencarian jalur terpendek/*pathfinding*. Terdapat beberapa *node* yang memiliki warna berbeda, berikut penjelasan dari masing -masing *node* tersebut :

a. *Node* warna Hijau : Node awal yang berada pada node di koordinat (1,1)
b. *Node* warna kuning : Node penghalang yang berada pada *node* di koordinat (1,3), (1,3), (1,4) dan (2,4), (3,4), (3,5).
c. *Node* warna merah : *Node* tujuan yang berada pada *node* di koordinat (2,6).

Untuk nilai-nilai yang digunakan untuk perhitungan A *star*, terdapat 3 nilai yaitu :

a. Nilai F, nilai evaluasi.
b. Nilai G, nilai atau biaya yang dikeluarkan dari *node* awal/*node* A ke *node* yang ada di sekitarnya.
c. Nilai H, nilai heuristic yang digunakan untuk menghitung perkiraan biaya yang dikeluarkan dari *node* awal/*node* A menuju *node* tujuan/*node* B.

Perhitungan nilai F didapat dari persamaan F= G+H. Sedangkan untuk nilai G digunakan nilai 10 untuk gerakan *vertical* atau *horizontal*, angka 10 digunakan hanya untuk memudahkan perhitungan. Sedangkan untuk gerakan *diagonal* maka akan dikalikan dengan √2 yang berarti 10x√2 = 14,14. Untuk memudahkan perhitungan angka 14,14 dibulatkan menjadi 14.

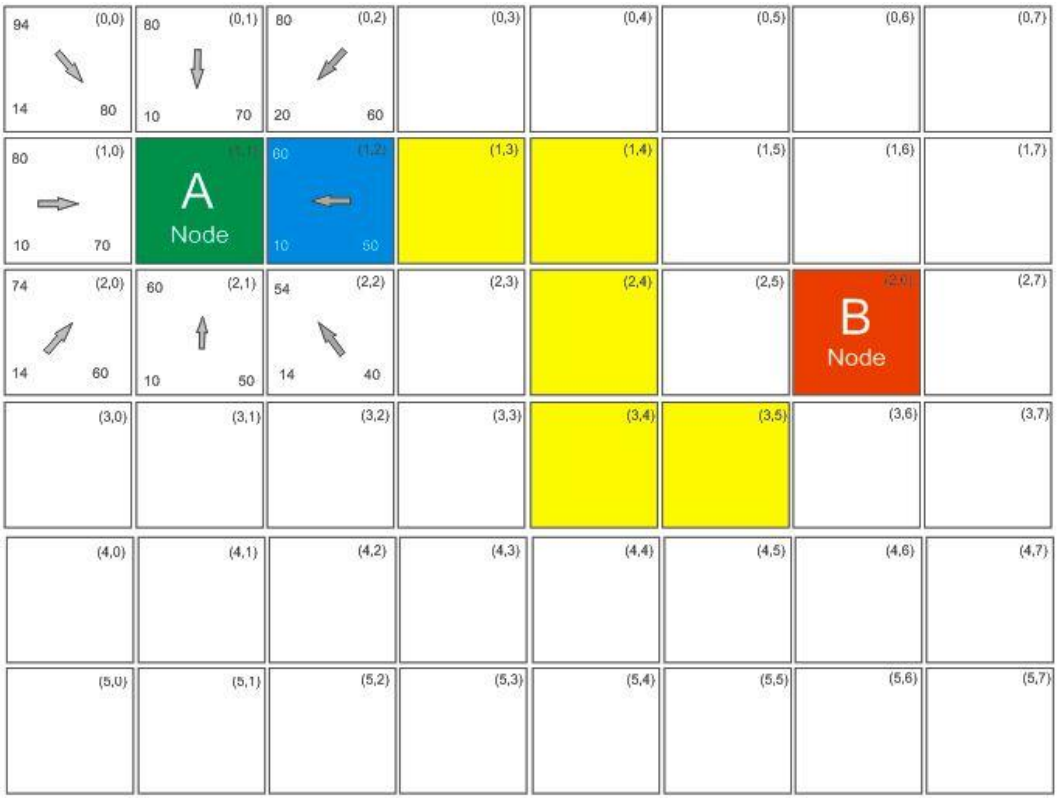
Untuk perhitungan nilai H digunakan fungsi *heuristic*, metode yang digunakan di dalam contoh ini adalah metode *Manhattan* dimana perhitungan jumlah *node* hanya yang bergerak secara *vertical* dan *horizontal* menuju tujuannya serta mengabaikan penghalang atau *collision*, yang kemudian nilainya dikalikan dengan 10. Atau dirumuskan dengan:

H = 10*(abs(*current*X-*target*X) + abs(*current*Y-*target*Y)) (3.1)

Langkah berikutnya ketika telah mendapatkan *node* awal adalah menentukan rute terpendek dengan cara memeriksa setiap *node* yang berdekatan dengan *node* A sampai *node* B. Di bawah ini merupakan penjelasan langkah-langkahnya:

1. Langkah pertama adalah memasukan *node* A kedalam *open list.*
2. Langkah kedua adalah memeriksa *node-node* yang berdekatan dengan *node* A, abaikan *node* dengan collision atau penghalang lalu tambahkan semua *node* tersebut kedalam *open list*. *Node* A sendiri nantinya akan menjadi "*parent*" dan digunakan untuk menelusuri jalan.
3. Langkah ketiga adalah menghapus *node* A dari *open list*, dan dimasukan ke dalam *closed list.*
4. Periksa semua *neighbor node* atau *node* yang berdekatan dengan *node* A apakah *walkable* atau tidak, abaikan *node* dengan *collision* atau penghalang lalu tambahkan semua *node* tersebut ke dalam *open list.* Dan untuk setiap *node* yang baru ditambahkan, simpan *node* A sebagai "*parent*" yang nantinya akan digunakan untuk menelusuri jalan.

Berikut ini perhitungan pada langkah pertama yang berada pada node dengan koordinat (1,1).



Gambar 3. 7 Ilustrasi langkah Pertama

Pada langkah pertama *node* (1,1) dimasukkan ke dalam *open list,* dan selanjutnya memeriksa *neighbor node* atau *node* yang berdekatan dengan *node* (1,1) yaitu *node* {(0,0), (0,1), (0,2), (1,0), (1,2), (2,0), (2,1), (2,2)} dan dimasukkan ke dalam *open list.* Hapus *node* (1,1) dari *open list* dan masukkan ke dalam *closed list.* Ambil *node* dengan fungsi heuristik yang memberikan biaya perkiraan yang mendekati biaya sebenarnya. Dalam contoh kasus ini menggunakan metode manhattan *city block distance* nilai F *node* (1,2) memiliki nilai F paling rendah setelah node (2,2) dan diberikan tanda dengan kotak berwarna biru dengan nilai F=60.

Langkah pertama menghasilkan *node-node* di dalam *open list* dan *closed list* dijelaskan dalam tabel 3.4.

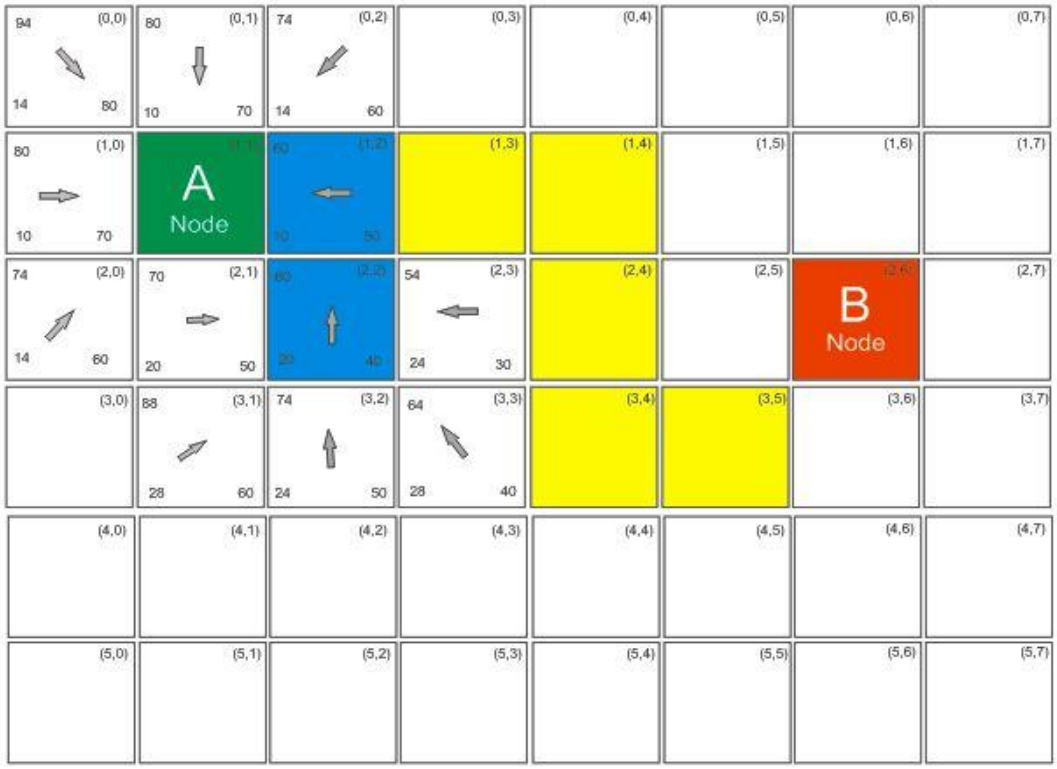
Tabel 3. 4 Perhitungan langkah pertama

Open List	(0,0), (0,1), (0,2), (1,0), (1,2), (2,0), (2,1), (2,2)
Parent	(1,1)
Closed list	(1,1)
Current node	(1,2)

Proses yang dijelaskan di atas, dilakukan berulang kali atau *looping*, dimulai dari *open list* dan menemukan *node* dengan nilai F paling rendah. Langkahnya adalah sebagai berikut :

1. Hapus *current node* tersebut dari *open list* dan masukkan ke dalam *closed list.*
2. Periksa semua *node* yang berdekatan dengan *current node*, abaikan *node* penghalang. Jika tidak ada pada *open list*, tambahkan ke dalam *open list.* Dan tambahkan "*parent*" untuk *node* tersebut.
3. Jika *node* sudah ada pada *open list*, periksa apakah ini jalan yang lebih baik dari *node* awal, dengan kata lain periksa apakah nilai G-nya lebih rendah apabila menggunakan *node* ini, jika lebih rendah maka hitung ulang nilai F dan G serta merubah arah *pointer*, jika tidak maka jangan lakukan apa-apa.

Berikut ini perhitungan pada langkah kedua yang berada pada node dengan koordinat (2,2).



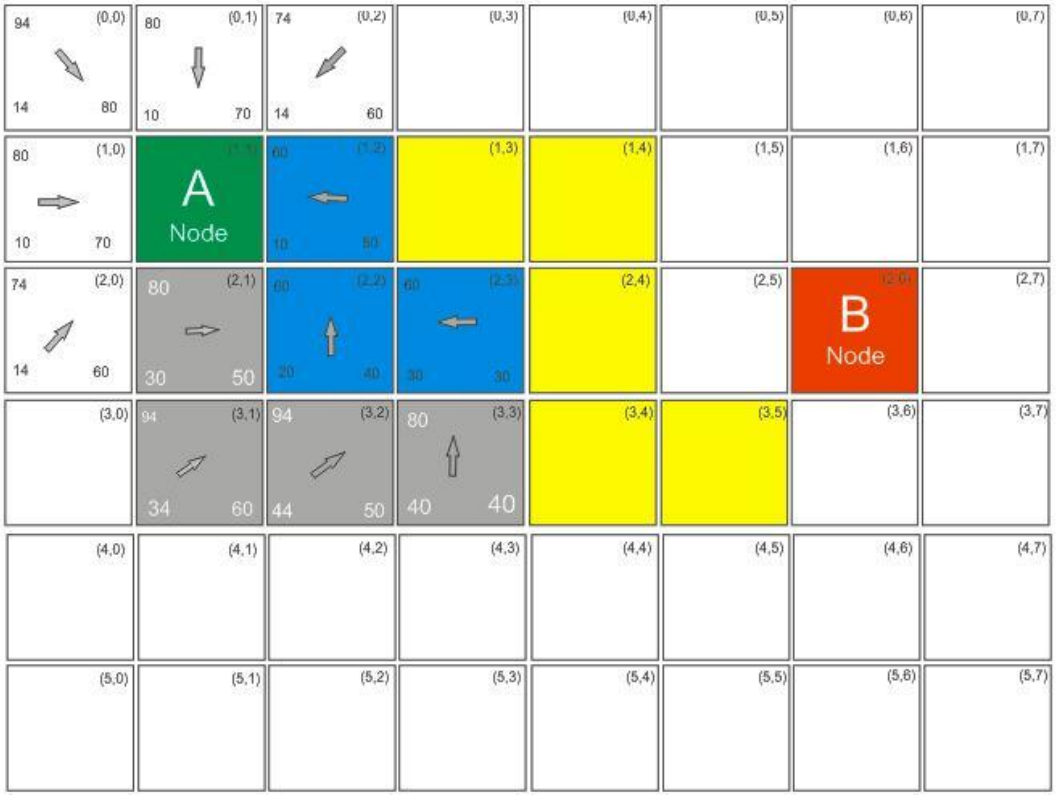
Gambar 3. 8 Ilustrasi A* langkah Kedua

Berikut ini penjelasan melalui tabel untuk pencarian jalan terpendek pada langkah kedua.

Tabel 3. 5 Perhitungan langkah kedua

Open List	(2,1), (2,3), (3,1), (3,2), (3,3)
Parent	(1,2)
Closed list	(1,1),(1,2),(2,2)
Current node	(2,3)

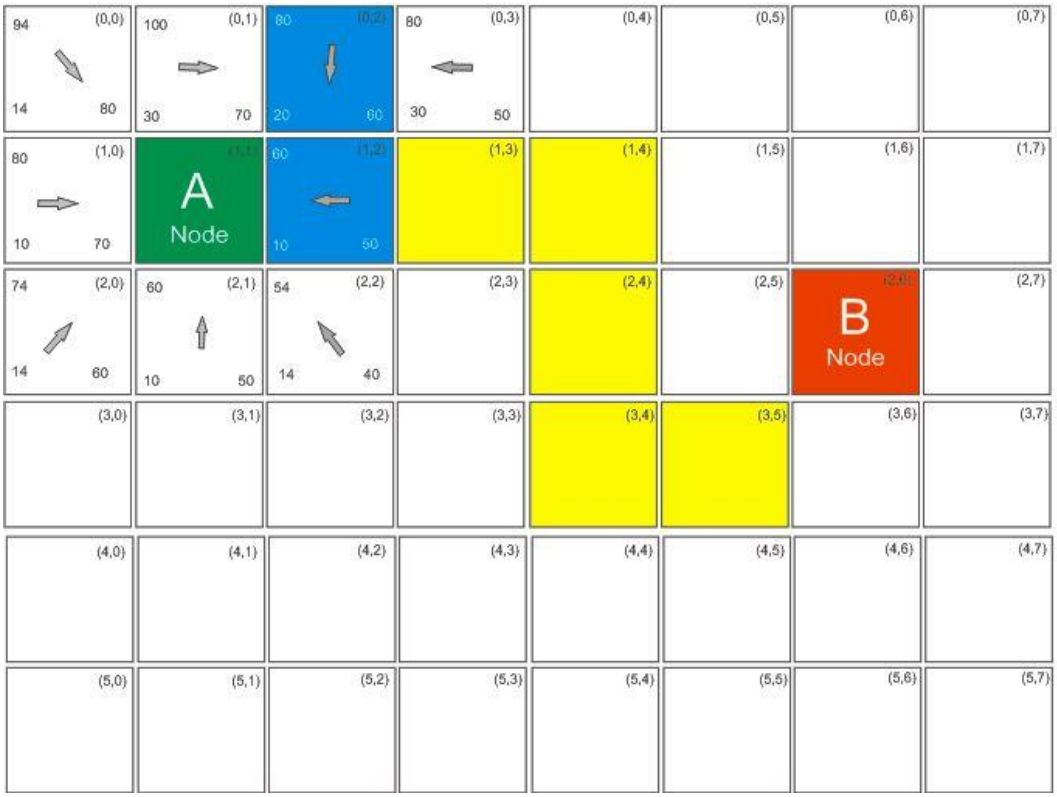
Berikut ini perhitungan pada langkah ketiga yang berada pada *node* dengan koordinat (2,3).



