BAB III
ANALISIS DAN PERANCANGAN SISTEM

III.1 Analisis Sistem

Pada analisis sistem akan mencakup analisis *game* sejenis, analisis AI, analisis kebutuhan non-fungsional dan analisis kebutuhan fungsional.

III.1.1 Analisis *Game* Sejenis

Analisis *game* sejenis menjelaskan proses identifikasi serta evaluasi mengenai *game* sejenis serta analisis *game* yang dibangun.

Game dengan *genre third person shooter* sudah banyak sekali dalam dunia *game, genre* ini menarik untuk dimainkan karena *gameplay*nya yang *simple* membuat orang-orang mudah untuk memainkannya. Beberapa contoh *game* dengan *genre third person shooter* adalah *Army Men 3D, Tomb Raider, Resident Evil,* dan *The Punisher.*

Sampai saat ini sudah ada *game* dengan alur cerita yang diambil dari *film The Walking Dead,* yaitu *The Walking Dead Social Game* yang berbasis *web* dan hanya bisa dimainkan di *social media facebook.*

Dalam pembahasan ini dibahas *game* sejenis dari jenis *genre game* dan alur cerita yang diambil. *Game* yang dibahas dan dijadikan acuan disini adalah *game Army Med 3D* dan *The Walking Dead Social Game.* Secara *genre game, game Army Men 3D* serupa dengan *genre game The Walker* yang akan dibangun, sedangkan secara alur cerita serupa dengan yang diterapkan di dalam *game The Walking Dead Social Game.* Berikut adalah ulasan detail mengenai perbandingan *game* yang sejenis.

1. *Game Army Men 3D*

   Game Army Men 3D yang mengambil alur cerita dari peperangan dengan *genre third person shooter* dan dibuat oleh 3DO bisa dikatakan sangat sukses di pasaran pada masanya. Di masa ketika *film Small Soldiers* menaiki puncak *Box Office* dan mampu menarik perhatian jutaan penonton.



Gambar III. 1 *Game Army Men 3D*

Army Men merupakan mainan plastik yang berbentuk para pasukan dengan berbagai gaya dan perlengkapan senjata. Unsur kekerasan dalam *game* ini diminimalisir dengan penggunaan tentara-tentara mainan dari plastik sebagai wujud manusia di dalam *game.*

Disini pengguna berperan sebagai *Sarge* (panggilan *seargent* = sersan) dari pasukan tentara hijau yang mendapat perintah untuk menjalankan misi-misi berbahaya seperti menghancurkan pusat komunikasi, menyelamatkan tawanan, mencari informasi dari mata-mata dan sebagainya. Peralatan dan senjata yang disediakan berbeda antara satu misi dengan misi lainnya. Pihak musuh adalah pasukan dari tentara cokelat yang mencoba untuk menginvasi pasukan dari tentara hijau.



Gambar III. 2 Tampilan Salah Satu Scene *Game Army Men 3D*

Army Men 3D menampilkan desain dari mainan plastik di dalam *game*nya. *Storyline* yang dibangun juga menampilkan sisi ketegangan dari penggunanya.

Game ini mempunyai berbagai keterbatasan yang ada. Warna musuh dan lingkungannya yang nyaris mirip, tidak ada indikator arah tembakan dan *crosshair* bidikan membuat pengguna harus sangat berhati-hati untuk sekadar melangkah maju. Kontrol dan variasi gerakan dari karakter tentara tidak sevariatif dari *game third person shooter* sekarang ini seperti tidak ada gerakan berlari dan menunduk, dan juga segi desain grafis tidak terlalu istimewa.

2. *The Walking Dead Social Game*

The Walking Dead Social Game dibangun oleh *Eyes Wide Game* mengambil alur cerita dari *film*nya yaitu *The Walking Dead*. Dari segi *gameplay game* ini menawarkan *formula* yang menarik, dan *genre* dari *game* ini adalah *tactical* RPG (*Role Playing Game*) dimana pengguna dapat mengendalikan seorang *survivor* yang *avatar*nya bisa dikreasikan sendiri.

Misi-misi dalam *game* ini dibagi menjadi tiga, mencari sumber daya, bertahan hidup dari serangan *Walkers* (sebutan *zombie* di dalam cerita), atau keduanya digabungkan dalam satu waktu sekaligus. Di dalam misi, ada banyak

hal yang bisa dilakukan, mulai bergerak, bersembunyi dari pandangan *Walkers*, berinteraksi dengan objek, hingga menyerang *Walkers*.



Gambar III. 3 Tampilan *Scene The Walking Dead Social Game*

Fitur yang menarik dari *game* ini adalah, pengguna bisa merekrut teman di dalam *facebook*nya sebagai *bodyguard* untuk menjaga markas, dan teman tersebut juga bisa menjadi musuh atau *walkers* apabila terkena gigitan. *Game* ini mempunyai desain grafis yang berbentuk kartun dan penuh warna.

3. Game yang Akan Dibangun

Game The Walker yang akan dibangun yaitu berupa *game* 3D yang ber-*genre action. Game* ini mempunyai alur cerita yang sama dengan *film The Walking Dead.* Dimana tokoh utamanya berupa seorang polisi yang berusaha menyelamatkan keluarga dan kelompoknya dari serangan para laba-laba.

Game dibangun dengan konsep sudut pandang orang ketiga atau *third person shooter* agar terlihat lebih nyata sesuai dengan *film*nya. Berikut adalah beberapa *fitur* yang ditawarkan di dalam *game* ini:

1). Sistem *single player.*
2). Mengangkat tema atau alur cerita dari *film.*
3). Desain grafis *game* 3D.
4). *Genre game* bertipe *action* dengan konsep sudut pandang orang ketiga atau *third person shooter.*

Secara garis besar perbandingan *game* sejenis diatas dijelaskan dengan tabel dibawah ini.

Tabel III. 1 Perbandingan *Game* Sejenis

	Game Army Men 3D	***The Walking Dead Social Game***	***Game yang Akan Dibangun***
Genre Game	*Action, third person shooter*	*Role playing game, tactical*	*Action, third person shooter*
Desain Grafis	3D tetapi warna, control, dan animasi masih kurang variatif.	3D tetapi lebih terkesan kartun dan penuh warna.	3D dukungan desain grafis dengan komposisi warna lebih banyak dan animasi lebih variatif seperti berjalan, berlari, menunduk, menembak, dan lain-lain.
Storyline* dan *Gameplay	Alur cerita yang unik dan orisinil.	Alur cerita mengambil dari *film* tetapi karakter karakter yang terlibat tidak sesuai dengan	Alur cerita disesuaikan dengan *film*nya.

		*film*nya	
Basis	*Desktop*	*Web*	*Desktop*

III.1.2 Analisis *Game* yang Akan Dibangun

III.1.2.1 *Storyline*

Game yang dibangun *single player* atau dengan satu pemain yang menjalankan seorang karakter utama berupa seorang polisi yang bernama Rick. Untuk lebih meningkatkan dampak positif yang akan dihasilkan kepada penggunanya, *game* ini akan lebih menonjolkan sisi *humanism* dan solidaritasnya kepada sesama manusia, dibandingkan dengan sisi tembak-menembak atau membasmi laba-laba. Jadi, alur cerita ataupun misi yang dibangun bukan berupa membasmi para laba-laba melainkan menghindarinya dan seminim mungkin menggunakan *weapon*/senjata.

Permainan dimulai dengan cerita dimana Rick sang karakter utama akan terbangun dari rumah sakit setelah tertidur lama akibat dari penyakit yang di deritanya. Rick yang tidak mengetahui dunia sudah berubah dengan adanya invasi dari laba-laba raksasa di luar sana akan berjalan keluar dari rumah sakit tanpa memiliki *equipment* atau *weapon* satu pun, disini karakter utama dituntut untuk melarikan diri dari serangan laba-laba yang ditemuinya dan menemukan semua karakter teman atau manusia yang ada di dalam misi atau *objective*.

Karakter utama akan mendapatkan *equipment* dan *weapon* setelah bertemu dengan karakter *human/*manusia (karakter teman) dalam usahanya untuk melarikan diri dari laba-laba.

Diakhir *level* karakter utama dituntut untuk melindungi karakter teman yang sudah ditemui dan membawanya ke tempat yang aman.

III.1.2.2 *Gameplay*

Supaya *game* yang akan dibangun dapat terlihat lebih nyata sesuai dengan *film* nya, maka *game* akan dibangun dengan konsep sudut pandang orang ketiga

atau lebih dikenal dengan *third-person shooters* (TPS), dimana animasi dari karakter akan terlihat lebih jelas dengan sudut pandang ini.

Tugas utama pemain dalam memainkan *game* ini adalah menghindari serangan laba-laba, mencari senjata atau *weapons,* dan menemukan keluarganya lalu menyelamatkan atau membawa ke tempat yang aman.

Pertama pemain akan memasuki *menu* utama yang berisi opsi-opsi seperti *start game, how to play,* dan *credits.* Dengan penjelasannya sebagai berikut.

1). *Menu start game* berfungsi untuk memulai *game* dari awal.
2). *Menu how to play* berfungsi untuk memberikan informasi tentang aturan-aturan yang diterapkan di dalam *game.*
3). *Menu credits* berfungsi untuk memberikan informasi tentang pembangun game.

Selanjutnya di dalam *game* pemain dituntut untuk menyelesaikan misi yang ada di dalam *menu* misi dan *objective.* Misi yang ada di *level* pertama adalah menghindari serangan laba-laba, untuk *level* kedua mencari teman-temannya untuk mendapatkan senjata, sedangkan misi untuk *level* ketiga adalah membawa teman-temannya ke tempat yang aman. Tersedia juga menu *health* yang berguna untuk parameter darah/*health* untuk informasi tentang kondisi darah yang dimiliki oleh pemain. Untuk menambah *fitur* disediakan juga *fitur health* yang berfungsi menambahkan darah kepada karakter utama yang bisa diambil pada setiap *level* di dalam *game.*

III.1.2.3 Deskripsi *Level*

Tingkat kesulitan dari tiap *level* akan dibedakan menurut misi yang harus diselesaikan, juga bagaimana cara menghindari laba-laba yang nantinya akan lebih banyak dari *level* ke *level.*

Untuk *level* akan dibangun tiga *level*, *level* pertama adalah karakter utama akan menghindari serangan laba-laba dan berlari menuju satu buah gudang atau *warehouse* untuk bertemu satu orang karakter teman yang akan membantunya

membunuh laba-laba dengan *sentry gun* yang sudah dipasang sebelumnya, dan karakter utama akan mendapatkan senjata atau *weapons*, *level* kedua mengamankan *warehouse* dari semua laba-laba dengan cara membunuh semua laba-laba. Dan *level* ketiga adalah karakter utama akan menemukan satu orang karakter teman lagi atau keluarganya dan menyelamatkannya dengan menuju ke tempat yang aman yaitu *warehouse* yang sudah diamankan di *level* 2.

Goal dari level pertama adalah, karakter utama akan berlari menghindari laba-laba menuju *target point* yang terletak di ujung dari *map*. *Target point* berbentuk bundar dengan cahaya putih yang menandakan akan melanjutkan ke *map level* kedua. *Target point* terletak di depan sebuah *warehouse* dan karakter teman. *Goal* untuk *level* kedua, karakter utama akan mengamankan *warehouse* dengan cara membasmi semua laba-laba yang ada didalamnya, gudang atau *warehouse* ini nantinya akan dipakai untuk mengamankan karakter keluarganya di *level* ketiga. *Goal* untuk *level* ketiga, karakter utama keluar dari *warehouse* dan mencari satu lagi karakter keluarganya.

Perbedaan disetiap *level* akan dibedakan berdasarkan jumlah *enemy* atau laba-laba yang akan dihadapi. Untuk *level* pertama laba-laba akan keluar dalam jumlah 10 dan karakter utamanya tidak memiliki senjata. Untuk *level* kedua laba-laba akan keluar dalam jumlah 20 dan di dalam ruangan atau di dalam *warehouse*, dan karakter memiliki senjata untuk membunuh semua laba-laba. Untuk *level* ketiga laba-laba akan keluar dalam jumlah 30, dan karakter utama dituntut untuk menemukan satu orang karakter keluarganya lalu melarikan diri kembali ke *warehouse* yaitu tempat yang lebih aman.

Deskripsi *level* lebih jelasnya pada tabel berikut:

Tabel III. 2 Tabel Deskripsi *Level*

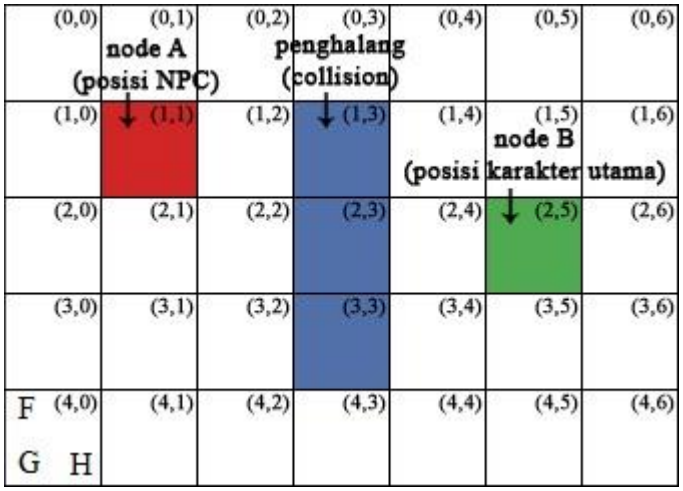
Level	**Karakter**	***Weapons***	**Lokasi**	**Jumlah *Enemy***
1	Satu karakter utama dan satu karakter	Tidak	*Outdoor*	10

	teman			
2	Satu karakter utama	Ya	*Indoor (warehouse)*	20
3	Satu karakter utama dan satu karakter keluarganya	Ya	*Outdoor*	30

III.2 Analisis AI

III.2.1 Analisis Algoritma A* (A *Star*)

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



Gambar III. 4 Ilustrasi A*

Pada gambar III.4 diasumsikan bahwa wilayah pencarian dibagi-bagi menjadi kotak-kotak atau disebut *node*, *node* yang berwarna merah atau node (1,1) merupakan *node* A (posisi NPC) dan *node* yang berwarna hijau atau

node (2,5) merupakan *node* B (posisi karakter utama), sedangkan *node* yang berwarna biru {(1,3), (2,3), (3,3)} merupakan dinding penghalang atau disebut *collision*. Diasumsikan pencarian jalan dimulai dari *node* A menuju *node* B.

Nilai F yang akan dihitung untuk setiap *node* terletak disudut kiri atas kotak, nilai G terletak di sudut kiri bawah kotak dan nilai H terletak disudut kanan bawah kotak.

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))

Inisialisasi nilai *graph* dan nilai *matrix* untuk menentukan posisi koordinat dari node-node yang digunakan. Dan inisialisasi awal untuk *open list* dan *closed list* berjumlah kosong atau *null*.

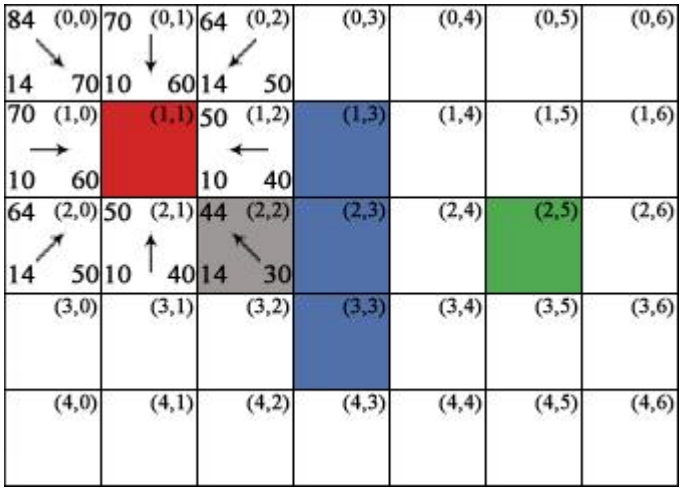
Selanjutnya adalah melakukan perhitungan untuk menemukan jalan dengan cara menghitung *node-node* yang berdekatan dari *node* A sampai *node* B. Langkah-langkahnya adalah sebagai berikut:

1. Pertama, dimulai dari *node* A dan tambahkan ke dalam *open list. Open list* merupakan *list* untuk *node-node* yang harus diperiksa.

2. Ambil *node* A sebagai *current node*, dan *set current node = best node*. Atau dalam artian *node* A merupakan bagian dari jalan terpendek yang sudah didapatkan
3. Periksa apakah *current node* sama dengan *node* B, atau *current node* yang sudah didapatkan merupakan *node* tujuan. Jika iya maka jalan sudah ditemukan, jika tidak maka hapus *node* A dari *open list*, dan tambahkan 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.

Untuk lebih jelasnya proses perhitungan dilakukan per-iterasi dan dijelaskan sebagai berikut.

Iterasi pertama:



Gambar III. 5 Ilustrasi Iterasi Pertama.

Pada iterasi 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 nilai F paling kecil sebagai *current node* berikutnya dalam perhitungan diatas didapat *node* (2,2) memiliki nilai F paling rendah dan diberikan tanda dengan kotak berwarna abu-abu.

Node yang menjadi *current node* atau *node* (2,2) diberikan nilai H = 30 yang didapatkan dari perhitungan 3 langkah atau 3 *node* menuju *node* B dan dikalikan 10, 3*10 = 30.

Iterasi pertama menghasilkan *node-node* di dalam *open list* dan *closed list* dijelaskan dengan tabel sebagai berikut.

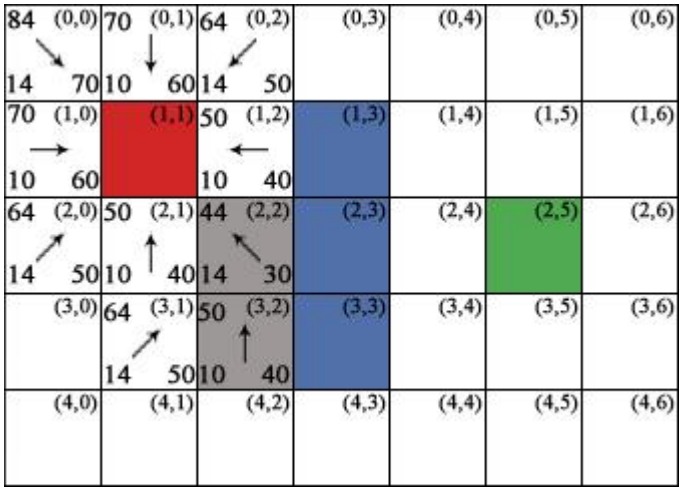
Tabel III. 3 Tabel Perhitungan Iterasi Pertama

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

Proses ini 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* yang *unwalkable* atau dalam contoh ini adalah *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.

Selanjutnya perhitungan pada iterasi kedua dengan *current node* berada pada *node* (2,2).



Gambar III. 6 Ilustrasi Iterasi Kedua

Periksa *neighbor node* atau *node* yang berdekatan dengan *current node*, disebelah kanan dan kanan bawah dari *node* ini adalah *node* penghalang atau *unwalkable* (gambar III.6) berarti diabaikan, dan disebelah kiri atas

adalah *node* (1,1) yang sudah berada di dalam *closed list*, yang berarti juga diabaikan.

Sedangkan *node* lainnya yang berdekatan sudah berada di dalam *open list*, jadi perlu diperiksa apakah *node* tersebut mempunyai jalan yang lebih baik atau tidak dengan menggunakan nilai G sebagai acuannya. Setelah diperiksa *node-node* tersebut ternyata tidak ada jalan yang lebih baik, maka tidak ada yang perlu diubah dan proses untuk *current node* ini selesai.

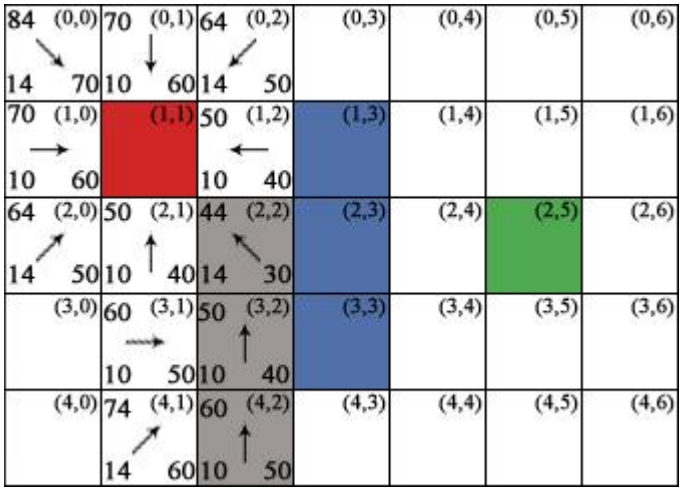
Node (3,2) *node* (2,1) dan *node* (1,2) memiliki nilai F yang sama, dapat dilihat pada gambar III.6 diatas, maka sistem akan memilih *node* (3,2) karena yang terakhir masuk ke dalam *open list*. Yang berarti *current node* selanjutnya adalah *node* (3,2).