Gambar 3. 9 Ilustrasi A* langkah Ketiga

Langkah keempat di dapat dengan nilai F lebih besar dari biaya *node* (2.3) yaitu 60. Oleh karena itu *parent* dari (2.3) dan *parent* sebelumnya harus dirubah. Dan hitung kembali nilai yang berada di *open list* sebelumnya.

Berikut ini perhitungan pada langkah kelima yang berada pada *node* dengan koordinat (0,2).



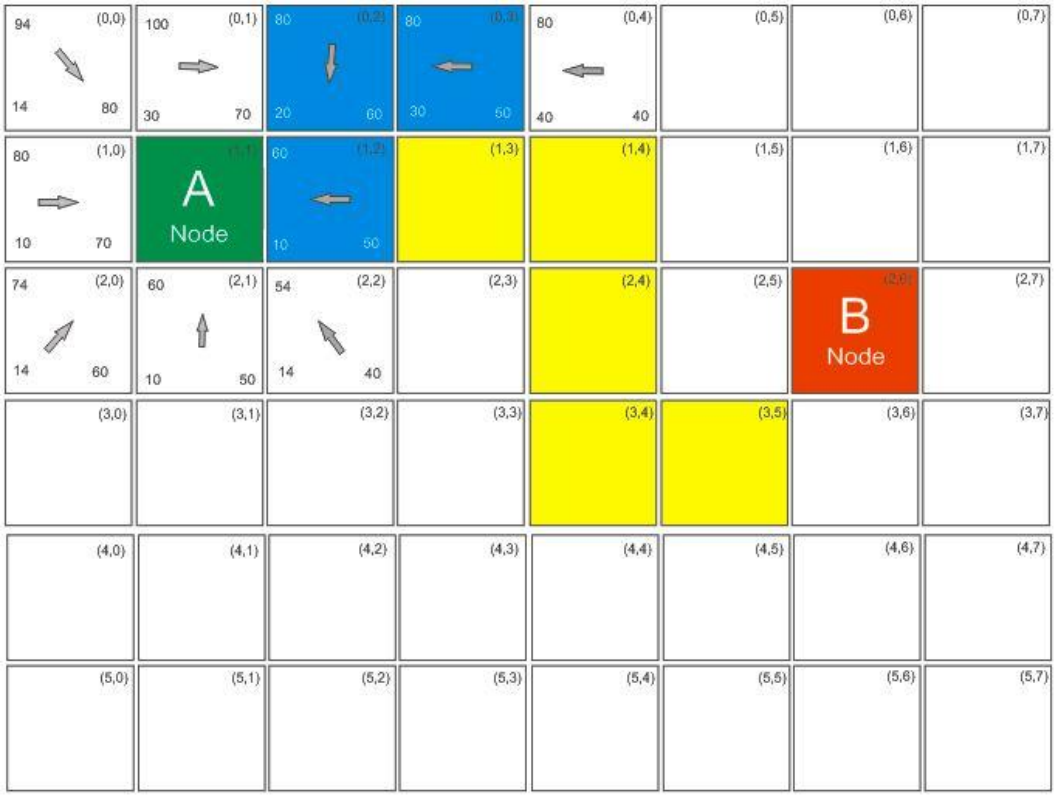
Gambar 3. 10 Ilustrasi A* langkah kelima

Berikut ini penjelasan melalui tabel untuk pencarian jalan terpendek pada langkah kelima.

Tabel 3. 6 Perhitungan langkah kelima

Open List	(0,1), (0,3)
Parent	(1,2)
Closed list	(1,1), (1,2), (0,2)
Current node	(0,3)

Berikut ini perhitungan pada langkah keenam yang berada pada *node* dengan koordinat (0,3).



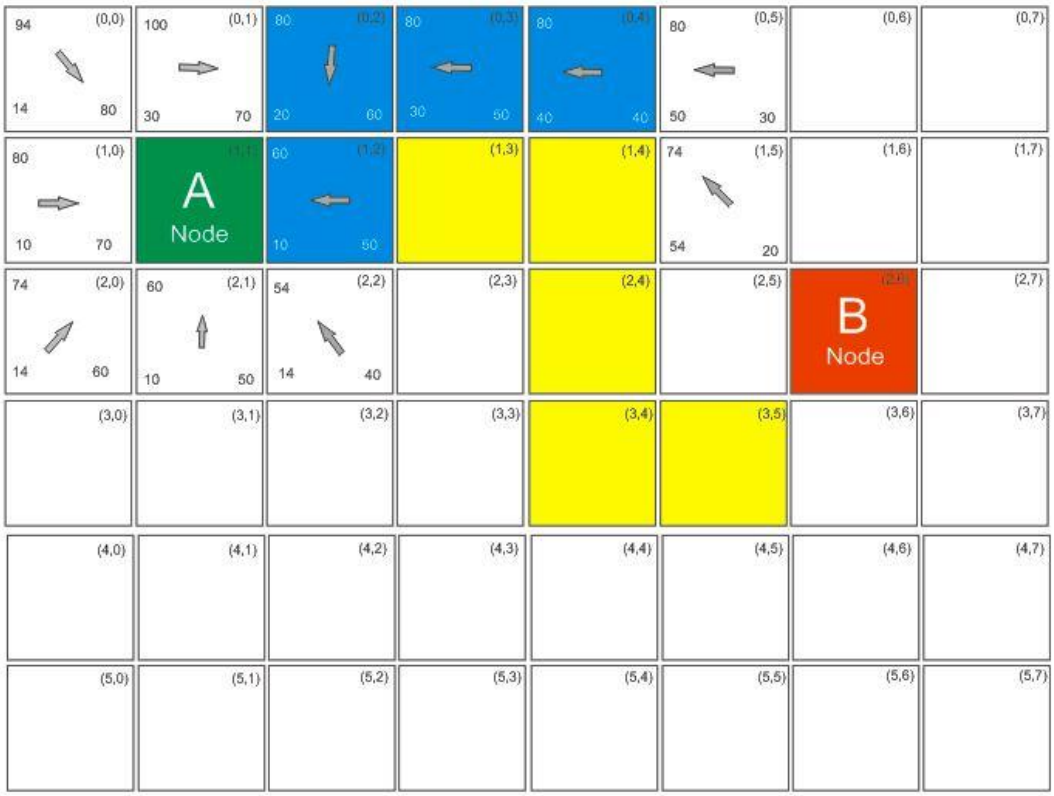
Gambar 3. 11 Ilustrasi A* langkah keenam

Berikut ini penjelasan melalui tabel untuk pencarian jalan terpendek pada langkah keenam.

Tabel 3. 7 Perhitungan Iterasi keenam

Open List	(0,4)
Parent	(0,2)
Closed list	(1,1), (1,2), (0,2), (0,3)
Current node	(0,4)

Berikut ini perhitungan pada langkah ketujuh yang berada pada *node* dengan koordinat (0,4).



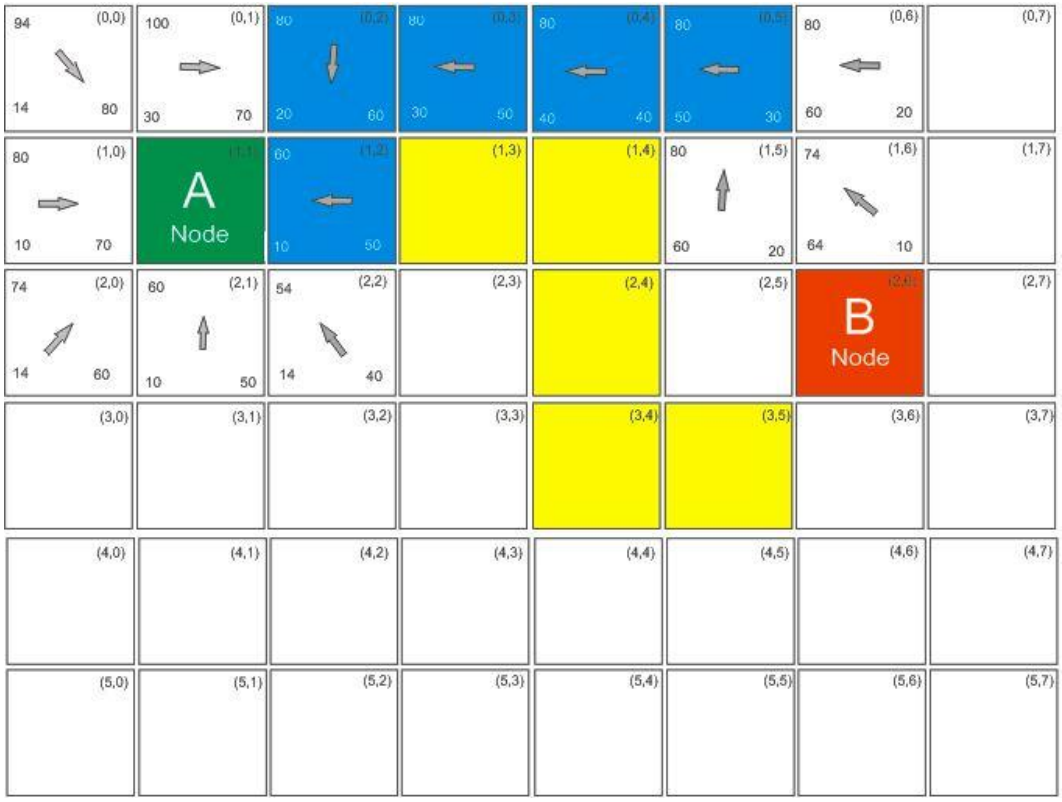
Gambar 3. 12 Ilustrasi A* langkah ketujuh

Berikut ini penjelasan melalui tabel untuk pencarian jalan terpendek pada langkah ketujuh.

Tabel 3. 8 Perhitungan langkah ketujuh

Open List	(0,5), (1,5)
Parent	(0,4)
Closed list	(1,1), (1,2), (0,2), (0,3), (0,4)
Current node	(0,5)

Berikut ini perhitungan pada langkah kedelapan yang berada pada *node* dengan koordinat (0,5).



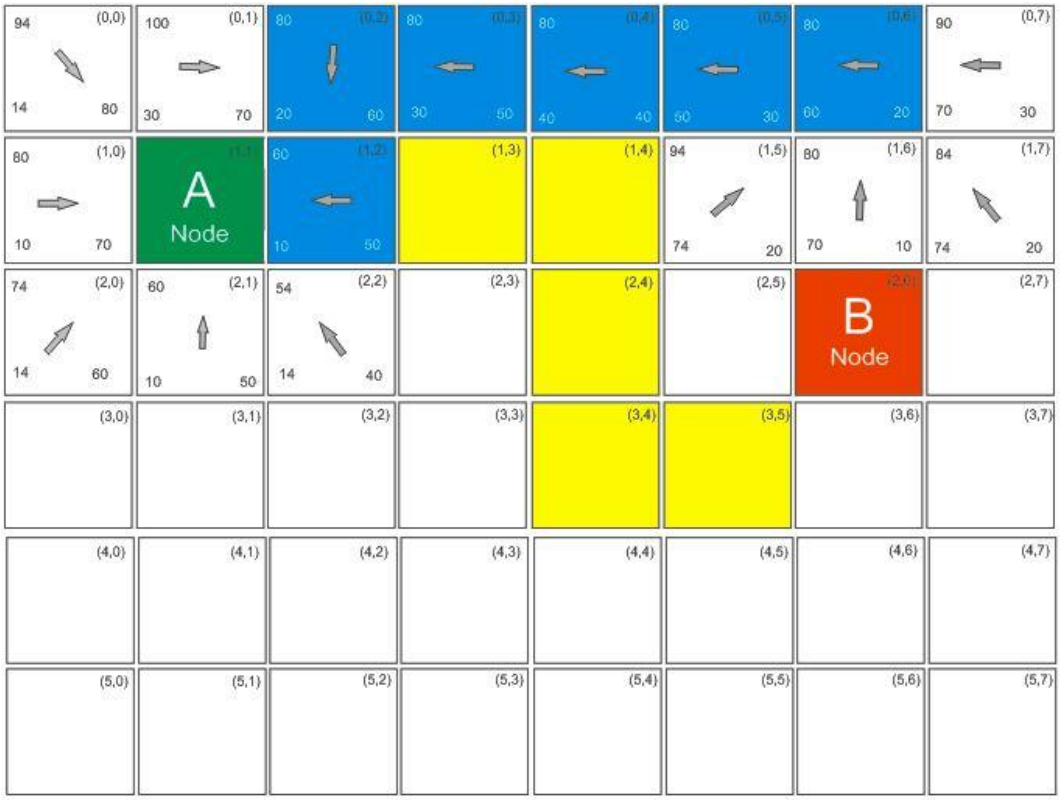
Gambar 3. 13 Ilustrasi A* langkah kedelapan

Berikut ini penjelasan melalui tabel untuk pencarian jalan terpendek pada langkah kedelapan.

Tabel 3. 9 Perhitungan langkah kedelapan

Open List	(0,6), (1,5), (1,6)
Parent	(0.5)
Closed list	(1,1), (1,2), (0,2), (0,3), (0,4), (0,5)
Current node	(0,6)

Berikut ini perhitungan pada langkah kesembilan yang berada pada *node* dengan koordinat (0,6).



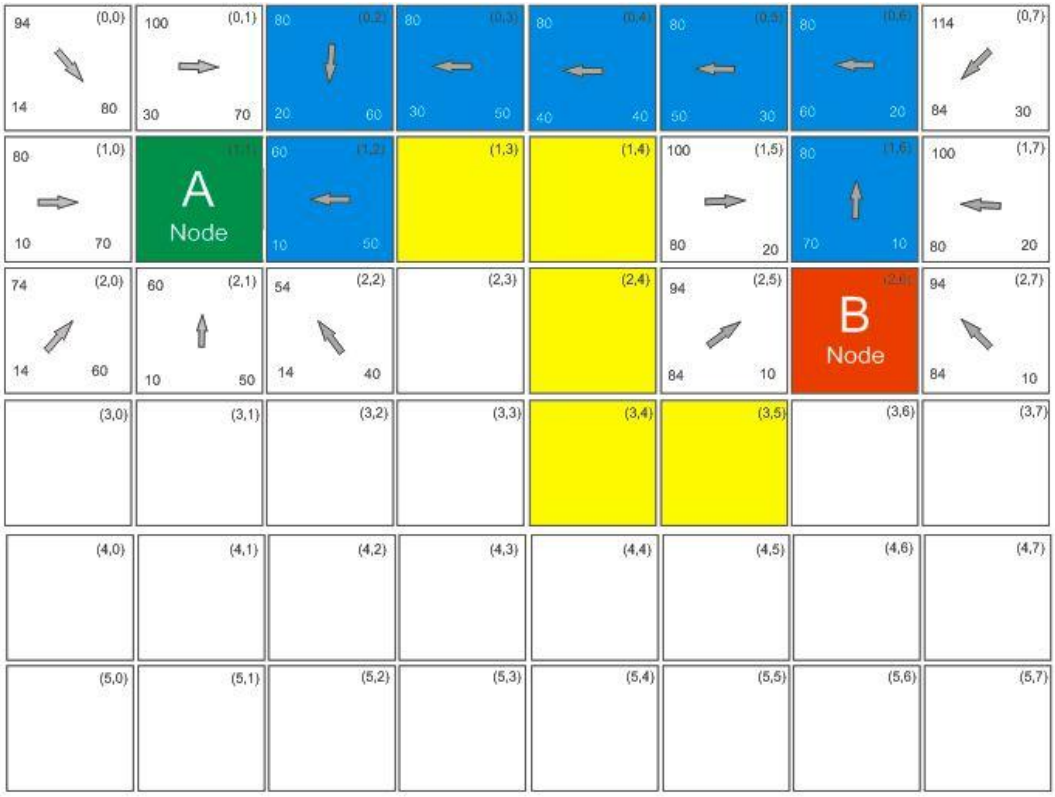
Gambar 3. 14 Ilustrasi A* langkah kesembilan

Berikut ini penjelasan melalui tabel untuk pencarian jalan terpendek pada langkah kesembilan.