Iterasi kedua menghasilkan *node-node* di dalam *open list* dan *closed list* dijelaskan dengan tabel sebagai berikut.

Tabel III. 4 Tabel Perhitungan Iterasi Kedua

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

Selanjutnya perhitungan pada iterasi ketiga dengan *current node* berada pada *node* (3,2).



Gambar III. 7 Ilustrasi Iterasi Ketiga

Pada gambar III.7 diatas *node* (4,3) tidak dihitung atau tidak masuk ke dalam *open list,* karena karakter tidak akan bisa bergerak secara *diagonal* menembus dinding atau *collision* sehingga *node* (4,3) otomatis menjadi *unwalkable*.

Looping lagi seperti langkah sebelumnya, hapus dari *open list* dan tambahkan ke dalam *closed list,* periksa semua *node* yang berdekatan, masukkan ke dalam *open list* dan tambahkan *parent*nya, dan periksa lagi jalan terbaik dan *parent*nya. Pada *node* (3,1) *parent*nya mengalami perubahan karena setelah diperiksa nilai G yang dihasilkan dari *current node* lebih kecil, maka nilai F nya juga dirubah.

Nilai F pada *node* (3,1) juga sama dengan *node* (4,2), sama seperti contoh kasus sebelumnya maka *node* (4,2) dipilih sebagai *current node.*

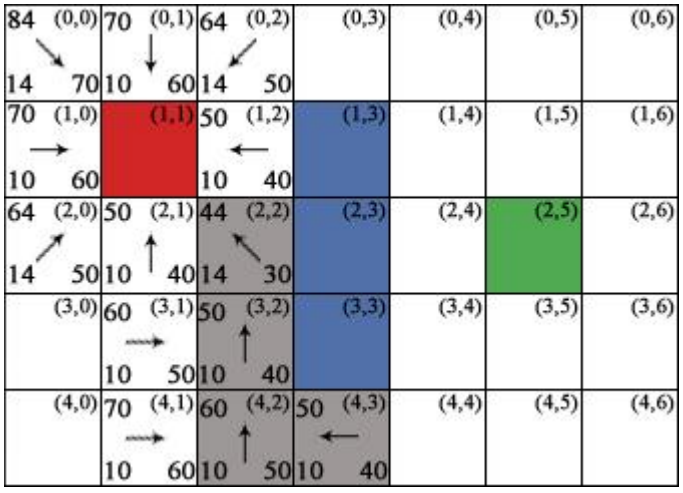
Iterasi ketiga menghasilkan *node-node* di dalam *open list* dan *closed list* dijelaskan dengan tabel sebagai berikut.

Tabel III. 5 Tabel Perhitungan Iterasi Ketiga

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

(0,1)	(1,1)	(2,2)	
(0,2)	(1,1)	(3,2)	
(1,0)	(1,1)		
(1,2)	(1,1)		
(2,0)	(1,1)		
(2,1)	(1,1)		
(3,1)	(3,2)		
(4,1)	(3,2)		
(4,2)	(3,2)		

Selanjutnya perhitungan pada iterasi keempat dengan *current node* berada pada *node* (4,2).



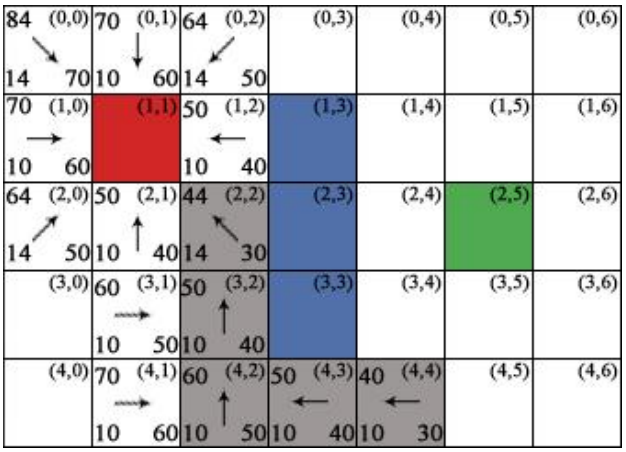
Gambar III. 8 Ilustrasi Iterasi Keempat

Iterasi keempat menghasilkan *node-node* di dalam *open list* dan *closed list* dijelaskan dengan tabel sebagai berikut.

Tabel III. 6 Tabel Perhitungan Iterasi Keempat

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

Perhitungan pada iterasi kelima dengan *current node* berada pada *node* (4,3).



Gambar III. 9 Ilustrasi Iterasi Kelima

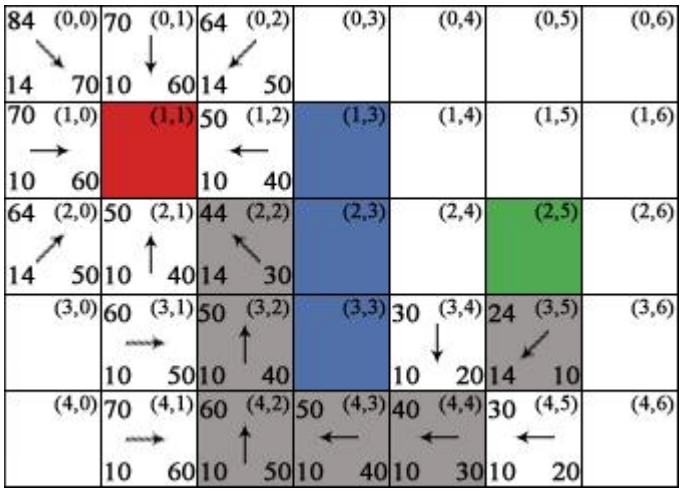
Node (3,4) menjadi *unwalkable* karena karakter tidak bisa langsung bergerak ke *node* tersebut, sehingga tidak masuk ke dalam *open list*.

Iterasi kelima menghasilkan *node-node* di dalam *open list* dan *closed list* dijelaskan dengan tabel sebagai berikut.

Tabel III. 7 Tabel Perhitungan Iterasi Kelima

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

Perhitungan pada iterasi keenam dengan *current node* berada pada *node* (4,4).



Gambar III. 10 Ilustrasi Iterasi Keenam

Iterasi keenam menghasilkan *node-node* di dalam *open list* dan *closed list* dijelaskan dengan tabel sebagai berikut.

Tabel III. 8 Tabel Perhitungan Iterasi Keenam

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

Pada iterasi ketujuh sudah dapat dipastikan *target point* masuk ke dalam *closed list* (gambar III.9) maka proses perhitungan selesai.

Penentuan jalan terbaik dengan melakukan pemeriksaan dimulai dari *target point* atau *node* (2,5) dengan mengikuti arah *pointer* yang berarti akan membawanya kembali ke *node* awal, maka akan didapatkan jalan terbaiknya.

Dari perhitungan yang dilakukan diatas dapat diambil kesimpulan fungsi heuristik yang digunakan yaitu fungsi *Manhattan* merupakan *unadmissible heuristic* atau tidak menjamin jalan yang ditemukan adalah jalan yang terbaik. Dalam contoh kasus perhitungan diatas, misalkan H*(n) merupakan *cost* sebenarnya dari simpul *n* ke simpul tujuan, maka pada algoritma A* terdapat beberapa kemungkinan yang terjadi, yaitu:

1). Jika H(n) = 0, sehingga hanya G(n) yang digunakan, maka A* akan bekerja seperti halnya algoritma Djikstra.

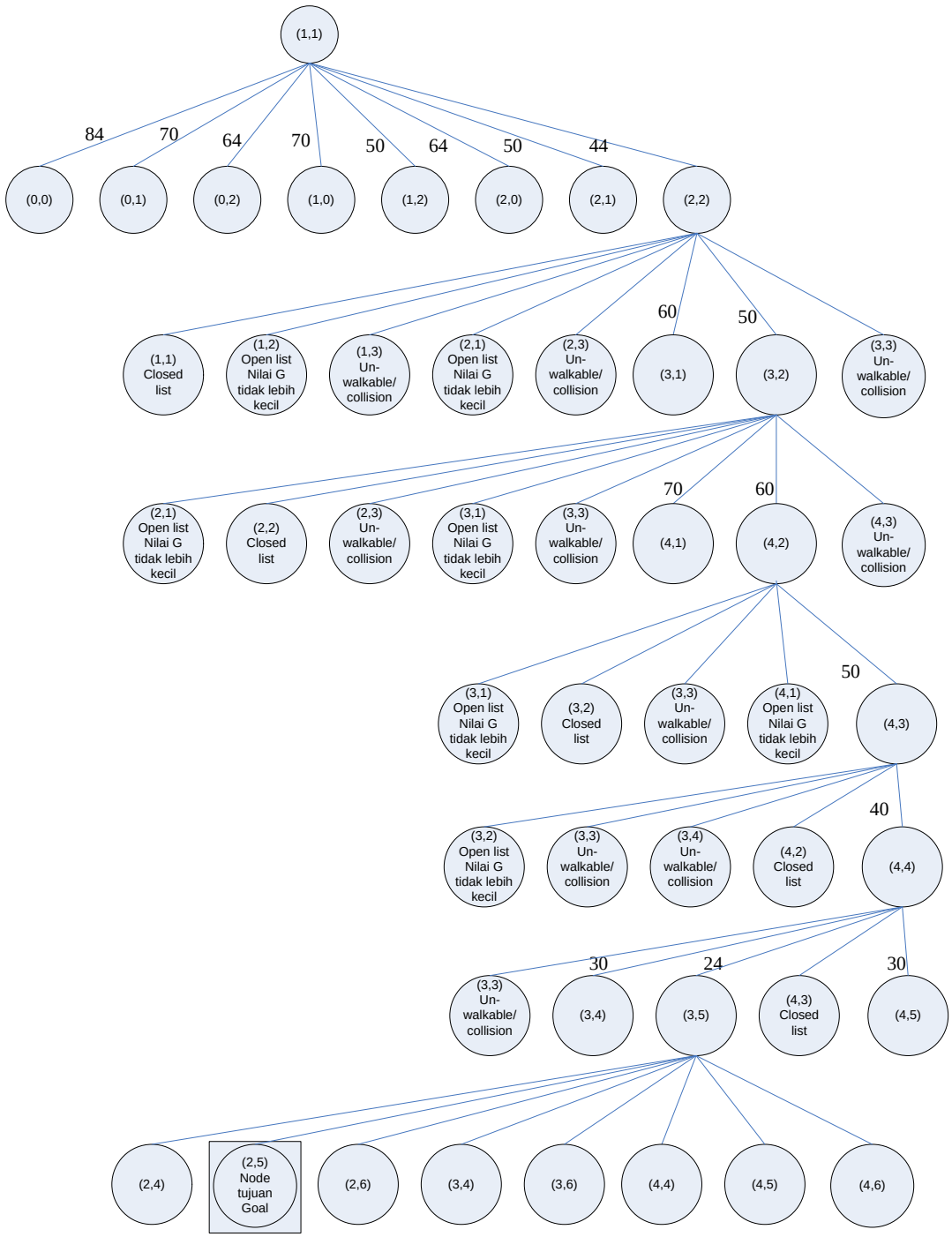
2). Jika H(n) ≤ H*(n), maka A* akan mengembangkan *node* dengan nilai paling rendah dan algoritma A* menjamin ditemukannya jalan terbaik. Nilai H(n) terendah akan membuat algoritma mengembangkan lebih banyak *node*. Jika H(n) ≤ H*(n), maka H(n) dikatakan heuristik yang *admissible*.

3). Jika H(n) = H*(n), maka A* akan mengikuti jalan terbaik dan tidak akan mengembangkan *node-node* yang lain sehingga akan berjalan cepat. Tetapi hal ini tidak akan terjadi pada semua kasus. Informasi yang baik akan mempercepat kinerja A*.

4). Jika H(n) ≥ H*(n), maka A* tidak menjamin pencarian rute terpendek, tetapi berjalan dengan cepat.

5). Jika H(n) terlalu tinggi *relative* dengan G(n) sehingga hanya H(n) yang bekerja, maka A* berubah jadi *Greedy Best First Search*.

Perhitungan langkah diatas digambarkan dengan ilustrasi *tree* sebagai berikut.



Gambar III. 11 Ilustrasi *Tree* Pencarian Jalan Terbaik

Analisis dari algoritma A* sebagai penentuan jalan terbaik dapat dijelaskan dengan *pseudocode* di dalam *game* yang dibangun, melalui tabel dibawah ini.

Tabel III. 9 Tabel analisis algoritma A* *pathfinding*

Langkah	**Kondisi**	***Pseudocode***
1.	Inisialisasi *graph*	System type GraphType Graph[i].Name == type Return graph[i] Return null
2.	Inisialisasi matrix	Matrix translate = X, Y, Z Matrix[] = X; Matrix[] = Y; Matrix[] = Z; Return;
3.	Inisialisasi *open list* dan *closed list.*	OPEN = *null* && CLOSED = *null*
4.	Masukkan *node* ke *open list.*	OPEN = priority queue containing START
5.	Ambil *node* dengan nilai F terendah dari *open list*, dan selanjutnya disebut sebagai *current node.* Hapus dari *open list.*	while lowest rank in OPEN is not the GOAL: current = remove lowest rank item from OPEN
6.	Masukkan *current node* ke *closed list.*	add current to CLOSED
7.	Untuk seluruh *node* yang terhubung langsung dengan *current node*, lakukan: a. Jika *node* yang diperiksa sudah termasuk dalam *closed list,*	for neighbors of current: cost = g(current) + movementcost(current, neighbor) if neighbor in OPEN and cost less than g(neighbor):

	abaikan.
	b. Jika kondisi a tidak terpenuhi dan *node* belum termasuk di dalam *open list*, masukkan *node* tersebut di *open list.*
	c. Jika kondisi b tidak terpenuhi (berarti *node* diperiksa berada pada *open list*), dan nilai G dari *current node* kurang dari nilai G dari *node* awal, ubah *parent* dari *node* awal menjadi *current node.*
		remove neighbor from OPEN, because new path is better
		if neighbor in CLOSED and cost less than g(neighbor):
		remove neighbor from CLOSED
		if neighbor not in OPEN and neighbor not in CLOSED:
		set g(neighbor) to cost
		add neighbor to OPEN
		set priority queue rank to g (neighbor) + h(neighbor)
		set neighbor's parent to current

8.	Ulangi langkah nomor 3 sampai 6 jika kondisi dibawah ini tidak terpenuhi: a. *Node* tujuan sudah berada di dalam *closed list*, atau b. *Open list* tidak berisi apa-apa (tidak ditemukan jalan menuju *node* tujuan).	stop looping if current = GOAL OPEN = null
9.	Telusuri *parent* dari setiap *node* dimulai dari *node* tujuan apabila jalan ditemukan.	reconstruct reverse path from goal to start by following parent pointers

III.2.2 Analisis Algoritma *Fuzzy Logic*

Fuzzy logic di dalam pembangunan *game* ini digunakan untuk menentukan variasi perilaku yang akan dihasilkan oleh NPC (*non playable character*). Dengan adanya *fuzzy logic*, masing-masing NPC dapat merubah

perilaku berbasis perubahan variabel masukan menjadi perilaku yang akan dirancang menggunakan *fuzzy logic*.

Metode *fuzzy* yang digunakan adalah metode Sugeno, karena metode ini menghasilkan keluaran yang berupa konstanta tegas, sehingga dapat mewakili nilai perilaku yang sudah dirancang sebelumnya.

Analisis *fuzzy logic* yang akan dilakukan menggunakan aplikasi *MATLAB* yang bertujuan untuk lebih mudah dalam perhitungan menentukan variasi perilaku NPC, tahap-tahapnya adalah sebagai berikut.

1. **Variabel *fuzzy***

   Terdapat tiga variabel yang digunakan yaitu variabel jarak dan variabel kekuatan (darah/*health*) sebagai variabel *input*, variabel perilaku NPC sebagai variabel *output*.

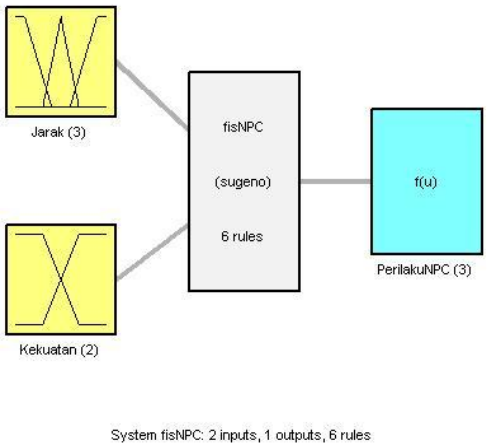
2. **Nilai linguistik / *terma***

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

   1). Jarak: Dekat, Sedang, Jauh
   2). Kekuatan: Lemah, Kuat
   3). Perilaku NPC: Melarikan diri, Bertahan, Menyerang, Menyerang Brutal

3. **Fuzzyfikasi**

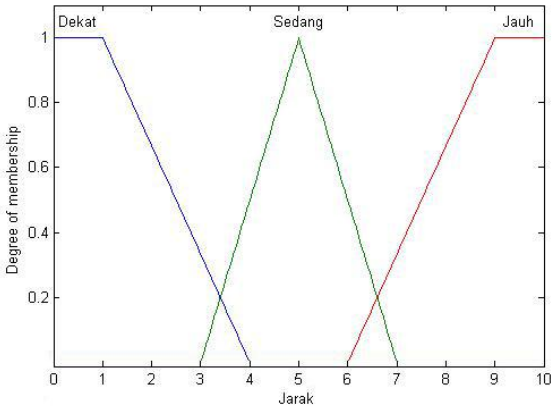
   Fuzzyfikasi merupakan proses memetakan nilai *crisp* (numerik) ke dalam himpunan *fuzzy* dan menentukan derajat keanggotaannya. Secara garis besar pemetaan nilai *crisp* ke dalam himpunan *fuzzy* dijelaskan dengan gambar sebagai berikut.



Gambar III. 12 Fuzzyfikasi

Fuzzyfikasi untuk setiap variabelnya digambarkan sebagai berikut.

a. Variabel Jarak



Gambar III. 13 Fuzzyfikasi Variabel Jarak

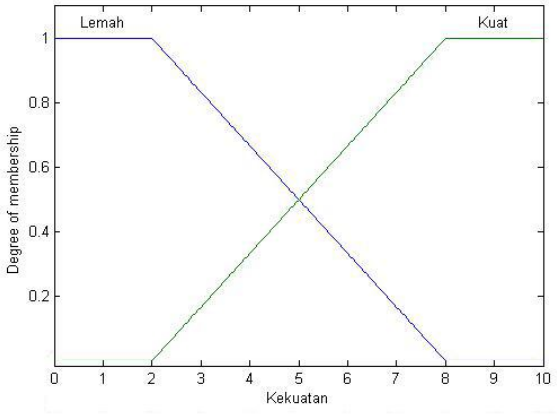
Range nilai untuk variabel jarak diantara 0 – 10 dengan penjelasan setiap *terma* sebagai berikut:

1). Range nilai untuk *terma* dekat antara 0 – 4

2). Range nilai untuk *terma* sedang antara 3 – 7

3). Range nilai untuk *terma* jauh antara 6 – 10

b. Variabel Kekuatan



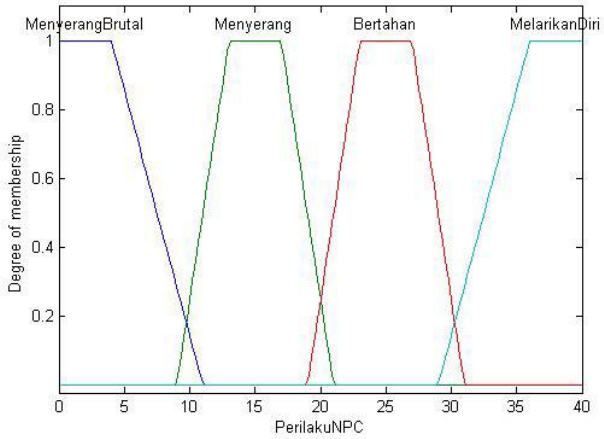
Gambar III. 14 Fuzzyfikasi Variabel Kekuatan

Range nilai untuk variabel kekuatan diantara 0 – 10 dengan penjelasan setiap *terma* sebagai berikut:

1). Range nilai untuk *terma* lemah antara 0 – 8

2). Range nilai untuk *terma* sedang antara 2 – 10

c. Variabel Perilaku NPC



Gambar III. 15 Fuzzyfikasi Variabel Perilaku NPC

Range nilai untuk variabel perilaku NPC diantara 0 – 30 dengan penjelasan setiap *terma* sebagai berikut:

1). Range nilai untuk *terma* menyerang brutal antara 0 – 11

2). Range nilai untuk *terma* menyerang antara 9 – 21

3). Range nilai untuk *terma* bertahan antara 19 – 31

4). Range nilai untuk *terma* melarikan diri antara 29 – 40

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 (PerilakuNPC is MenyerangBrutal)