Tabel 3. 10 Perhitungan langkah kesembilan

Open List	(0,7), (1,5), (1,6), (1,7)
Parent	(0,6)
Closed list	(1,1), (1,2), (0,2), (0,3), (0,4), (0,5), (0,6)
Current node	(1,6)

Berikut ini perhitungan pada langkah kesepuluh yang berada pada *node* dengan koordinat (1,6).



Gambar 3. 15 Ilustrasi A* langkah kesepuluh

Selanjutnya (2,6) dengan biaya terkecil yaitu 70 terpilih sebagai *best node* karena *best node* sama dengan *goal*, berarti solusi sudah ditemukan. Rute dan total biaya bisa ditelusuri balik dari (2,6) menuju (1,1) atau dari B menuju A karena tiap node hanya memiliki satu *parent* dan setiap node memiliki biaya sebenarnya penulusan balik menghasilkan rute (2,6), (1,6), (0,6), (0,5), (0,4), (0,3), (0,2), (1,2), (1,1). Dengan total jarak sama dengan 70. Rute ini merupakan rute terpendek. Jadi algoritma A* adalah complite dan optimal.

Tabel 3. 11 Perhitungan langkah kesepuluh

Open List	(0,7), (1,5), (1,7), (2,5), (2,7)
Parent	(1,6)
Closed list	(1,1), (1,2), (0,2), (0,3), (0,4), (0,5), (0,6), (1,6),(2,6)
Current node	(2,6)

3.1.3. 2 Analisis *Fuzzy Logic*

Pada *game* ini *fuzzy logic* digunakan untuk menentukan variasi prilaku yang dilakukan oleh NPC. Dengan adanya *fuzzy logic* masing-masing NPC dapat merubah perilaku berbasis perubahan variabel masukan. Metode *fuzzy* yang digunakan adalah metode sugeno karena metode ini menghasilkan keluaran yang berupa konstan tegas, sehingga dapat mewakili nilai perilaku yang telah dirancang.

Untuk menghasilkan prilaku pada NPC ada dua variabel yang digunakan, yaitu Jarak (dekat, sedang dan jauh) dan Kekuatan (lemah dan kuat). Dengan menggunakan dua variabel diharapkan NPC mampu menentukan prilaku yang akan dilakukan.

Tahapan-tahapan dari analisis *fuzzy logic* adalah sebagai berikut :

a. Variabel Fuzzy

   Terdapat tiga variabel yang digunakan dalam fungsi fuzzy, yaitu sebagai berikut :

   1. Variabel Jarak sebagai variabel *input.*
   2. Variabel Kekuatan sebagai variabel *input.*
   3. Variabel Keputusan sebagai variabel *output.*

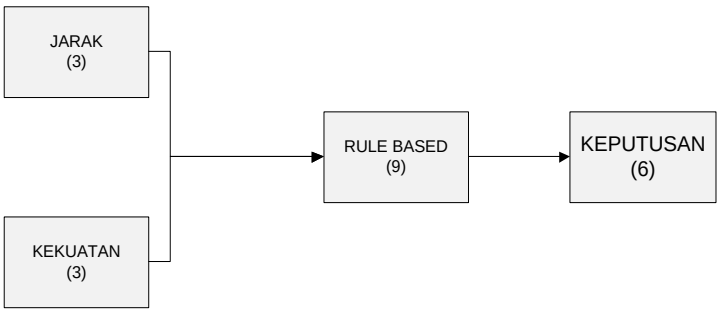
b. Nilai linguistik

   Dari tiga variabel yang digunakan, memiliki nilai linguistik sebagai berikut :

   1. Jarak :Dekat, Sedang, Jauh
   2. Kekuatan :Lemah, sedang, Kuat
   3. Keputusan :Menyerang brutal, Menyerang, Bertahan, Menghindar, Menambah darah, Melarikan Diri

c. Fuzzyfikasi

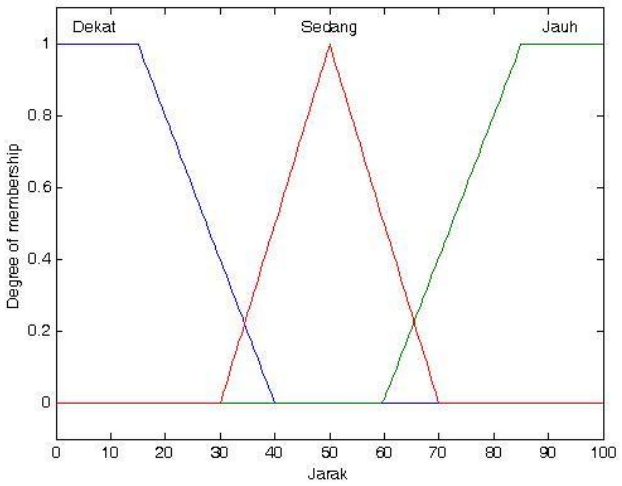
   Fuzzyfikasi merupakan proses memetakan nilai *crisp* (numerik) ke dalam himpunan *fuzzy* dan menentukan derajat keanggotaannya. Secara garis besar pemetaan nilai *crisp* ke dalam himpunan *fuzzy* dijelaskan dengan gambar sebagai berikut :



Gambar 3. 16 *Fuzzyfikasi*

Berdasarkan *fuzzy interface system* di atas maka pemetaan *input-output* adalah sebagai berikut:

1. Variabel Jarak, terbagi menjadi 3 input yaitu : Dekat, Sedang dan Jauh. *Range* nilai untuk variabel Jarak antara 0-100 akan dijelaskan sebagai berikut.
   1) Dekat : 0 - 40
   2) Sedang : 30 - 70
   3) Jauh : 60 - 100



Gambar 3. 17 Grafik *input* jarak

Pada gambar 3.17 menunjukan sebuah grafik jarak yang mempunyai range nilai dari 0-100, setiap nilai linguistik dari variabel jarak seperti dekat, sedang, dan jauh mempunyai nilai *fuzzyfikasi* yang berbeda-beda.

1) Variabel linguistik Dekat yang memiliki *range* nilai antara 0 – 40 nilai fuzzyfikasinya dihitung berdasarkan fungsi trapesium, berikut persamaannya :

$$\mu Dekat[x]\begin{cases} 1; x \leq 15 \\ \frac{40-x}{25}; 15 \leq x \leq 40 \\ 0; x \geq 40 \end{cases} \quad (3.2)$$

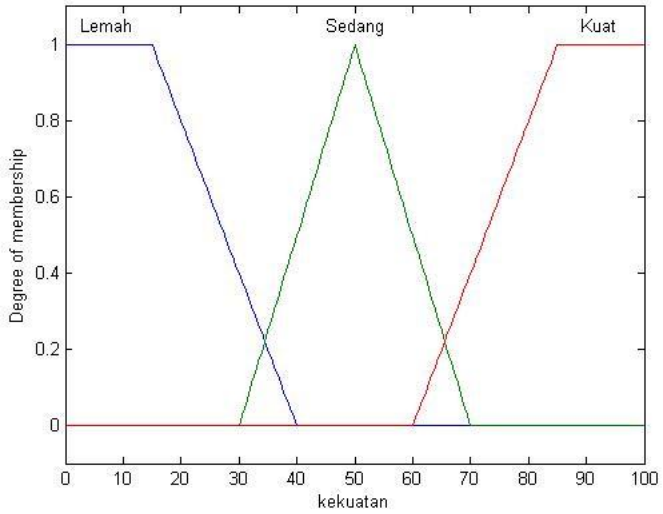
2) Variabel linguistik Sedang yang memiliki *range* nilai antara 30 – 70 nilai *fuzzyfikasi*nya dihitung berdasarkan fungsi segitiga, berikut persamaannya :

$$\mu sedang[x]\begin{cases} 0; x \leq 30 \; atau \; x \geq 70 \\ \frac{x-30}{20}; 30 \leq x \leq 50 \\ \frac{70-x}{20}; 50 \leq x \leq 70 \end{cases} \quad (3.3)$$

3) Variabel linguistik Jauh yang memiliki *range* nilai antara 60 – 100 nilai *fuzzyfikasi*nya dihitung berdasarkan fungsi trapesium, berikut persamaannya :

$$\mu Jauh[x]\begin{cases} 0; x \leq 60 \\ \frac{x-60}{25}; 60 \leq x \leq 85 \\ 1; x \geq 85 \end{cases} \quad (3.4)$$

2. Variabel Kekuatan, terbagi menjadi 3 input yaitu : Lemah, Sedang dan Kuat. *Range* nilai variabel Kekuatan diantara 0 – 100 akan dijelaskan sebagai berikut :
   1) Lemah : 0 – 40
   2) Sedang : 30-70
   3) Kuat : 60 -100



Gambar 3. 18 Grafik *input* kekuatan

Pada gambar 3.18 menunjukan sebuah grafik kekuatan yang mempunyai range nilai dari 0-100, setiap nilai linguistik dari variabel kekuatan seperti lemah dan kuat mempunyai nilai *fuzzyfikasi* yang berbeda-beda.

1) Variabel linguistik Lemah yang memiliki *range* nilai antara 0 – 80 nilai *fuzzyfikasi*nya dihitung berdasarkan fungsi trapesium, berikut persamaannya :

$$\mu Lemah[x]\begin{cases} 1; x \leq 15 \\ \frac{40-x}{25}; 15 \leq x \leq 40 \\ 0; x \geq 40 \end{cases} \quad (3.5)$$

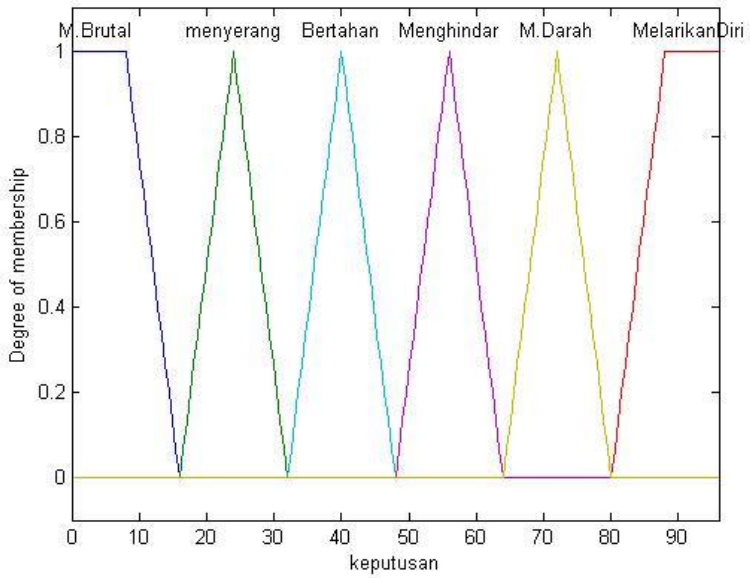
2) Variabel linguistik Sedang yang memiliki *range* nilai antara 30 – 70 nilai *fuzzyfikasi*nya dihitung berdasarkan fungsi segitiga, berikut persamaannya :

$$\mu sedang[x]\begin{cases} 0; x \leq 30 \; atau \; x \geq 70 \\ \frac{x-30}{20}; 30 \leq x \leq 50 \\ \frac{70-x}{20}; 50 \leq x \leq 70 \end{cases} \quad (3.6)$$

3) Variabel linguistik Kuat yang memiliki *range* nilai antara 60 – 100 nilai *fuzzyfikasi*nya dihitung berdasarkan fungsi trapesium, berikut persamaannya :

$$\mu Kuat[x]\begin{cases} 0; x \leq 60 \\ \frac{x-60}{25}; 60 \leq x \leq 85 \\ 1; x \geq 85 \end{cases} \quad (3.7)$$

3. Variabel Keputusan, yaitu Menyerang Brutal, Menyerang, Bertahan, Menghindar, Mencari darah dan Melarikan Diri. *Range* nilai variabel keputusan diantara 0-96, akan dijelaskan sebagai berikut:
   1) Menyerang Brutal : 0-16
   2) Menyerang : 16-32
   3) Bertahan : 32-48
   4) Menghindar : 48-64
   5) Menambah darah : 64-80
   6) Melarikan diri : 80-96



Gambar 3. 19 Grafik Keputusan

Pada gambar 3.19 menunjukan sebuah grafik keputusan yang mempunyai range nilai dari 0-96, setiap nilai linguistik dari variabel keputusan seperti Menyerang Brutal, Menyerang, Bertahan, Menghindar, Menambah darah dan Melarikan diri memiliki nilai fuzzifikasi yang berbeda-beda.

1) Variabel linguistik Menyerang brutal yang memiliki *range* nilai antara 0 – 16 nilai *fuzzyfikasi*nya dihitung berdasarkan fungsi trapesium, berikut persamaannya :

$$\mu Menyerang\ brutal[z]\begin{cases}1; z \leq 8\\ \frac{16-z}{8}; 8 \leq z \leq 16\\ 0; z \geq 16\end{cases} \quad (3.8)$$

2) Variabel linguistik menyerang yang memiliki *range* nilai antara 16 – 32 nilai *fuzzyfikasi*nya dihitung berdasarkan fungsi segitiga, berikut persamaannya :

$$\mu Menyerang[z]\begin{cases}0; z \leq 16\ atau\ z \geq 32\\ \frac{z-16}{8}; 16 \leq z \leq 24\\ \frac{32-z}{8}; 24 \leq z \leq 32\end{cases} \quad (3.9)$$

3) Variabel linguistik Bertahan yang memiliki *range* nilai antara 32 – 48 nilai *fuzzyfikasi*nya dihitung berdasarkan fungsi segitiga, berikut persamaannya :

$$\mu Berahan[z]\begin{cases}0; z \leq 32\ atau\ z \geq 48\\ \frac{z-32}{8}; 32 \leq z \leq 40\\ \frac{48-z}{8}; 40 \leq z \leq 48\end{cases} \quad (4.0)$$

4) Variabel linguistik Menghindar yang memiliki *range* nilai antara 48 – 64 nilai *fuzzyfikasi*nya dihitung berdasarkan fungsi segitiga, berikut persamaannya :

$$\mu Menghindar[z]\begin{cases}0; z \leq 48\ atau\ z \geq 64\\ \frac{z-48}{8}; 48 \leq z \leq 56\\ \frac{64-z}{8}; 56 \leq z \leq 64\end{cases} \quad (4.1)$$

5) Variabel linguistik Menambah darah yang memiliki *range* nilai antara 64 – 80 nilai *fuzzyfikasi*nya dihitung berdasarkan fungsi segitiga, berikut persamaannya :

$$\mu Mencari\ darah[z]\begin{cases}0; z \leq 64\ atau\ z \geq 80\\ \frac{z-64}{8}; 64 \leq z \leq 72\\ \frac{80-z}{8}; 72 \leq z \leq 80\end{cases} \quad (4.2)$$

6) Variabel linguistik Melarikan diri yang memiliki *range* nilai antara 80 – 96 nilai *fuzzyfikasi*nya dihitung berdasarkan fungsi trapesium, berikut persamaannya:

$$\mu Melarikan\ diri[z] \begin{cases} 0; z \leq 80 \\ \frac{z-80}{8}; 80 \leq z \leq 88 \\ 1; z \geq 88 \end{cases} \quad (4.3)$$

4. Kaidah *fuzzy* atau *fuzzy's rules*

Kaidah *fuzzy (rules)* atau aturan-aturan yang diterapkan dalam penentuan perilaku NPC berjumlah enam *rules* yaitu :

1) *If* (Jarak *is* Dekat) *and* (Kekuatan is Lemah) *then* (Perilaku NPC *is* Melarikan diri).
2) *If* (Jarak *is* Dekat) *and* (Kekuatan *is* Sedang) *then* (Perilaku NPC *is* Menghindar).
3) *If* (Jarak *is* Dekat) *and* (Kekuatan *is* Kuat) *then* (Perilaku NPC *is* menyerang Brutal).
4) *If* (Jarak *is* Sedang) *and* (Kekuatan *is* Lemah) *then* (Perilaku NPC *is* Melarikan diri).
5) *If* (Jarak *is* Sedang) *and* (Kekuatan *is* Sedang) *then* (Perilaku NPC *is* Menambah darah).
6) *If* (Jarak *is* Sedang) *and* (Kekuatan *is* Kuat) *then* (Perilaku NPC *is* Menyerang).
7) *If* (Jarak *is* Jauh) *and* (Kekuatan *is* Lemah) *then* (Perilaku NPC *is* Melarikan diri).
8) *If* (Jarak *is* Jauh) *and* (Kekuatan *is* Sedang) *then* (Perilaku NPC *is* Menambah darah).
9) *If* (Jarak *is* Jauh) *and* (Kekuatan *is* Kuat) *then* (Perilaku NPC *is* Bertahan).