2). If (Jarak is Dekat) and (Kekuatan is Kuat) then (PerilakuNPC is Menyerang)

3). If (Jarak is Sedang) and (Kekuatan is Lemah) then (PerilakuNPC is MelarikanDiri)

4). If (Jarak is Sedang) and (Kekuatan is Kuat) then (PerilakuNPC is Menyerang)

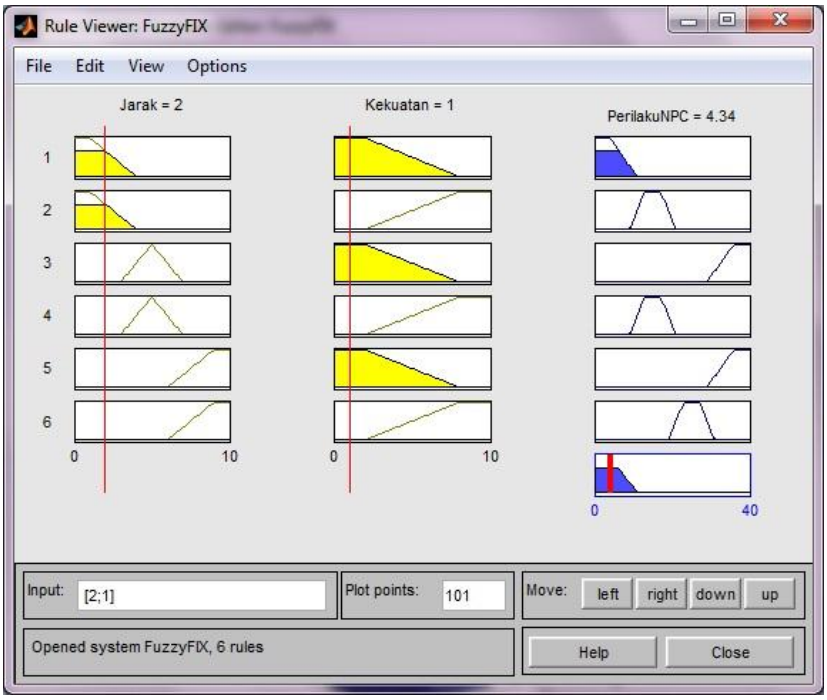
5). If (Jarak is Jauh) and (Kekuatan is Lemah) then (PerilakuNPC is MelarikanDiri)

6). If (Jarak is Jauh) and (Kekuatan is Kuat) then (PerilakuNPC 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.*

Misalkan variabel jarak diberikan nilai *crisp* sebesar 1, dan variabel kekuatan diberikan nilai sebesar 9, maka proses agregasi yang menentukan keluaran sesuai dengan *rules* yang sudah dibuat akan dijelaskan seperti gambar berikut.



Gambar III. 16 Proses Agregasi dan Defuzzyfikasi

Gambar diatas memperlihatkan proses agregasi dengan hitungan menggunakan aplikasi MATLAB mencakup kombinasi dari enam *rules* yang sudah dibuat, dan proses defuzzyfikasi pada gambar diatas menghasilkan nilai 4.34 yang berarti perilaku NPC yang dihasilkan adalah "MenyerangBrutal" atau sesuai dengan *rules* nomer 1 yaitu "IF (Jarak is Dekat) AND (Kekuatan is Lemah) THEN (PerilakuNPC is MenyerangBrutal)".

6. Pengujian

Pengujian nilai-nilai yang dimasukkan ke dalam variabel input (jarak dan kekuatan) dan nilai yang dihasilkan oleh variabel output (perilaku NPC) dijelaskan dengan tabel sebagai berikut.

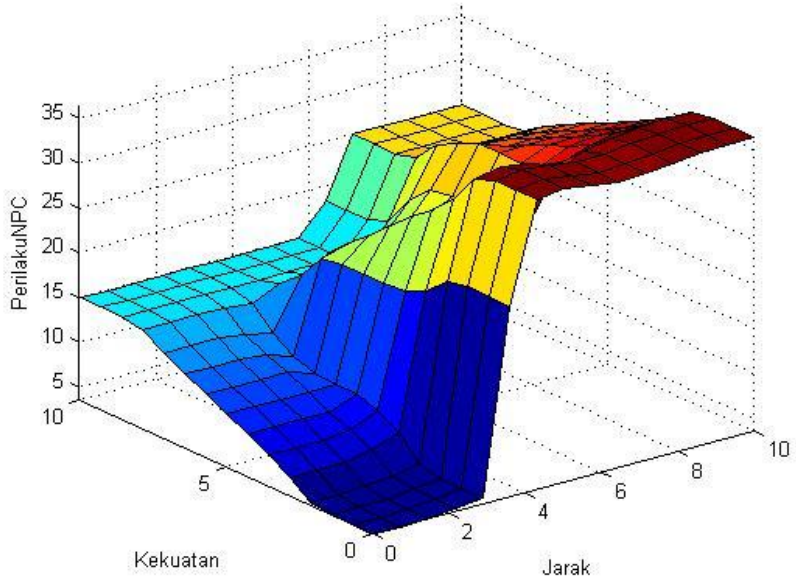
Tabel III. 10 Nilai Keluaran Dalam Proses Pengujian

		Kekuatan									
Jarak		**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
	1	3.92	3.92	6.38	8.31	9.94	11.5	13.1	15	15	15
	2	4.34	4.34	6.85	8.31	9.94	11.5	12.9	15	15	15
	3	4.84	4.84	8.46	10.1	10.1	10.1	11.8	15	15	15
	4	35.4	35.4	29.6	26.6	24.9	23.1	20.2	15	15	15
	5	36.1	36.1	31.4	27.9	24.9	22	18.9	15	15	15
	6	35.4	35.4	29.6	26.6	24.9	23.1	20.2	15	15	15
	7	35.2	35.2	31.5	29.9	29.9	29.9	28.2	25	25	25
	8	35.7	35.7	33.1	31.7	30.1	28.5	27.1	25	25	25
	9	36.1	36.1	33.6	31.7	30.1	28.5	26.9	25	25	25
	10	36.1	36.1	33.6	31.7	30.1	28.5	26.9	25	25	25

Penjelasan dari tabel nilai keluaran dalam proses pengujian adalah sebagai berikut:

a. Tabel dengan *shading* warna (merah) menunjukkan *range* nilai untuk *terma* menyerang brutal.
b. Tabel dengan *shading* warna (orange) menunjukkan *range* nilai untuk *terma* menyerang.
c. Tabel dengan *shading* warna (hijau) menunjukkan *range* nilai untuk *terma* bertahan.
d. Tabel dengan *shading* warna (biru) menunjukkan *range* nilai untuk *terma* melarikan diri.

Atau secara *surface* hasil dari nilai keluaran dapat digambarkan sebagai berikut:



Gambar III. 17 *Surface* Proses Fuzzyfikasi

III.3 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.

III.3.1 Analisis Kebutuhan Perangkat Lunak

Game action 3D *The Walker* memiliki kebutuhan perangkat lunak sebagai berikut:

1. Aplikasi *game* yang dihasilkan berbasis *desktop*.
2. Pemodelan dan perancangan sistem dalam penulisan laporan ini menggunakan pemodelan berbasis objek dengan *tools* UML (*Unified Modeling Language)*.
3. Dikembangkan dengan menggunakan *tools game engine Unity 3D,* dengan bahasa pemrograman berupa C#, *Javascript*, dan *Boo.*
4. *Rendering* gambar 3D menggunakan *3ds Max,* dan gambar 2D menggunakan *Adobe Photoshop CS3*, sedangkan suara/*sound effect* menggunakan *Adobe After Effects.*

Dari hasil pengamatan perangkat lunak, terdapat beberapa hal penting yang harus diperhatikan agar sistem yang akan dirancang dapat berjalan dengan baik dan sesuai dengan yang diharapkan. Pengguna diharapkan dapat memenuhi kebutuhan perangkat lunak yang telah direkomendasikan agar aplikasi dapat berjalan dengan baik.

Perangkat lunak yang dibutuhkan pengguna untuk memainkan *game* ini adalah :

1. Sistem operasi minimal *Windows Xp*
2. DirectX 9

III.3.2 Analisis Kebutuhan Perangkat Keras

Agar aplikasi dapat berjalan dengan baik, maka dibutuhkan perangkat keras yang sesuai dengan kebutuhan aplikasi. Spesifikasi minimum perangkat keras bagi pengembang untuk membangun *game action* 3D *The Walker* ini adalah sebagai berikut:

1. Prosesor yang setara dengan kecepatan 2.0 GHz.
2. RAM 2 GB.

3. Harddisk dengan ruang kosong 3 GB.
4. VGA Card shared 128 MB.
5. Monitor dengan resolusi 1024 x 768 pixels.
6. *Speaker*.
7. *Keyboard* dan *mouse.*

Spesifikasi minimum perangkat keras bagi pengguna untuk memainkan *game action* 3D *The Walker* ini adalah sebagai berikut:

1. Prosesor Dual Core dengan kecepatan 1.8 GHz.
2. RAM 1 GB.
3. Harddisk dengan ruang kosong 500 MB.
4. VGA Card 512 MB.
5. Monitor dengan resolusi 1024 x 768 pixels.
6. *Speaker*.
7. *Keyboard* dan *mouse.*

III.3.3 Analisis Pengguna

Adapun kriteria pengguna yang dapat menggunakan *game* ini adalah sebagai berikut:

1. Pengguna berumur 17 tahun keatas, dengan artian pengguna bisa membedakan dampak positif dan dampak negatif yang ditimbulkan dari bermain *game*.
2. Pengguna tidak buta warna.
3. Pengguna memiliki kemampuan bahasa inggris.
4. Pengguna tidak cacat kedua tangan dengan artian bisa menggunakan *mouse* dan *keyboard*.
5. Digunakan satu pengguna saat bermain game *(single player)*.