5. Agregasi dan *Defuzzyfikasi*

Agregasi untuk menentukan keluaran semua *rules* yang dikombinasikan menjadi sebuah *fuzzy set* tunggal. *Defuzzyfikasi* merupakan proses pemetaan besaran dari himpunan *fuzzy set* yang dihasilkan ke dalam

bentuk nilai *crisp*. *Defuzzifikasi* pada metode Sugeno adalah dengan menghitung *center of single-ton* atau titik pusat nilai *crisp* dengan metode rata-rata (*average*).

$$z^* = \frac{\sum \mu i.zi}{\sum \mu i} \qquad (4.4)$$

1) z* = nilai *crisp*
2) μi =derajat keanggotaan masing-masing variable
3) *zi* = nilai domain pada variabel linguistik z.

6. Contoh soal *fuzzy logic*

Misal, apabila jarak memiliki nilai [35] dan Kekuatan memiliki nilai [45], maka tahapan-tahapan atau proses agregasi yang menentukan keluaran sesuai dengan *rules* yang sudah dibuat akan dijelaskan seperti berikut :

A. *Fuzzifikasi*

Perhitungan *Fuzzyfikasi* untuk Jarak dari setiap variabel lingustik (Dekat, Sedang dan Jauh) apabila nilai jarak [35] adalah sebagai berikut :

$$\mu Dekat[35] \quad = \frac{40-35}{25} = 0{,}2$$

$$\mu Sedang[35] = \frac{35-30}{20} = 0{,}25$$

$$\mu Jauh[35] \quad = 35 \leq 60 = 0$$

Dari hasil perhitungan *fuzzifikasi* menggunakan fungsi trapezium dan segitiga untuk mencari nilai derajat keanggotaan, maka didapatkan nilai derajat keanggotaan dari masing-masing variabel lingustik yaitu :

1. Fungsi Keanggotaan Dekat = 0,2
2. Fungsi Keanggotaan Sedang = 0,25
3. Fungsi Keanggotaan Jauh = 0

Perhitungan *Fuzzyfikasi* untuk Kekuatan dari setiap variabel lingustik (Lemah, sedang dan Kuat) apabila nilai Kekuatan [45] adalah sebagai berikut :

$$\mu Lemah[45] = \ 45 \geq 40 = 0$$

$$\mu Sedang[45] = \frac{45 - 30}{20} = 0{,}75$$

$$\mu Kuat[45] \quad = 45 \leq 60 = 0$$

Dari hasil perhitungan *fuzzifikasi* menggunakan fungsi trapezium dan segitiga untuk mencari nilai derajat keanggotaan, maka didapatkan nilai derajat keanggotaan dari masing-masing variabel lingustik yaitu :

1. Fungsi Keanggotaan Lemah = 0
2. Fungsi Keanggotaan Sedang = 0,75
3. Fungsi Keanggotaan Kuat = 0

B. Implikasi

Setelah nilai derajat keanggotaan didapatkan, maka nilai-nilai derajat tersebut harus dibandingkan sesuai dengan *Rule* yang ada, tujuannya untuk mencari nilai minimum untuk setiap *rule* yang dihitung.

1. *If* (Jarak *is* Dekat) *and* (Kekuatan is Lemah) *then* (Perilaku NPC *is* Melarikan diri)
   → Jarak (Dekat) = 0.2 dan
   Kekuatan (Lemah) =0
   Min (0.6 , 0) = 0
2. *If* (Jarak *is* Dekat) *and* (Kekuatan *is* Sedang) *then* (Perilaku NPC *is* Menghindar)
   → Jarak (Dekat) = 0.2
   Kekuatan (Sedang) = 0.75
   Min (0.2 , 0.75) = 0.2
3. *If* (Jarak *is* Dekat) *and* (Kekuatan *is* Kuat) *then* (Perilaku NPC *is* menyerang Brutal)

→ Jarak (Dekat) = 0,2

Kekuatan (Kuat) = 0

Min (0.2 , 0) = 0

4. *If* (Jarak *is* Sedang) *and* (Kekuatan *is* Lemah) *then* (Perilaku NPC *is* Melarikan diri)

→ Jarak (Sedang) = 0,25

Kekuatan (Lemah) = 0

Min (0.25, 0) = 0

5. *If* (Jarak *is* Sedang) *and* (Kekuatan *is* Sedang) *then* (Perilaku NPC *is* Menambah darah)

→ Jarak (Sedang) = 0.25

Kekuatan (Sedang) = 0.75

Min (0.25 ,0.75) = 0 .25

6. *If* (Jarak *is* Sedang) *and* (Kekuatan *is* Kuat) *then* (Perilaku NPC *is* Menyerang)

→ Jarak (Sedang) = 0.25

Kekuatan (Kuat) = 0

Min (0.25 ,0) = 0

7. *If* (Jarak *is* Jauh) *and* (Kekuatan *is* Lemah) *then* (Perilaku NPC *is* Melarikan diri)

→ Jarak (Jauh) = 0

Kekuatan (Lemah) = 0

Min (0,0) = 0

8. *If* (Jarak *is* Jauh) *and* (Kekuatan *is* Sedang) *then* (Perilaku NPC *is* Menambah darah)

→ Jarak (Jauh) = 0

Kekuatan (Sedang) = 0,75

Min (0,0.75) = 0

9. *If* (Jarak *is* Jauh) *and* (Kekuatan *is* Kuat) *then* (Perilaku NPC *is* Bertahan)

→ Jarak (Jauh) = 0

Kekuatan (Kuat) = 0

Min (0,0) = 0

C. Defuzzy

1. Rule 1 = Melarikan diri

$$\frac{z-80}{8}=0$$

Z = 80 – 0 = 80

2. Rule 2 = Menghindar

$$\frac{z-48}{8}=0{,}2$$

Z = 48 – 1,6 = 46,4

3. Rule 3 = menyerang Brutal

$$\frac{16-z}{8}=0$$

Z = 16 – 0 = 16

4. Rule 4 = Melarikan diri

$$\frac{z-80}{8}=0$$

Z = 80 – 0 = 80

5. Rule 5 = Menambah darah

$$\frac{z-64}{8}=0{,}25$$

Z = 64 – 2 = 66

6. Rule 6 = Menyerang

$$\frac{z-16}{8}=0$$

Z = 16 – 0 = 16

7. Rule 7 = Melarikan diri

$$\frac{z-80}{8}=0$$

Z = 80 – 0 = 80

8. Rule 8 = Menambah darah

$$\frac{z-64}{8}=0$$

$$Z = 64 - 0 = 64$$

9. Rule 9 = Bertahan

$$\frac{z - 32}{8} = 0$$

$$Z = 32 - 0 = 32$$

$$z^* = \frac{\sum \mu i.zi}{\sum \mu i}$$

$$z = \frac{0\text{x } 80 + 0{,}2 \text{ x } 46{,}4 + 0 \text{ x } 16 + 0 \text{ x } 80 + 0{,}25 \text{ x } 66 + 0 \text{ x } 16 + 0\ x\ 80 + 0\ x\ 64 + 0\ x\ 32}{0 + 0{,}2 + 0 + 0 + 0{,}25 + 0 + 0 + 0 + 0}$$

$$= \frac{25{,}78}{0{,}45}$$

$$= 57{,}28$$

Z* = 57,28

Jadi Keputusan yang dihasilkan berdasarkan dari nilai jarak [35] dan Kekuatan [45] adalah Menghindar. Sesuai dengan nilai z* yang didapatkan yaitu 57,28.

3.1. 4 Analisis User

Pada analisis *user* (pemakai) ini akan mencakup analisis beberapa parameter terhadap calon *user* dari aplikasi.

1. *User Knowledge and Experience* dari target user yang akan menggunakan *game*.

   Game The Tracker ini bisa digunakan oleh kalangan apapun, tetapi pengetahuan dan pengalaman akan memudahkan *user* dalam penggunaanya. Terutama pengetahuan dan pengalaman dalam memainkan *game action* atau *first person shooter*. Berikut ini klasifikasi *knowledge and experience* dari pengguna applikasi :

Tabel 3. 12 Analisi Klasifikasi *Knowledge and experience*

Educational level	***Reading Level***	***Typing Skills***
Game ini bisa digunakan oleh berbagai kalangan, seperti pelajar, mahasiswa hingga masyarakat awam.	*Game* ini bisa digunakan oleh berbagai *level* pendidikan dengan *reading level* yang sedang	*Game* ini tidak memerlukan *typing skills* yang tinggi
Computer Literacy	***Task Experience***	***System Experience***
Game ini bisa digunakan oleh pengguna yang memiliki kemampuan komputer yang *moderate* (menengah)	*Game* ini bisa digunakan oleh dengan pengalaman penggunaan computer dan *game* yang sedang	*Game* ini bisa digunakan oleh dengan pengalaman penggunaan komputer dan *game* yang sedang
Application Experience	***Native Language***	***Use Of Other System***
Game ini bisa digunakan dalam semua sistem operasi Windows xp/vista/7	*Game* ini menggunakan dua bahasa, yakni Indonesia dan Inggris	*Game* ini bisa di jalankan tanpa perlu *install* aplikasi lain

2. *Users Physical Characteristic*

Keadaan fisik seseorang mungkin akan berpengaruh pada penggunaan aplikasi *game* ini. Ada hal-hal yang harus diperhatikan juga terhadap user dari karakteristik fisiknya untuk dapat menggunakan aplikasi ini yaitu, *Color Blind, Handednes, dan Gender*.

Tabel 3. 13 Analisis *Users Physical Characteristic*

Age	13 tahun keatas
Gender	Pria dan Wanita
Handedness	Kanan dan Kiri
Color Blind	*User* yang tidak bisa membedakan warna yang satu dengan yang lainnya (buta warna) masih mampu menggunakan aplikasi ini, karena tidak ada *indicator* warna-warna khusus yang membedakan antara fungsional yang satu dengan fungsional yang lainnya. Akan tetapi penggunaannya tidak akan optimal karena dalam *game* ini terdapat banyak sekali perbedaan warna yang menunjang interaksi dan ketertarikan dalam permainan.

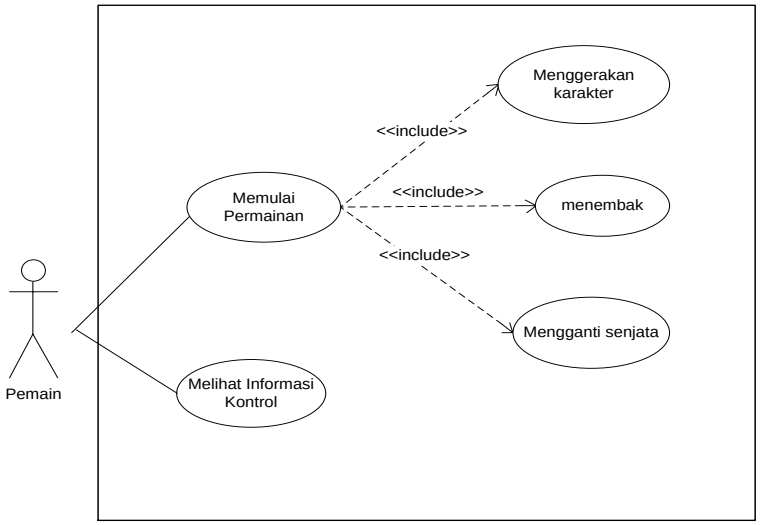
3.1. 5 Analisis Kebutuhan Fungsional

Analisis kebutuhan fungsional menggambarkan proses kegiatan yang akan diterapkan dalam sebuah sistem dan menjelaskan kebutuhan yang diperlukan sistem agar sistem dapat berjalan dengan baik sesuai kebutuhan.

Analisis kebutuhan fungsional dimodelkan dengan menggunakan *UML (Unified Modeling Language).* Dimana tahap-tahap analisis yang dilakukan dalam membangun *game first person shooter* 3D *The Tracker* antara lain *Use Case Diagram, Activity Diagram, Class Diagram, Sequence Diagram.*

3.1.5. 1 *Use Case Diagram*

Use case adalah abstraksi dari interaksi antara sistem dan dan aktor. *Use case* mendeskripsikan bagaimana sistem akan terlihat di mata *user*. Secara umum, interaksi antara pengguna dengan aplikasi permainan *The Tracker* digambarkan pada *use case* diagram 3.20 dibawah ini :



Gambar 3. 20 *Use Case Diagram*

3.1.5. 2 *Definisi Actor*

Definisi *Actor* berfungsi untuk menjelaskan *actor* yang terdapat pada *use case diagram*. Definisi *actor* diterangkan pada tabel 3.25 berikut.

Tabel 3. 14 *Definisi* Aktor

No	Aktor	Deskripsi
1.	Pengguna	Pemain atau orang yang memainkan *game.*

3.1.5.3 *Definisi Use Case*

Definisi *use case* berfungsi untuk menjelaskan fungsi *use case* yang terdapat pada *use case diagram*. Definisi *use case* diterangkan pada *table*.

Tabel 3. 15 *Definisi Use Case*

No.	*Use Case*	Deskripsi
1.	Memulai Permainan	Proses untuk memulai permainan.
2.	Melihat Informasi kontrol	Proses untuk menampilkan petunjuk dan panduan bermain.
3.	Menggerakan karakter	Proses untuk menggerakkan karakter dengan animasi berjalan, berlari, melompat.
4.	Menembak	Proses untuk menembak
5.	Mengganti Senjata	Proses untuk menggerakkan karakter dengan animasi mengganti senjata

3.1.5. 4 *Scenario Use Case*

Skenario pada diagram *use case* berisi mengenai alur kejadian '*flow of event' use case* utama. Didalamnya menjelaskan urutan interaksi antara aktor dengan *use case* tersebut dari awal sampai akhir. Adapun skenario *use case* yang terdapat pada aplikasi *Game The Tracker* adalah sebagai berikut.