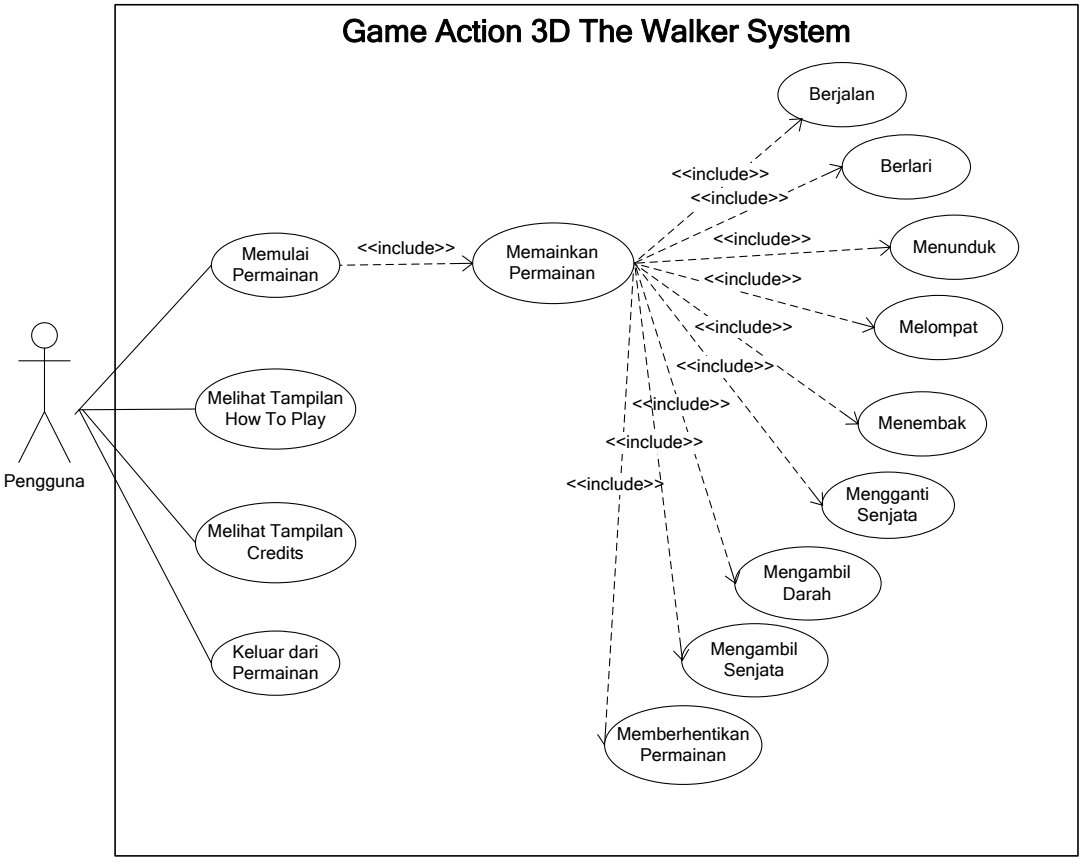
III.4 Analisis Kebutuhan Fungsional

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 action* 3D *The Walker* antara lain *Use Case Diagram, Class Diagram, Sequence Diagram, Activity Diagram.*

III.4.1 *Use Case Diagram*

Berikut ini adalah perancangan proses-proses yang terdapat pada *game action* 3D *The Walker*, yang digambarkan dengan *Use Case Diagram* yang dapat dilihat pada gambar III.18



Gambar III. 18 *Use Case Diagram*

1. *Definisi Actor*

Definisi *Actor* berfungsi untuk menjelaskan *actor* yang terdapat pada *use case diagram*. Definisi *actor* diterangkan pada tabel III.11 berikut.

Tabel III. 11 *Definisi Actor*

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

2. *Definisi Use Case*

Definisi *use case* berfungsi untuk menjelaskan fungsi *use case* yang terdapat pada *use case diagram*. Definisi *use case* diterangkan pada tabel III.12

Tabel III. 12 *Definisi Use Case*

No.	*Use Case*	Deskripsi
1.	Memulai Permainan	Proses untuk memulai permainan.
2.	Melihat Tampilan *How To Play*	Proses untuk menampilkan petunjuk dan panduan bermain.
3.	Melihat Tampilan *Credits*	Proses untuk melihat informasi pembuat *game.*
4.	Keluar dari Permainan	Proses untuk keluar dari *game.*
5.	Memainkan Permainan	Proses permainan sedang berjalan.
6.	Berjalan	Proses untuk menggerakkan karakter dengan animasi berjalan.
7.	Berlari	Proses untuk menggerakkan karakter dengan animasi berlari.
8.	Menunduk	Proses untuk menggerakkan karakter dengan animasi menunduk.
9.	Melompat	Proses untuk menggerakkan karakter dengan animasi melompat.
10.	Menembak	Proses untuk menggerakkan karakter dengan animasi

		menembak
11.	Mengganti Senjata	Proses untuk menggerakkan karakter dengan animasi mengganti senjata
12.	Mengambil Darah	Proses untuk menambah darah untuk karakter.
13.	Mengambil Senjata	Proses untuk mengambil senjata untuk karakter.
14.	Memberhentikan Permainan	Proses untuk memberhentikan permainan untuk sementara.

III.4.2 *Scenario Use Case*

Skenario setiap bagian pada *use case* menunjukkan proses apa yang terjadi pada setiap bagian didalam *use case* tersebut, dimana pengguna memberikan perintah pada setiap bagian dan respon apa yang diberikan oleh sistem kepada pengguna setelah pengguna memberikan perintah pada setiap bagian - bagian *use case*.

Tabel III. 13 *Scenario Use Case* Memulai Permainan

Identifikasi	
Nomor	1
Nama	Memulai Permainan
Tujuan	Memulai permainan
Deskripsi	
Tipe	*Primary Actor*
Aktor	Pengguna
Skenario Utama	
Kondisi awal	Pengguna berada di *menu* utama
Aksi Aktor	Reaksi Sistem
1. Pengguna memilih menu *start*	2. Sistem menjalankan tampilan *scene video*
3. Pengguna memainkan permainan *level* 1	4. Sistem menjalankan permainan *level* 1

Kondisi Akhir	*Level* 1 dimainkan

Tabel III. 14 *Scenario Use Case* Melihat Tampilan *How To Play*

Identifikasi	
Nomor	2
Nama	Melihat Tampilan *How To Play*
Tujuan	Menampilkan petunjuk dan panduan bermain
Deskripsi	
Tipe	*Primary Actor*
Aktor	Pengguna
Skenario Utama	
Kondisi awal	Pengguna berada di *menu* utama
Aksi Aktor	**Reaksi Sistem**
1. Pengguna memilih menu *how to play*	2. Sistem menampilkan halaman petunjuk bermain
Kondisi Akhir	Menampilkan halaman petunjuk dan panduan bermain

Tabel III. 15 *Scenario Use Case* Melihat Tampilan *Credits*

Identifikasi	
Nomor	3
Nama	Melihat Tampilan *Credits*
Tujuan	Menampilkan informasi pembuat *game*
Deskripsi	
Tipe	*Primary Actor*
Aktor	Pengguna
Skenario Utama	
Kondisi awal	Pengguna berada di *menu* utama

Aksi Aktor	Reaksi Sistem
1. Pengguna memilih menu *credits*	2. Sistem menampilkan halaman informasi pembuat *game.*
Kondisi Akhir	Menampilkan halaman informasi pembuat *game.*

Tabel III. 16 *Scenario Use Case* Keluar Dari Permainan

Identifikasi	
Nomor	4
Nama	Keluar dari Permainan
Tujuan	Keluar dari permainan.
Deskripsi	
Tipe	*Primary Actor*
Aktor	Pengguna
Skenario Utama	
Kondisi awal	Pengguna berada di *menu* utama
Aksi Aktor	**Reaksi Sistem**
1. Pengguna memilih menu *exit*	2. Sistem keluar dari permainan.
Kondisi Akhir	Keluar dari permainan.

Tabel III. 17 *Scenario Use Case* Memainkan Permainan

Identifikasi	
Nomor	5
Nama	Memainkan Permainan.
Tujuan	Berada di dalam permainan.
Deskripsi	

Tipe	*Primary Actor*	
Aktor	Pengguna	
Skenario Utama		
Kondisi awal	Pengguna berada di *menu* utama	
Aksi Aktor		**Reaksi Sistem**
1. Pengguna memilih menu *start*		2. Sistem menjalankan tampilan *scene video*
3. Pengguna memainkan permainan		4. Sistem menjalankan permainan *level* 1, *level* 2, dan *level* 3
5. Pengguna berhasil menyelesaikan misi		6. Sistem menampilkan *scene* permainan selesai
7. Pengguna gagal menyelesaikan misi		8. Sistem mengulang menampilkan permainan
Kondisi Akhir	Jika pengguna berhasil menyelesaikan misi, sistem menampilkan *scene* permainan selesai	

Tabel III. 18 *Scenario Use Case* Berjalan

Identifikasi		
Nomor	6	
Nama	Berjalan	
Tujuan	Menggerakkan karakter dengan animasi berjalan.	
Deskripsi		
Tipe	*Primary Actor*	
Aktor	Pengguna	
Skenario Utama		
Kondisi awal	Pengguna memainkan permainan	
Aksi Aktor		**Reaksi Sistem**

1. Pengguna menggerakkan karakter dengan interaksi menggunakan *keyboard* menekan tombol CTRL dan W.	2. Sistem menggerakkan karakter menjalankan animasi berjalan ke depan.
3. Pengguna menggerakkan karakter dengan interaksi menggunakan *keyboard* menekan tombol CTRL dan A.	4. Sistem menggerakkan karakter menjalankan animasi berjalan ke kiri.
5. Pengguna menggerakkan karakter dengan interaksi menggunakan *keyboard* menekan tombol CTRL dan S.	6. Sistem menggerakkan karakter menjalankan animasi berjalan ke belakang.
7. Pengguna menggerakkan karakter dengan interaksi menggunakan *keyboard* menekan tombol CTRL dan D.	8. Sistem menggerakkan karakter menjalankan animasi berjalan ke kanan.
Kondisi Akhir	Sistem menggerakkan karakter menjalankan animasi berjalan ke depan, kiri, belakang, dan ke kanan sesuai dengan inputan pengguna.

Tabel III. 19 *Scenario Use Case* Berlari

Identifikasi	
Nomor	7
Nama	Berlari

<table>
<tr><td>Tujuan</td><td colspan="2">Menggerakkan karakter dengan animasi berlari.</td></tr>
<tr><td colspan="3">Deskripsi</td></tr>
<tr><td>Tipe</td><td colspan="2">Primary Actor</td></tr>
<tr><td>Aktor</td><td colspan="2">Pengguna</td></tr>
<tr><td colspan="3">Skenario Utama</td></tr>
<tr><td>Kondisi awal</td><td colspan="2">Pengguna memainkan game</td></tr>
<tr><td colspan="2">Aksi Aktor</td><td>Reaksi Sistem</td></tr>
<tr><td colspan="2">1. Pengguna menggerakkan karakter dengan interaksi menggunakan keyboard menekan tombol W.</td><td>2. Sistem menggerakkan karakter menjalankan animasi berlari ke depan.</td></tr>
<tr><td colspan="2">3. Pengguna menggerakkan karakter dengan interaksi menggunakan keyboard menekan tombol A.</td><td>4. Sistem menggerakkan karakter menjalankan animasi berlari ke kiri.</td></tr>
<tr><td colspan="2">5. Pengguna menggerakkan karakter dengan interaksi menggunakan keyboard menekan tombol S.</td><td>6. Sistem menggerakkan karakter menjalankan animasi berlari ke belakang.</td></tr>
<tr><td colspan="2">7. Pengguna menggerakkan karakter dengan interaksi menggunakan keyboard menekan tombol D.</td><td>8. Sistem menggerakkan karakter menjalankan animasi berlari ke kanan.</td></tr>
<tr><td colspan="3"></td></tr>
<tr><td>Kondisi Akhir</td><td colspan="2">Sistem menggerakkan karakter menjalankan animasi berlari ke depan, kiri, belakang, dan ke kanan sesuai dengan inputan pengguna.</td></tr>
</table>

Tabel III. 20 *Scenario Use Case* Menunduk

Identifikasi	
Nomor	8
Nama	Menunduk
Tujuan	Menggerakkan karakter dengan animasi menunduk.
Deskripsi	
Tipe	*Primary Actor*
Aktor	Pengguna
Skenario Utama	
Kondisi awal	Pengguna memainkan *game*
Aksi Aktor	**Reaksi Sistem**
1. Pengguna menggerakkan karakter dengan interaksi menggunakan *keyboard* menekan tombol X.	2. Sistem menggerakkan karakter menjalankan animasi menunduk.
Kondisi Akhir	Sistem menggerakkan karakter menjalankan animasi menunduk sesuai inputan pengguna.

Tabel III. 21 *Scenario Use Case* Melompat

Identifikasi	
Nomor	9
Nama	Melompat
Tujuan	Menggerakkan karakter dengan animasi melompat.
Deskripsi	
Tipe	*Primary Actor*
Aktor	Pengguna
Skenario Utama	

Kondisi awal	Pengguna memainkan *game*	
Aksi Aktor		**Reaksi Sistem**
1. Pengguna menggerakkan karakter dengan interaksi menggunakan *keyboard* menekan tombol SPASI.		2. Sistem menggerakkan karakter menjalankan animasi melompat.
Kondisi Akhir	Sistem menggerakkan karakter menjalankan animasi melompat sesuai inputan pengguna.	

Tabel III. 22 *Scenario Use Case* Menembak

Identifikasi		
Nomor	10	
Nama	Menembak	
Tujuan	Menggerakkan karakter dengan animasi menembak.	
Deskripsi		
Tipe	*Primary Actor*	
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 menggerakkan karakter menjalankan animasi menembak.
Kondisi Akhir	Sistem menggerakkan karakter menjalankan animasi menembak	

	sesuai inputan pengguna.

Tabel III. 23 *Scenario Use Case* Mengganti Senjata

Identifikasi		
Nomor	11	
Nama	Mengganti Senjata	
Tujuan	Menggerakkan karakter dengan animasi mengganti senjata.	
Deskripsi		
Tipe	*Primary Actor*	
Aktor	Pengguna	
Skenario Utama		
Kondisi awal	Pengguna memainkan *game*	
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 *machine gun*.
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 *machine gun* dan *sniper* sesuai inputan pengguna.	

Tabel III. 24 *Scenario Use Case* Mengambil Darah

Identifikasi	
Nomor	12
Nama	Mengambil Darah
Tujuan	Menambah darah untuk karakter.
Deskripsi	
Tipe	*Primary Actor*
Aktor	Pengguna
Skenario Utama	
Kondisi awal	Pengguna memainkan *game*
Aksi Aktor	**Reaksi Sistem**
1. Pengguna menggerakkan karakter dan menjalankan menuju *item health*.	2. Sistem menambah darah untuk karakter.
Kondisi Akhir	Sistem menampilkan darah karakter bertambah.

Tabel III. 25 *Scenario Use Case* Mengambil Senjata

Identifikasi	
Nomor	13
Nama	Mengambil Senjata
Tujuan	Menambah senjata untuk karakter.
Deskripsi	
Tipe	*Primary Actor*
Aktor	Pengguna
Skenario Utama	
Kondisi awal	Pengguna memainkan *game*
Aksi Aktor	**Reaksi Sistem**

1. Pengguna menggerakkan karakter dan menjalankan menuju *item* senjata.	2. Sistem menambah senjata untuk karakter.
Kondisi Akhir	Sistem menampilkan senjata karakter bertambah.

Tabel III. 26 *Scenario Use Case* Memberhentikan Permainan

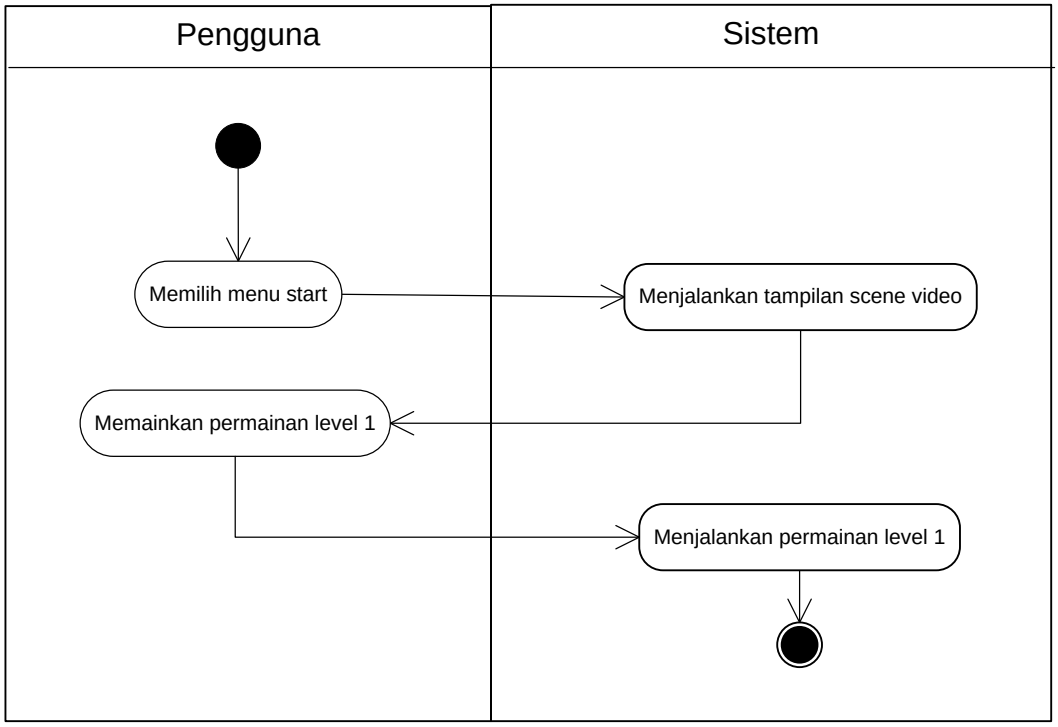
Identifikasi	
Nomor	14
Nama	Memberhentikan Permainan
Tujuan	Memberhentikan permainan untuk sementara.
Deskripsi	
Tipe	*Primary Actor*
Aktor	Pengguna
Skenario Utama	
Kondisi awal	Pengguna memainkan *game*
Aksi Aktor	**Reaksi Sistem**
1. Pengguna menekan tombol ESC pada *keyboard*.	2. Sistem menampilkan menu jeda.
3. Pengguna memilih *menu* kembali ke permainan.	4. Sistem mengembalikan permainan ke tampilan pengguna sebelumnya.
5. Pengguna memilih *menu* mengulang permainan.	6. Sistem mengembalikan permainan ke tampilan awal permainan.
7. Pengguna memilih *menu* keluar dari permainan.	8. Sistem keluar dari permainan dan menampilkan *menu* utama.
Kondisi Akhir	Sistem memberhentikan permainan mengeluarkan menu jeda. Jika pengguna memilih kembali ke permainan, maka sistem

	mengembalikan tampilan pengguna sebelumnya. Jika pengguna memilih mengulan permainan, maka sistem mengembalikan permainan ke tampilan awal. Jika pengguna memilih keluar dari permainan, maka sistem keluar dari permainan dan menampilkan *menu* utama.

III.4.3 *Activity Diagram*

Activity diagram menggambarkan berbagai alir aktivitas dalam sistem yang sedang dirancang, bagaimana masing-masing alir berawal, *decision* yang mungkin terjadi, dan bagaimana mereka berakhir. *Activity diagram* juga dapat menggambarkan proses paralel yang mungkin terjadi pada beberapa eksekusi. *Activity diagram* merupakan *state diagram* khusus, dimana sebagian besar *state* adalah action dan sebagian besar transisi di-*trigger* oleh selesainya *state* sebelum (*internal processing*).

1. *Activity Diagram* Proses Memilih Memulai Permainan

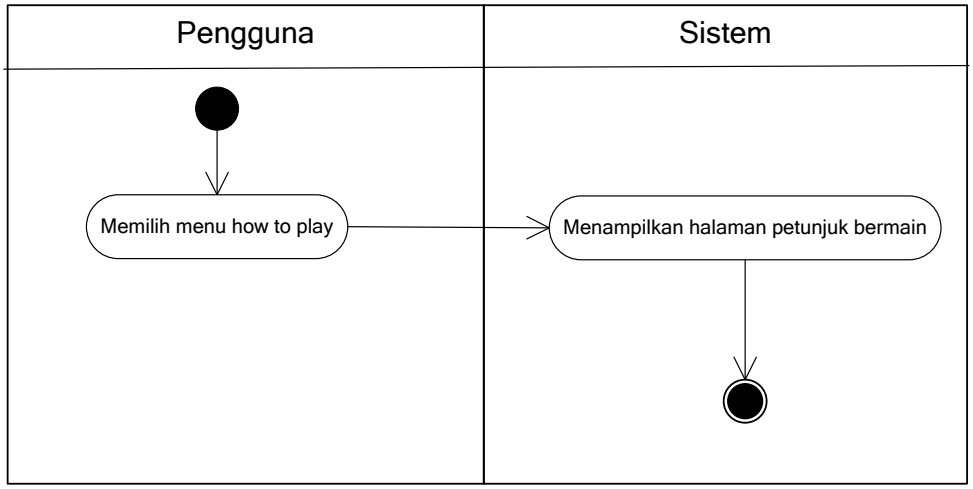


Gambar III. 19 *Activity diagram* Memilih Memulai Permainan

Tabel III. 27 Penjelasan Proses Memulai Permainan

Deskripsi	Menu mainkan merupakan menu yang digunakan untuk mulai memainkan permainan.
Normal Flow	1. Pengguna memilih menu *start* di menu utama 2. Sistem menjalankan tampilan *scene video* atau penggalan cerita sebelum masuk ke permainan 3. Pengguna memainkan permainan *level* 1 4. Sistem menjalankan permainan *level* 1