Tabel 3. 16 *Scenario Use Case* Memulai Permainan

Identifikasi		
Nomor	1	
Nama	Memulai Permainan	
Tujuan	Melihat *cutscene* / cerita permainan dan masuk ke dalam permainan	
Deskripsi	Proses untuk melihat *cutscene* / cerita permainan dan masuk ke dalam permainan	
Aktor	Pengguna	
Skenario Utama		
Kondisi awal	Pemain berada di menu utama	
Aksi Aktor		Reaksi Sistem
1. Pengguna memilih menu mainkan.		2. Sistem menjalankan tampilan *cutscene.*
3. Pengguna memainkan.		4. Sistem menjalankan permainan.
Kondisi Akhir	Sistem menampilkan permainan	

Tabel 3. 17 *Scenario Use Case* Melihat Informasi kontrol

Identifikasi		
Nomor	2	
Nama	Melihat Informasi Kontrol	
Tujuan	Menampilkan informasi kontrol	
Deskripsi	Proses untuk menampilkan informasi kontrol	
Aktor	Pengguna	
Skenario Utama		
Kondisi awal	Pemain berada di menu utama	
Aksi Aktor		**Reaksi Sistem**
1. Pengguna memilih menu Petunjuk		2. Menampilkan informasi tentang kontrol *keyboard* dan *mouse*.
Kondisi Akhir	Sistem akan menampilkan informasi kontrol	

Tabel 3. 18 *Scenario Use Case* Menggerakan karakter

Identifikasi	
Nomor	3
Nama	Menggerakan karakter
Tujuan	Menggerakkan karakter
Deskripsi	Proses untuk menggerakan karakter atau tokoh utama yang dimainkan dalam permainan menggunakan tombol *keybord*
Aktor	Pengguna
Skenario Utama	
Kondisi awal	Pemain menekan tombol pada *keyboard*
Aksi Aktor	**Reaksi Sistem**
1. Pengguna menggerakkan karakter dengan interaksi menggunakan *keyboard* menekan tombol W.	2. Sistem menggerakkan karakter menjalankan animasi berjalan ke depan.
Alternatif	
3. Pengguna menggerakkan karakter dengan interaksi menggunakan *keyboard* menekan tombol A.	4. Sistem menggerakkan karakter menjalankan animasi berjalan ke kiri.
Alternatif	
5. Pengguna menggerakkan karakter dengan interaksi menggunakan *keyboard* menekan tombol S.	6. Sistem menggerakkan karakter menjalankan animasi berjalan ke belakang.
Alternatif	
7. Pengguna menggerakkan karakter dengan interaksi menggunakan *keyboard* menekan tombol D.	8. Sistem menggerakkan karakter menjalankan animasi berjalan ke kanan.
Alternatif	
9. Pengguna menggerakkan karakter dengan interaksi menggunakan *keyboard* menekan tombol Shift + W.	10. Sistem menggerakkan karakter menjalankan animasi berlari kedepan.
Alternatif	
11. Pengguna menggerakkan karakter dengan interaksi menggunakan *keyboard* menekan tombol Shift + A.	12. Sistem menggerakkan karakter menjalankan animasi berlari kekiri.
Alternatif	
13. Pengguna menggerakkan karakter dengan interaksi menggunakan *keyboard* menekan tombol Shift + D.	14. Sistem menggerakkan karakter menjalankan animasi berlari kekanan.
15. Menekan tombol spasi pada keybord.	16. Karekter pemain melompat
Kondisi Akhir	Karakter bergerak.

Tabel 3. 19 *Scenario Use Case* Menembak

Identifikasi	
Nomor	4
Nama	Menembak

Tujuan	Menggerakkan karakter dengan animasi menembak.	
Deskripsi	Proses menggerakkan karakter dengan animasi menembak.	
Aktor	Pengguna	
Skenario Utama		
Kondisi awal	Pengguna memainkan *game*	
Aksi Aktor		**Reaksi Sistem**
1. Pengguna menggerakkan karakter untuk animasi menembak dengan interaksi menggunakan tombol kiri *mouse*.		2. Sistem mengeluarkan animasi menembak.
Kondisi Akhir	Sistem menggerakkan karakter menjalankan animasi menembak sesuai inputan pengguna.	

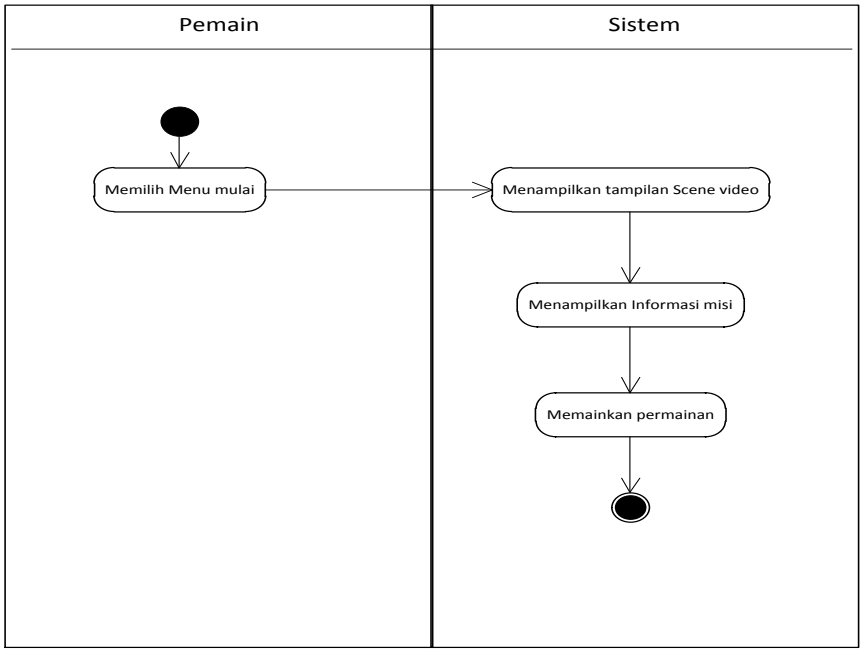
Tabel 3. 20 *Scenario Use Case* Mengganti Senjata

Identifikasi		
Nomor	5	
Nama	Mengganti Senjata	
Tujuan	Menggerakkan karakter dengan animasi mengganti senjata.	
Deskripsi	Proses menggerakkan karakter dengan animasi mengganti senjata.	
Aktor	Pengguna	
Skenario Utama		
Kondisi awal	Pengguna menggunakan senjata ScarNew	
Aksi Aktor		**Reaksi Sistem**
1. Pengguna menggerakkan karakter menjalankan animasi mengganti senjata dengan interaksi menggunakan *keyboard* menekan tombol 1.		2. Sistem menampilkan karakter menjalankan animasi mengganti senjata ScarNew.
Alternatif		
3. Pengguna menggerakkan karakter menjalankan animasi mengganti senjata dengan interaksi menggunakan *keyboard* menekan tombol 2.		4. Sistem menampilkan karakter menjalankan animasi mengganti senjata *sniper*.
Kondisi Akhir	Sistem menampilkan karakter menjalankan animasi mengganti senjata ScarNew dan *sniper* sesuai inputan pengguna.	

3.1.5. 5 *Activity Diagram*

Activity diagram menggambarkan berbagai aliran aktivitas dalam sistem yang sedang dirancang. Berikut ini beberapa *activity diagram* yang terdapat pada *game* ini yaitu :

A. *Activity Diagram* Proses mulai permainan

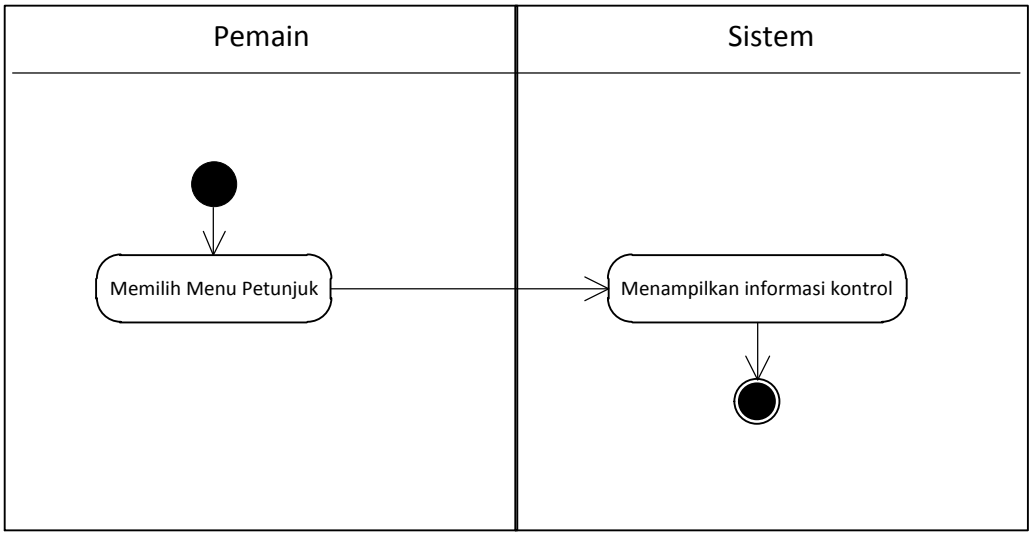


Gambar 3. 21 *Activity diagram* Memilih Memulai Permainan

Tabel 3. 21 Penjelasan Proses Memulai Permainan

Deskripsi	
Menu mulai merupakan menu yang digunakan untuk mulai memainkan permainan.	
Normal Flow	
1. Pengguna memilih menu mulai di menu utama	2.Sistem menjalankan tampilan scene video. 3. Sistem menampilkan Informasi misi. 4. Sistem menjalankan permainan.

B. *Activity Diagram* Melihat Informasi kontrol

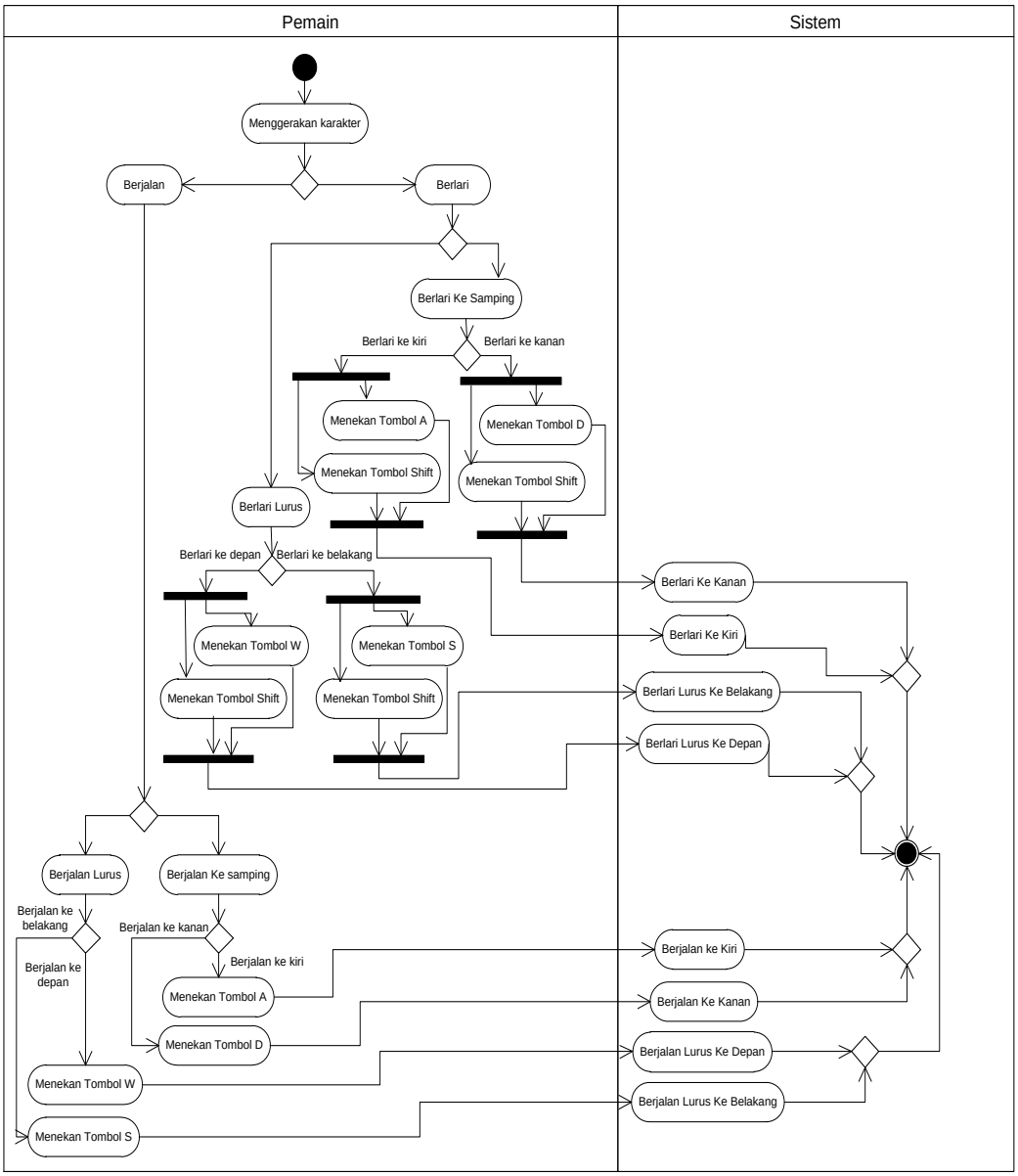


Gambar 3. 22 *Activity diagram* Memilih Melihat Tampilan Petunjuk

Tabel 3. 22 Penjelasan Proses Melihat Informasi kontrol

Deskripsi	
Menu petunjuk digunakan pemain untuk melihat informasi kontrol.	
Normal Flow	
1. Pengguna memilih menu petunjuk di menu utama.	2. Sistem menampilkan halaman informasi kontrol

C. *Activity Diagram* Menggerakan karakter



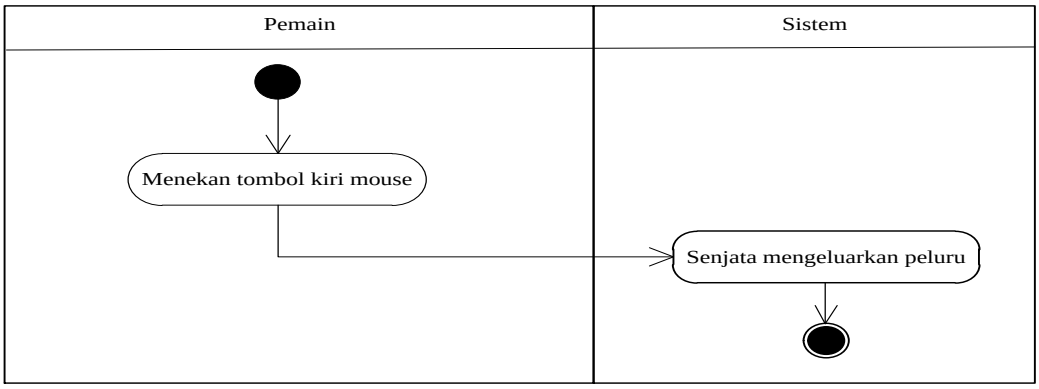
Gambar 3. 23 *Activity diagram* Menggerakan karakter

Tabel 3. 23 Penjelasan Proses Menggerakan karakter

Deskripsi	
Proses untuk menggerakan karakter atau tokoh utama yang dimainkan dalam permainan menggunakan tombol *keyboard.*	
Normal Flow	
1. Pemain menggerakan karakter dengan menekan tombol W,	2. karakter berjalan kedepan.
Alternatif	

3. Pemain menekan tombol Shift dan W secara bersamaan,	4. Karakter berlari kedepan.
Alternatif	
5. Pemain menggerakan karakter dengan menekan tombol A,	6. Karakter berjalan kekiri
Alternatif	
7. Pemain menekan tombol Shift dan A secara bersamaan,	8. Karakter berlari kekiri
Alternatif	
9. Pemain menekan tombol s	10. Karakter mundur kebelakang
Alternatif	
11. Pemain menggerakan karakter dengan menekan tombol D,	12. Karakter berjalan kekanan
Alternatif	
13. Pemain menekan tombol Shift dan D secara bersamaan,	14. karakter berlari kekanan
Alternatif	
15. pemain menekan tombol spasi,	16.karakter melompat

D. *Activity Diagram* Menembak

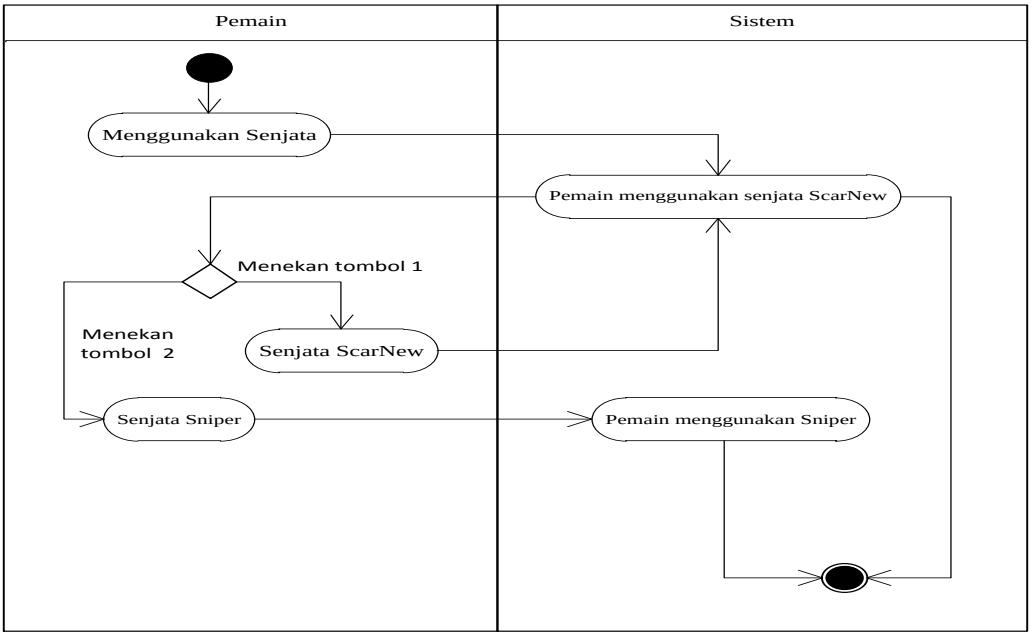


Gambar 3. 24 *Activity Diagram* Menembak

Tabel 3. 24 Penjelasan Proses Menembak

Deskripsi	
Proses untuk melakukan tembakan.	
Normal Flow	
1. Pemain menekan tombol kiri pada *mouse* untuk menembak	2.Sistem menampilkansenjata yang mengeluarkan peluru sesuai inputan pemain.

E. *Activity Diagram* Mengganti Senjata



Gambar 3. 25 *Activity Diagram* Mengganti Senjata

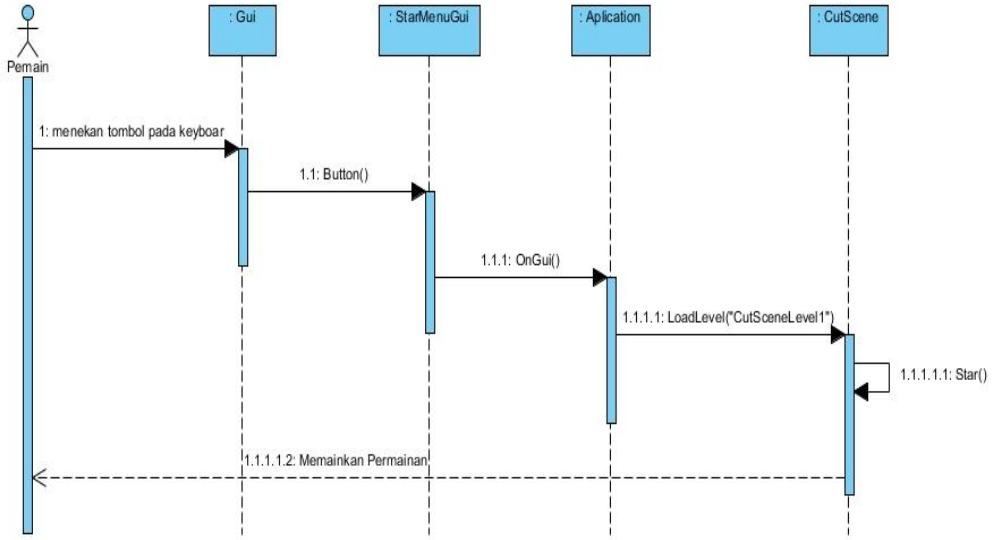
Tabel 3. 25 Penjelasan proses Mengganti Senjata

Deskripsi	
Proses mengganti senjata dengan pilihan senjata ScarNew atau senjata *sniper*.	
Normal Flow	
1. Pemain menggunakan senjata. 2. Pemain menekan tombol 1 pada *keyboard* 3. Pemain menekan tombol 2 pada *keyboard*	1. Sistem menampilkan pemain menggunakan ScarNew 2. Sistem tidak mengganti senjata 3. Sistem menampilkan pemain menggunakan senjata *sniper*

3.1.5.6 *Sequence Diagram*

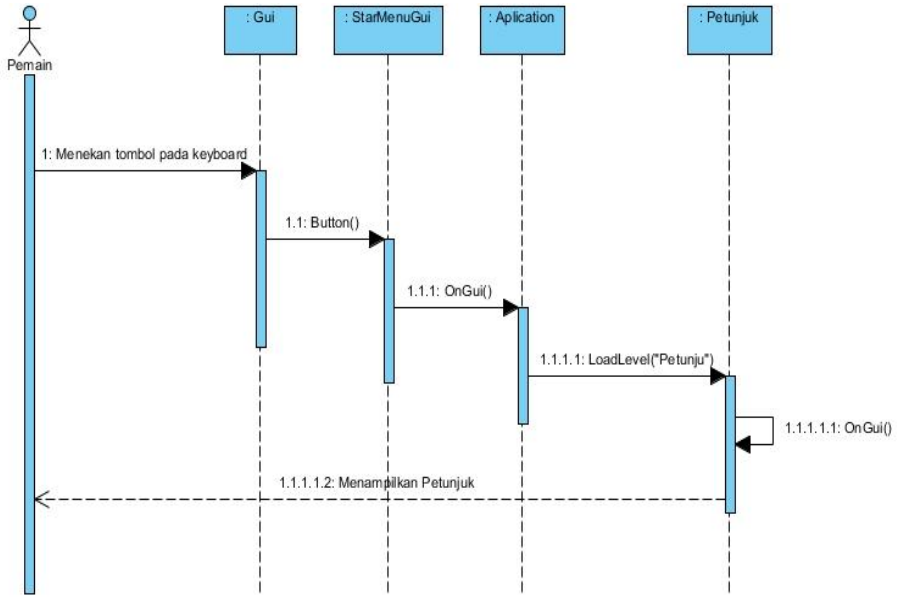
Sequencial diagram menggambarkan interaksi antar objek di dalam dan disekitar sistem berupa *message* yang digambarkan terhadap waktu. Berikut ini beberapa *sequencial diagram* yang terdapat pada *game The Tracker* yaitu :

1. *Sequence Diagram* Memulai Permainan.



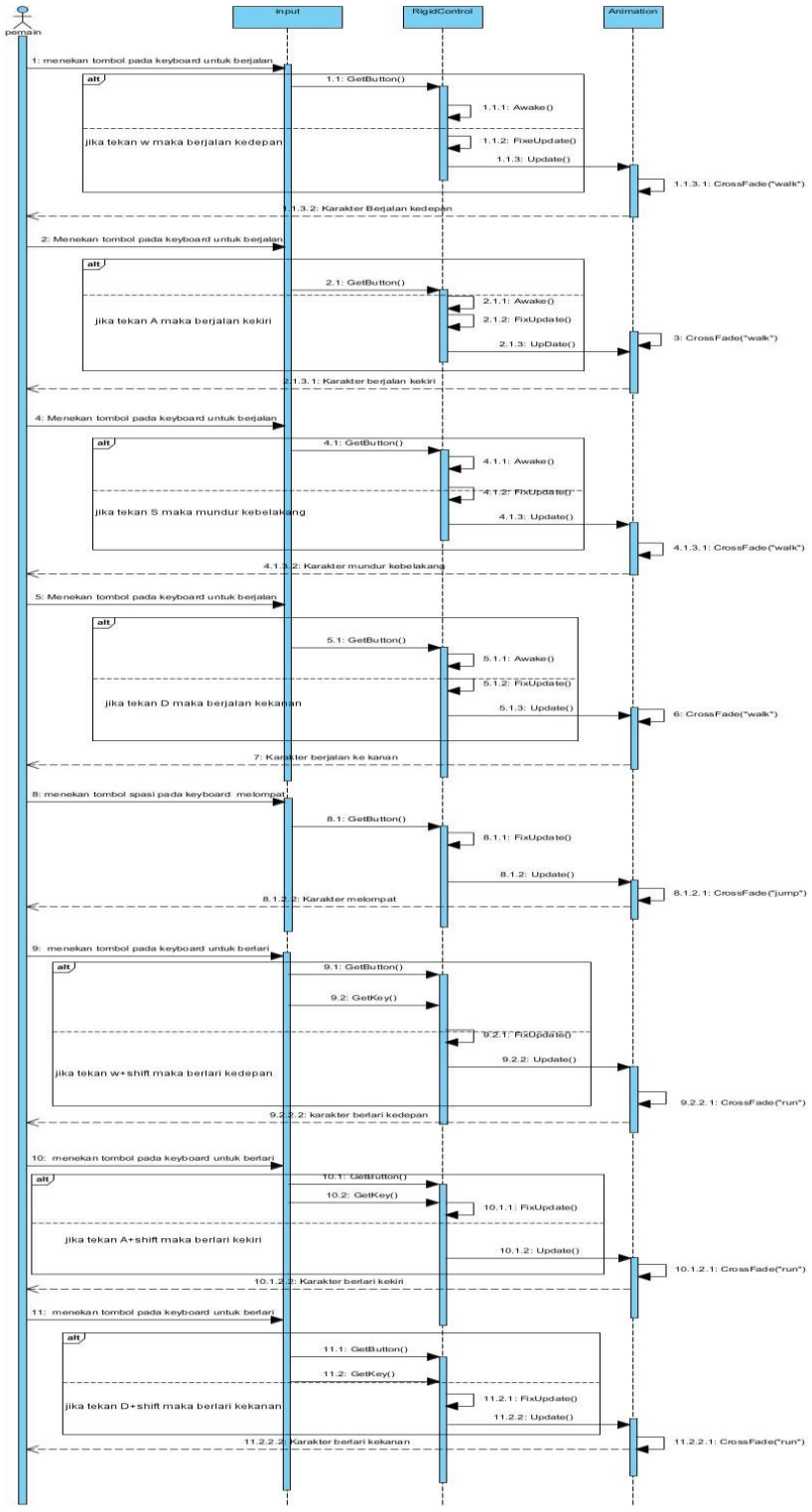
Gambar 3. 26 *Sequence Diagram* Memulai Permainan

2. *Sequence Diagram* Melihat Informasi kontrol



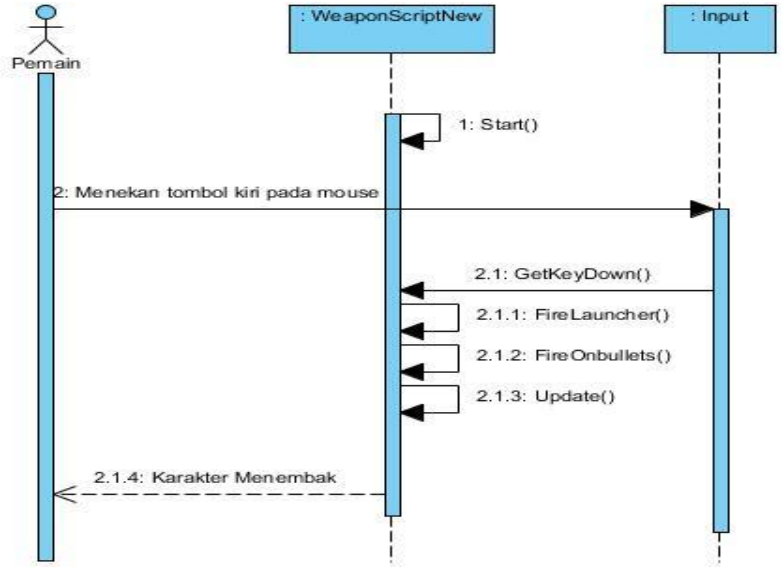
Gambar 3. 27 *Sequence Diagram* Melihat Informasi kontrol

3. *Sequence Diagram* Menggerakan Karakter



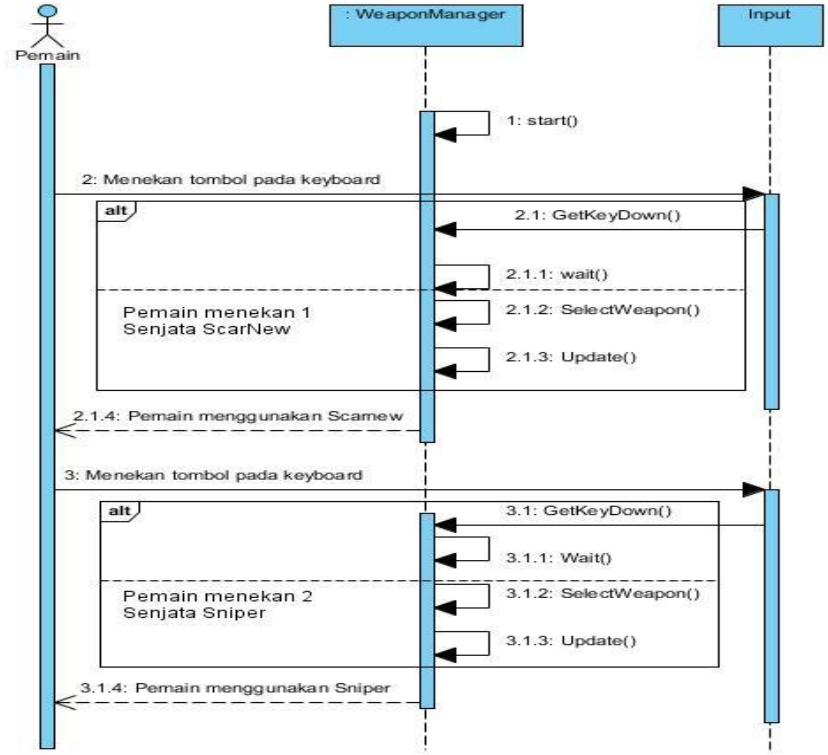
Gambar 3. 28 *Sequence Diagram* Menggerakan karakter

4. *Sequence Diagram* Menembak



Gambar 3. 29 *Sequence Diagram* Menembak

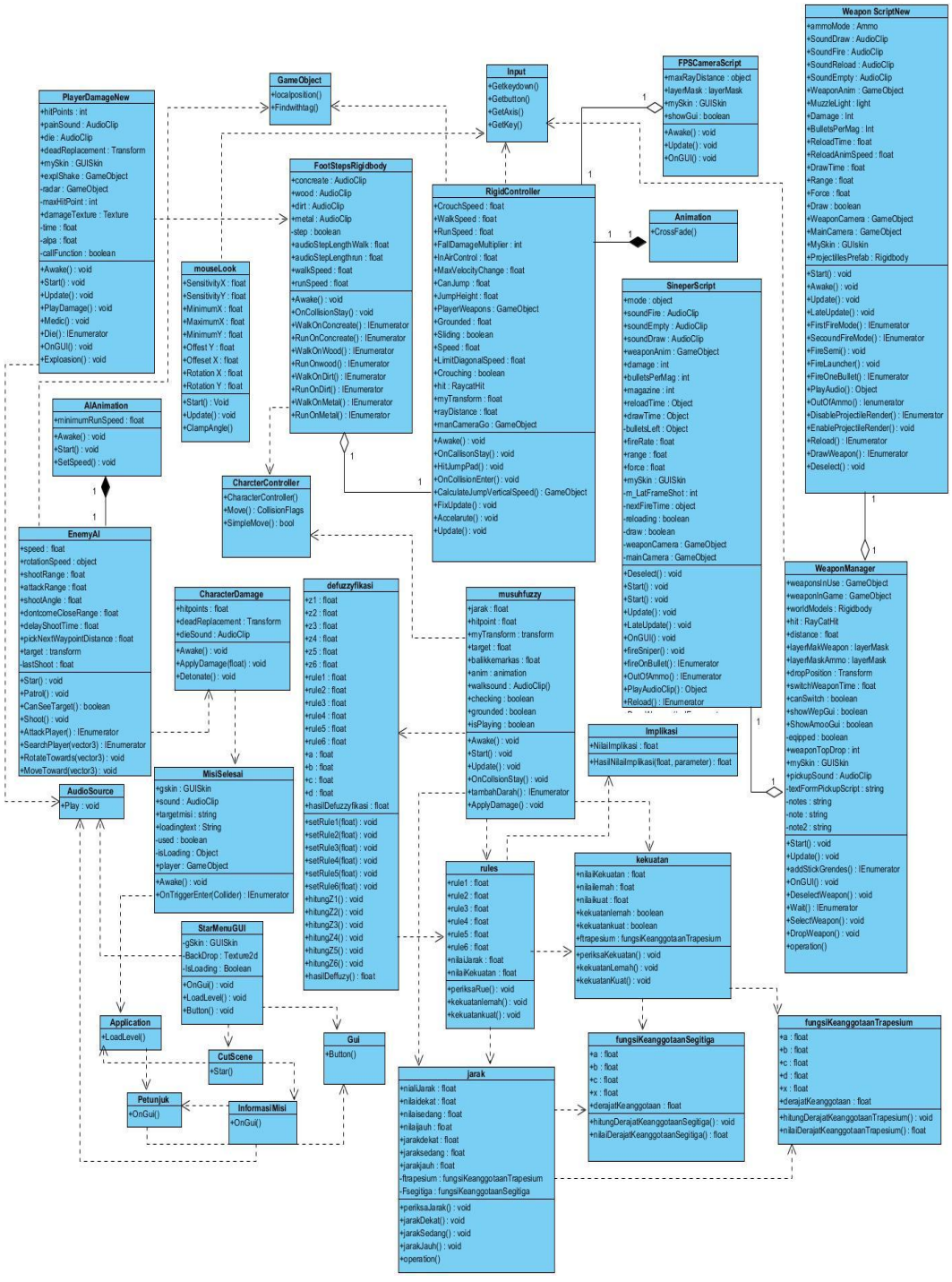
5. *Sequence Diagram* Mengganti senjata



Gambar 3. 30 *Sequence Diagram* Mengganti senjata

3.1.5.7 *Class Diagram*

Class diagram menggambarkan struktur sistem dari segi pendefinisian kelas-kelas yang dibuat untuk membangun sistem. Berikut adalah class diagram pada game *The Tracker*.



Gambar 3. 31 *Class Diagram game The Tracker*

3.1. 6 Analisis Kebutuhan Non-Fungsional

Analisis non-fungsional merupakan analisis yang dibutuhkan untuk menentukan spesifikasi kebutuhan sistem. Spesifikasi ini juga meliputi elemen atau komponen-komponen apa saja yang dibutuhkan untuk sistem yang akan dibangun sampai dengan sistem tersebut diimplementasikan. Analisis kebutuhan ini juga menentukan spesifikasi masukan yang diperlukan sistem, keluaran yang akan dihasilkan sistem dan proses yang dibutuhkan untuk mengolah masukan sehingga menghasilkan suatu keluaran yang diinginkan. Pada analisis kebutuhan sistem non fungsional ini dijelaskan analisis kebutuhan perangkat lunak, analisis kebutuhan perangkat keras, dan analisis pengguna.

3.1.6. 1 Analisis Kebutuhan Perangkat Lunak

Perangkat lunak digunakan dalam sebuah sistem merupakan perintah-perintah yang diberikan kepada perangkat keras agar bisa saling berinteraksi diantara keduanya.

Perangkat lunak yang dibutuhkan pengembang untuk pembangunan *Game* ini adalah :

1. Sistem Operasi Microsoft Windows 7
2. Unity versi 4.0
3. 3ds Max 2011
4. Movie Edit Pro MX Plus

3.1.6.2 Analisis Kebutuhan Perangkat Keras

Komputer terdiri dari perangkat keras dan perangkat lunak yang saling berinteraksi. Perangkat lunak memberikan instruksi-instruksi kepada perangkat keras untuk melakukan suatu tugas tertentu, sehingga dapat menjalankan suatu sistem di dalamnya.

Perangkat keras minimum bagi pengguna untuk dapat memainkan game ini dengan baik adalah sebagai berikut :

1. Processor Dual Core 1,80 Ghz
2. RAM 1 GB
3. VGA 512 MB
4. Harddisk kosong tersedia 300 MB

5. Mouse dan Keyboard.

3. 2 Perancangan Sistem

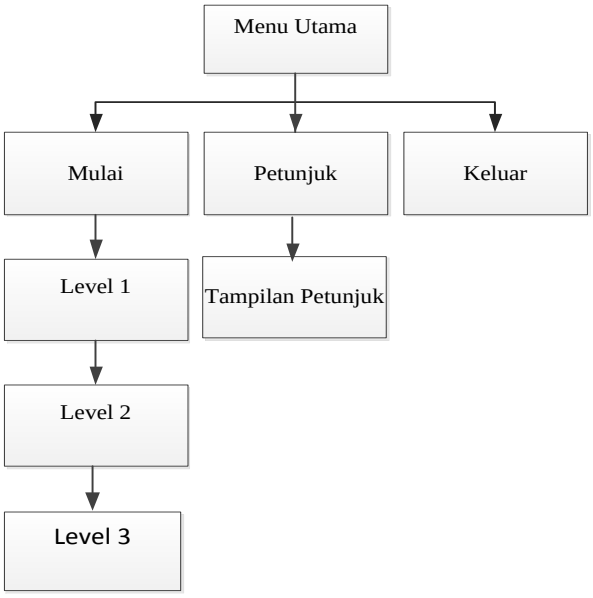
Perancangan adalah suatu bagian dari metodologi pengembangan suatu perangkat lunak yang dilakukan setelah tahapan untuk memberikan gambaran secara terperinci. Dalam tahap ini digambarkan rancangan sistem yang akan dibangun sebelum dilakukan pengkodean ke dalam suatu bahasa pemrograman.

3.2. 1 Perancangan Perangkat Lunak

Perancangan perangkat lunak adalah kegiatan untuk merancang aplikasi yang akan di bangun, tahapannya dimulai dari perancangan arsitektur menu, lalu berlanjut ke perancangan antar muka dan jaringan semantik.

3.2. 2 Perancangan Arsitektur Menu

Untuk mempermudah pembuatan sistem diperlukan perancangan struktur menu program yang akan dibangun. Perancangan struktur menu program ini membantu dalam merancang bagian-bagian dari sistem yang sebenarnya dan untuk mengetahui bagian mana yang terlebih dahulu nantinya yang akan diakses setelah program tersebut selesai. Berikut di bawah ini menunjukkan perancangan arsitektur menu dari sistem yang akan dibangun.



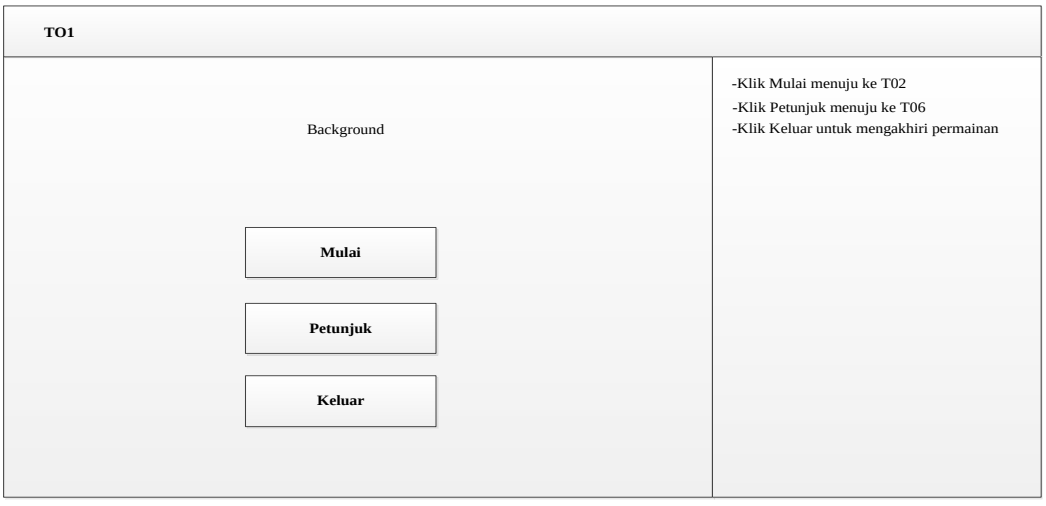
Gambar 3. 32 Arsitektur Menu *Game The Tracker*

1. Mulai Permainan : merupakan submenu untuk memulai permainan
2. Petunjuk : merupakan submenu yang menampilkan informasi seputar petunjuk permainan seperti penekanan tombol tertentu pada keyboard beserta reaksinya.
3. Keluar Permainan : merupakan submenu untuk keluar dari permainan.
4. Level 1 : merupakan submenu untuk menampilkan layar permainan pada level 1.
5. Level 2 : merupakan submenu untuk menampilkan layar permainan pada level 2.
6. Level 3 : merupakan submenu untuk menampilkan layar permainan pada level 3.

3.2. 3 Perancangan Antar Muka

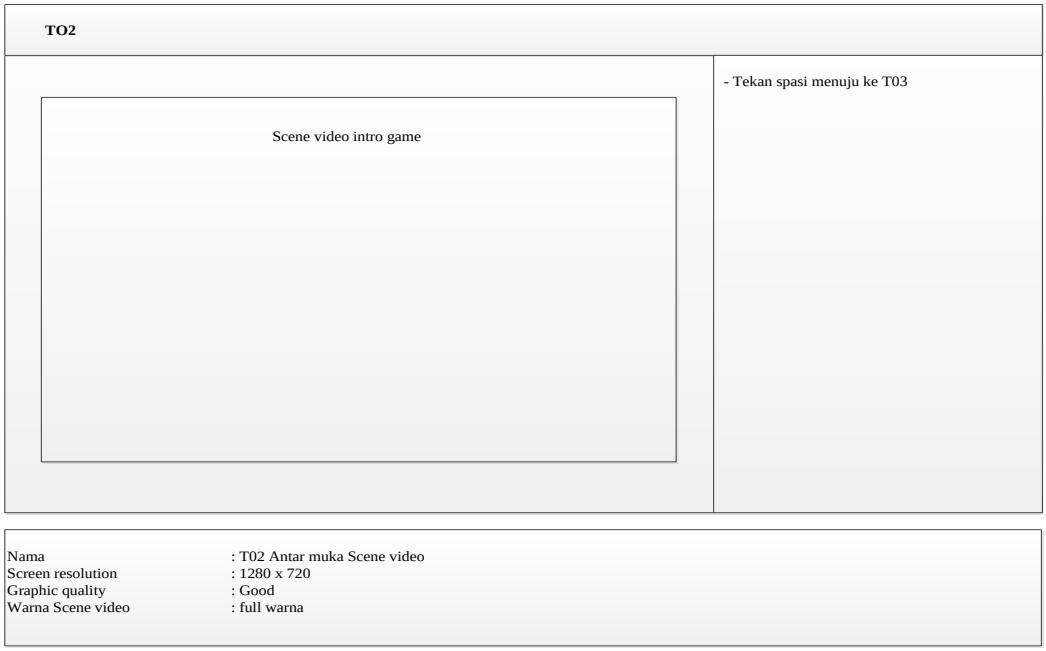
Perancangan antarmuka dilakukan untuk mempermudah dalam mengimplementasikan sistem yang akan dibangun. Berikut ini adalah antar muka pada *game The Tracker*.

1. Antar Muka Menu Utama



Gambar 3. 33 Tampilan Menu Utama

2. Antar Muka *Scene video* intro *game*



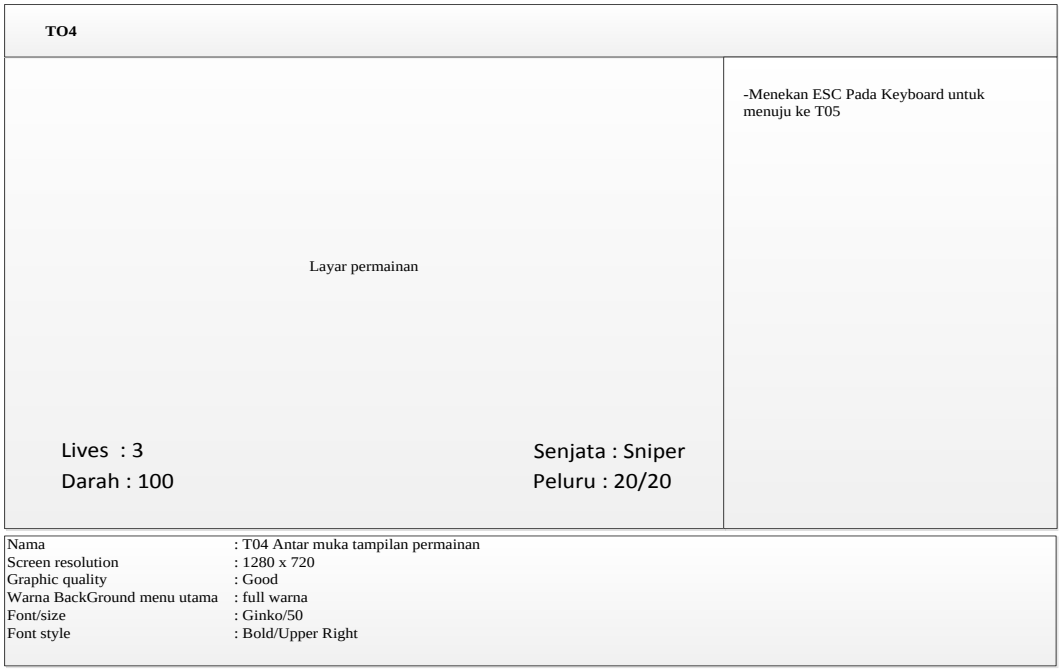
Gambar 3. 34 Tampilan *Scene video* intro *game*

3. Antar Muka Informasi misi



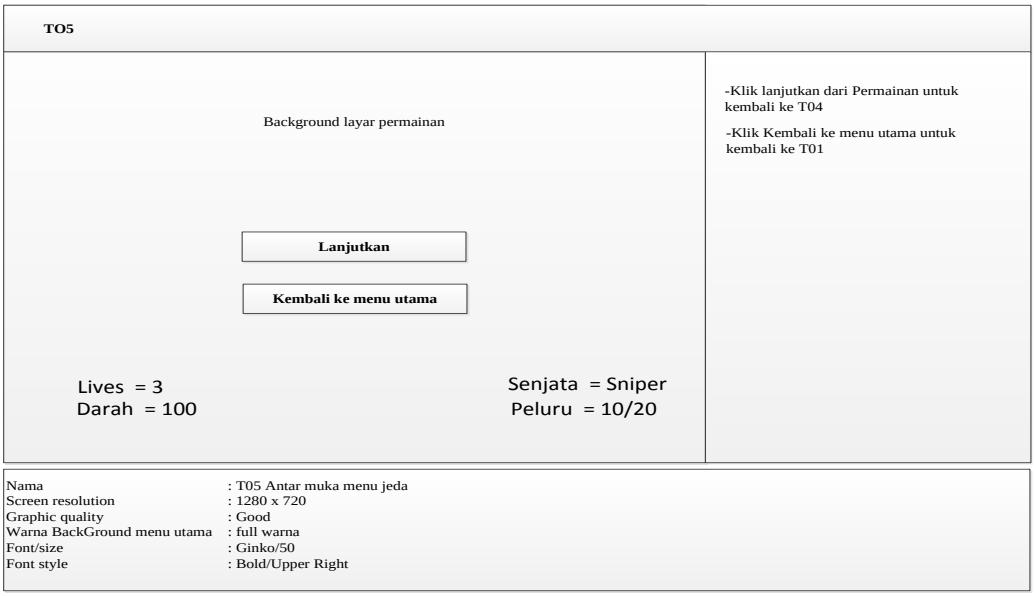
Gambar 3. 35 Tampilan Informasi misi

4. Antar Muka tampilan permainan



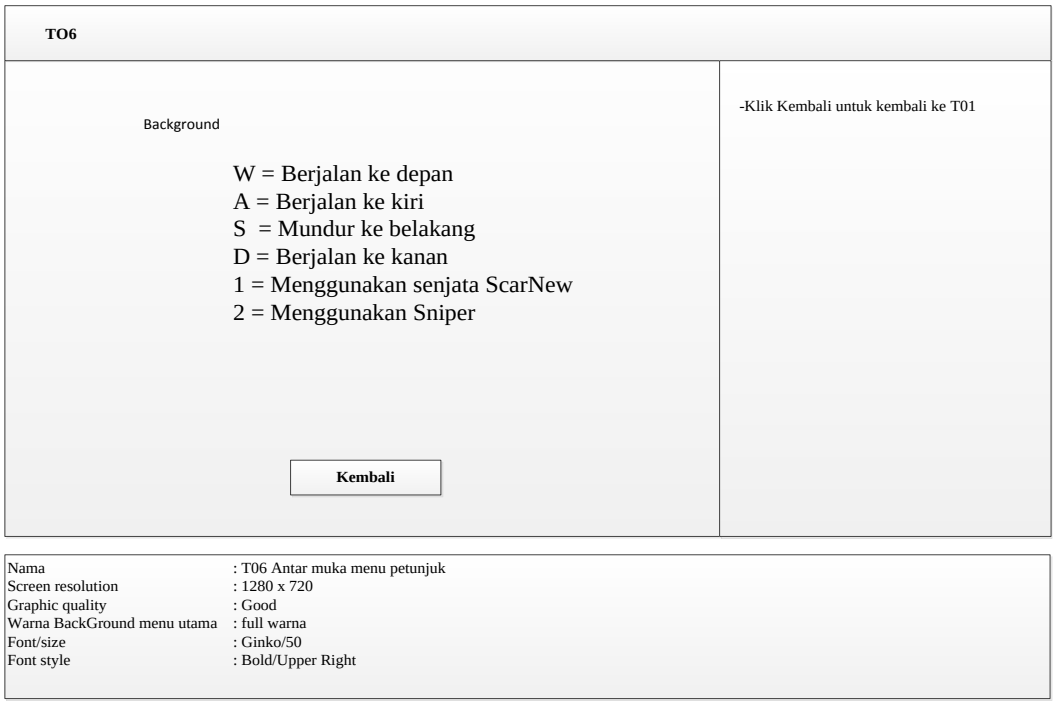
Gambar 3. 36 Tampilan Permainan

5. Antar Muka Jeda



Gambar 3. 37 Tampilan Jeda

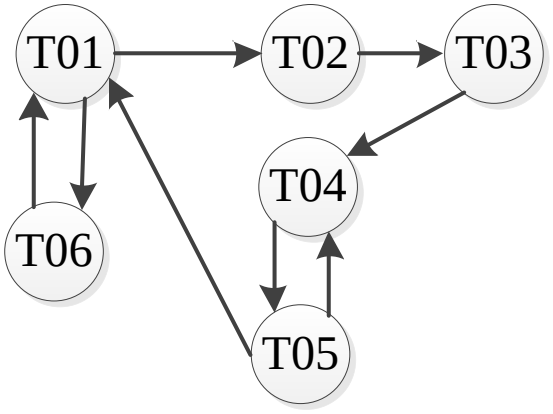
6. Antar muka tampilan menu petunjuk



Gambar 3. 38 Tampilan Petunjuk

3.2. 4 Jaringan Semantik

Struktur aplikasi ini dibuat secara moduler, yaitu program dipecah menjadi modul-modul kecil yang mudah dibuat, mudah dites, dan mudah dimodifikasi. Dalam pembuatan *game* ini menggunakan alat bantu bagan struktural yang disebut juga jaringan semantik.



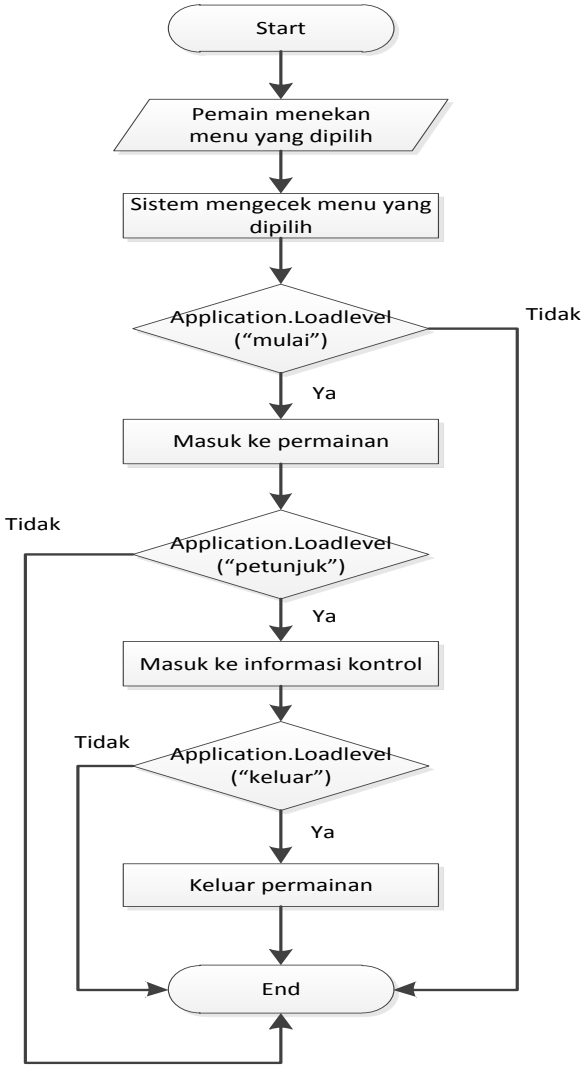
Gambar 3. 39 Jaringan Semantik

3.2. 5 Perancangan *Method*

Perancangan komponen *method* merupakan perancangan yang dibuat setelah perancangan arsitektur menu, antar muka, dan jaringan semantik. Perancangan ini berfungsi untuk mendeskripsikan method-method yang berada di dalam aplikasi. Adapun method-method yang terdapat dalam *game The Tracker* yang akan dibangun adalah sebagai berikut.

A. *Method OnGui*

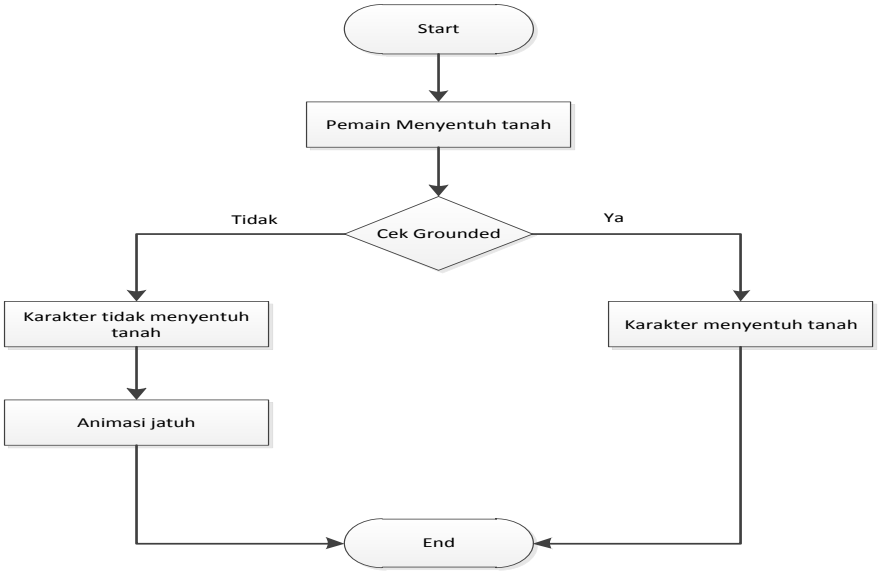
Method ini digunakan ketika pemain berada di menu utama tujuannya untuk memilih sebelum memulai permainan.



Gambar 3. 40 ***Flowchart Method OnGui***

B. *Method OnCollisionEnter*

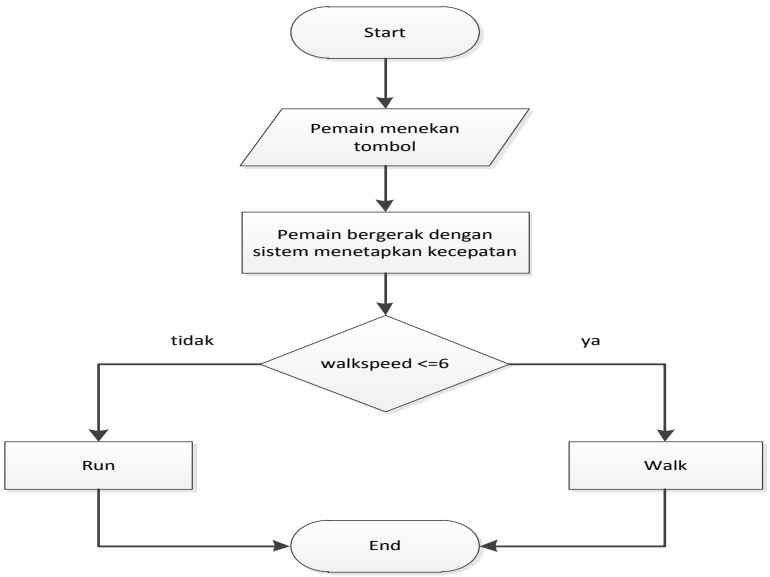
Method ini untuk mengecek apakah objek player sudah bersentuhan dengan tanah atau tidak.



Gambar 3. 41 *Flowchart Method OnCollisionEnter*

C. *Method Update*

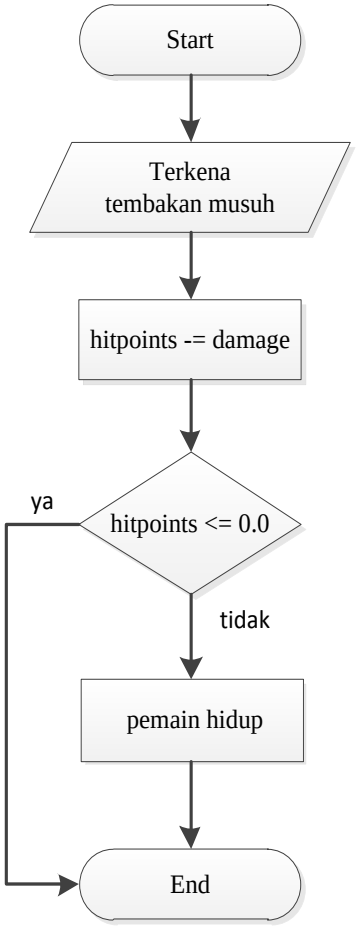
Method ini akan menampilkan pergerakan sebuah karakter dengan inputan keyboard dari pengguna, yang bertujuan untuk mengetahui apakah pemain sedang berjalan atau berlari.



Gambar 3. 42 *Flowchart Method Update*

D. *Method ApplyDamage*

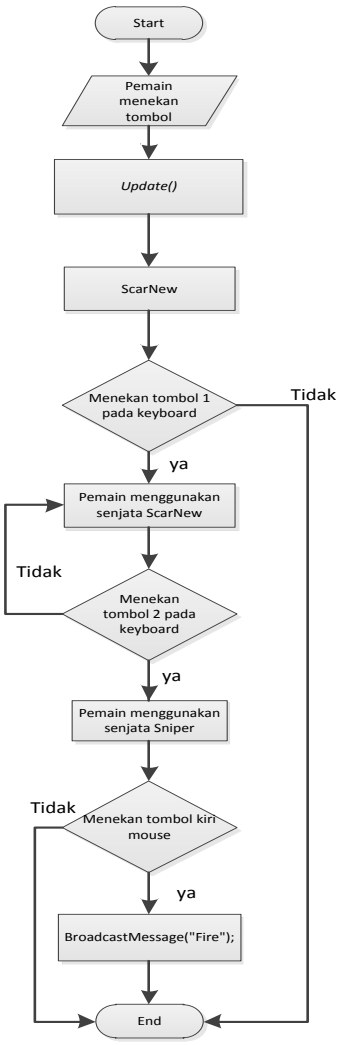
Method ini akan melakukan pengecekan terhadap setiap serangan dari musuh yang mengenai pemain.



Gambar 3. 43 *Flowchart Method ApplyDamage*

E. *Method Update* pada *weapon*

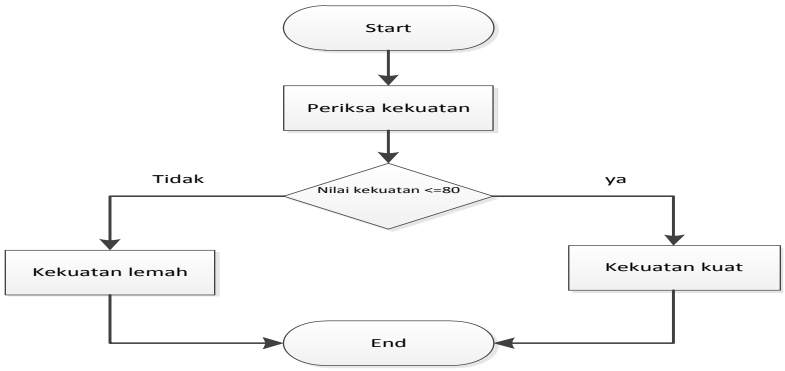
Method ini digunakan untuk memilih dan menggunakan senjata dengan inputan keyboard dari pengguna, sehingga karakter pemain dapat menggunakan senjata sesuai dengan keinginan dari pengguna di arena permainan.



Gambar 3. 44 *Flowchart Update* pada w*eapon*

F. *Method* Periksa kekuatan

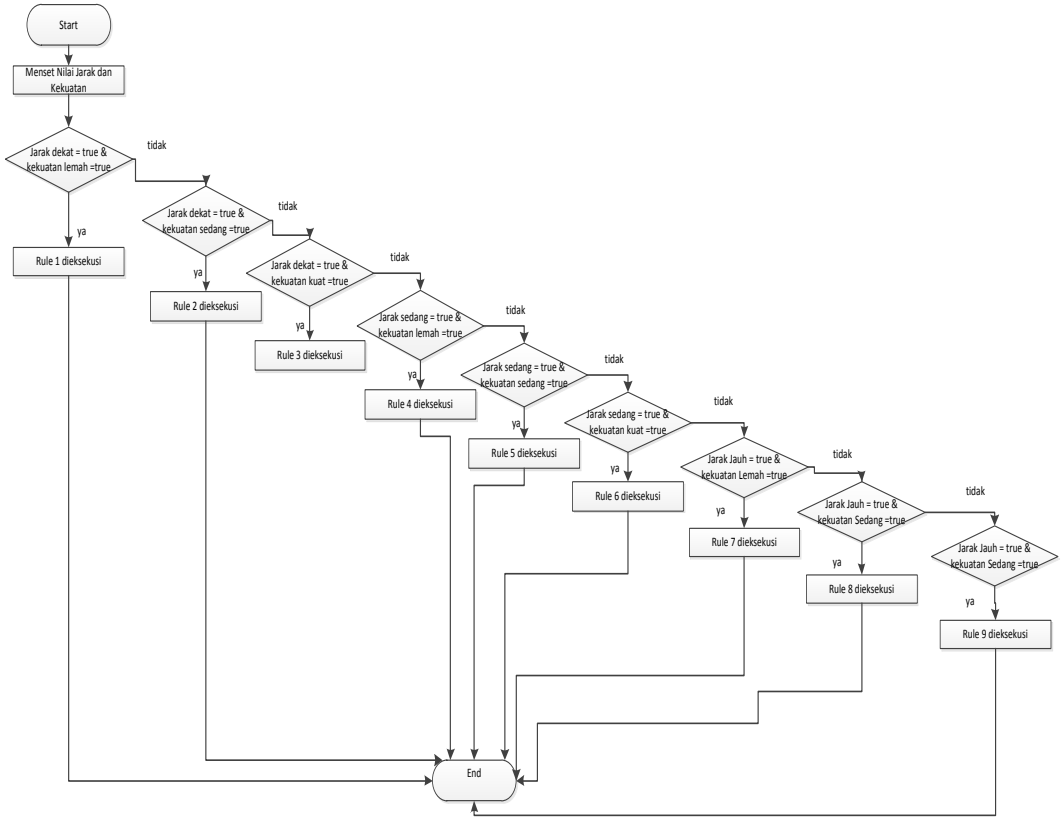
Method ini akan melakukan pengecekan kekuatan terhadap musuh di arena permainan.



Gambar 3. 45 *Flowchart Method* Periksa kekuatan

E. *Method* Periksa Rule

Method ini akan melakukan pengecekan Rule terhadap musuh di arena permainan.



Gambar 3. 46 *Flowchart Method* Periksa Rule