2. *Activity Diagram* Proses Memilih Melihat Tampilan *How To Play*

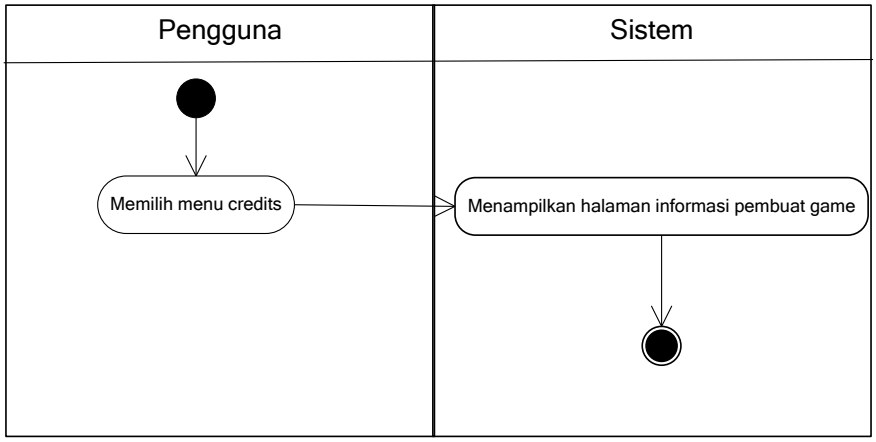


Gambar III. 20 *Activity diagram* Memilih Melihat Tampilan *How To Play*

Tabel III. 28 Penjelasan Proses Melihat Tampilan *How To Play*

Deskripsi	Menu *how to play* digunakan pemain untuk melihat petunjuk permainan
Normal Flow	1. Memilih menu *how to play* di menu utama 2. Sistem menampilkan halaman petunjuk bermain

3. *Activity Diagram* Proses Memilih Melihat Tampilan *Credits.*

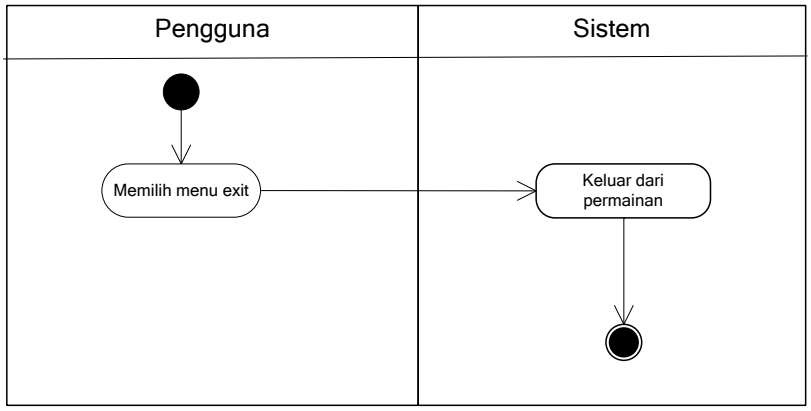


Gambar III. 21 *Activity diagram* Memilih Melihat Tampilan *Credits*

Tabel III. 29 Penjelasan Proses Melihat Tampilan *Credits*

Deskripsi	Menu *credits* digunakan pemain untuk melihat informasi pembuat *game.*
Normal Flow	1. Pengguna memilih menu *credits* 2. Sistem menampilkan halaman informasi pembuat *game.*

4. *Activity Diagram* Proses Memilih Keluar Dari Permainan.

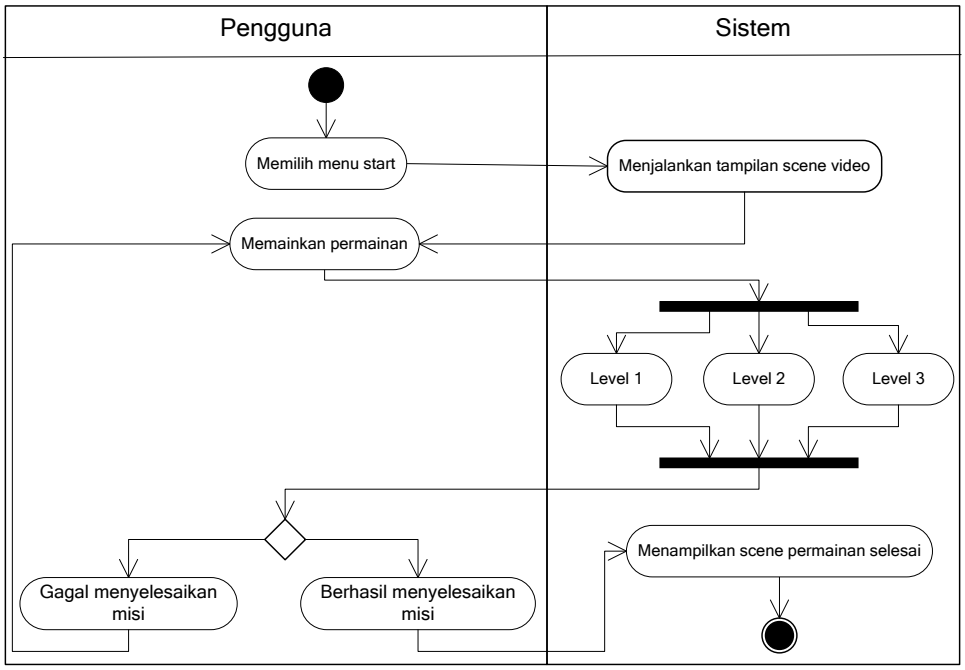


Gambar III. 22 *Activity diagram* Memilih Keluar Dari Permainan

Tabel III. 30 Penjelasan Proses Memilih Keluar Dari Permainan

Deskripsi	Menu *exit* merupakan menu yang digunakan untuk keluar dari permainan
Normal Flow	1. Pengguna memilih menu *exit* pada menu utama 2. Sistem keluar dari permainan

5. *Activity Diagram* Proses Memilih Memainkan Permainan



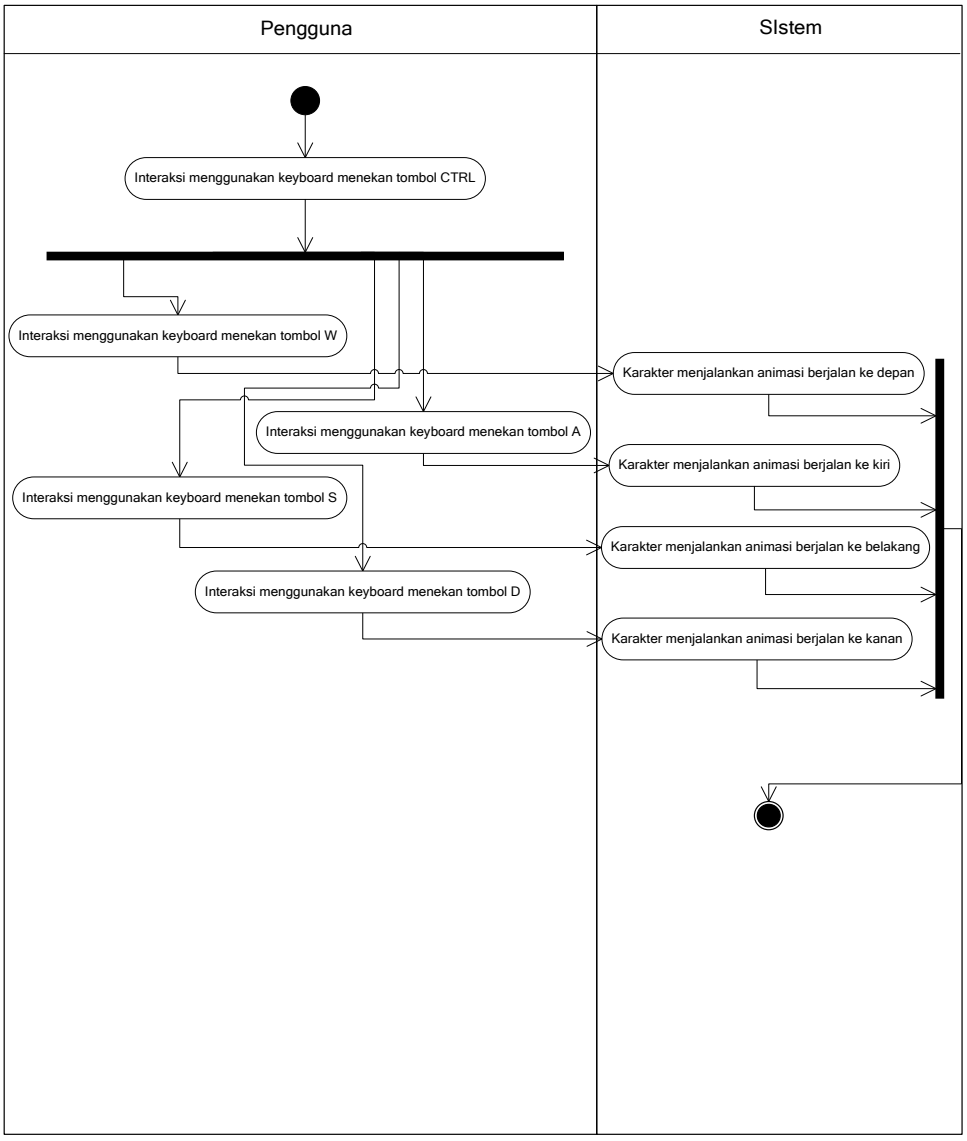
Gambar III. 23 *Activity diagram* Memilih Memainkan Permainan

Tabel III. 31 Penjelasan Proses Memilih Memainkan Permainan

Deskripsi	Memulai permainan merupakan menu yang digunakan untuk memulai permainan
Normal Flow	1. Pengguna memilih menu *start* 2. Sistem menjalankan tampilan *scene video* 3. Pengguna memainkan permainan 4. Sistem menjalankan permainan *level* 1, *level* 2, dan *level* 3

	5. Pengguna berhasil menyelesaikan misi 6. Sistem menampilkan *scene* permainan selesai 7. Pengguna gagal menyelesaikan misi 8. Sistem mengulang menampilkan permainan

6. *Activity Diagram* Proses Memilih Berjalan

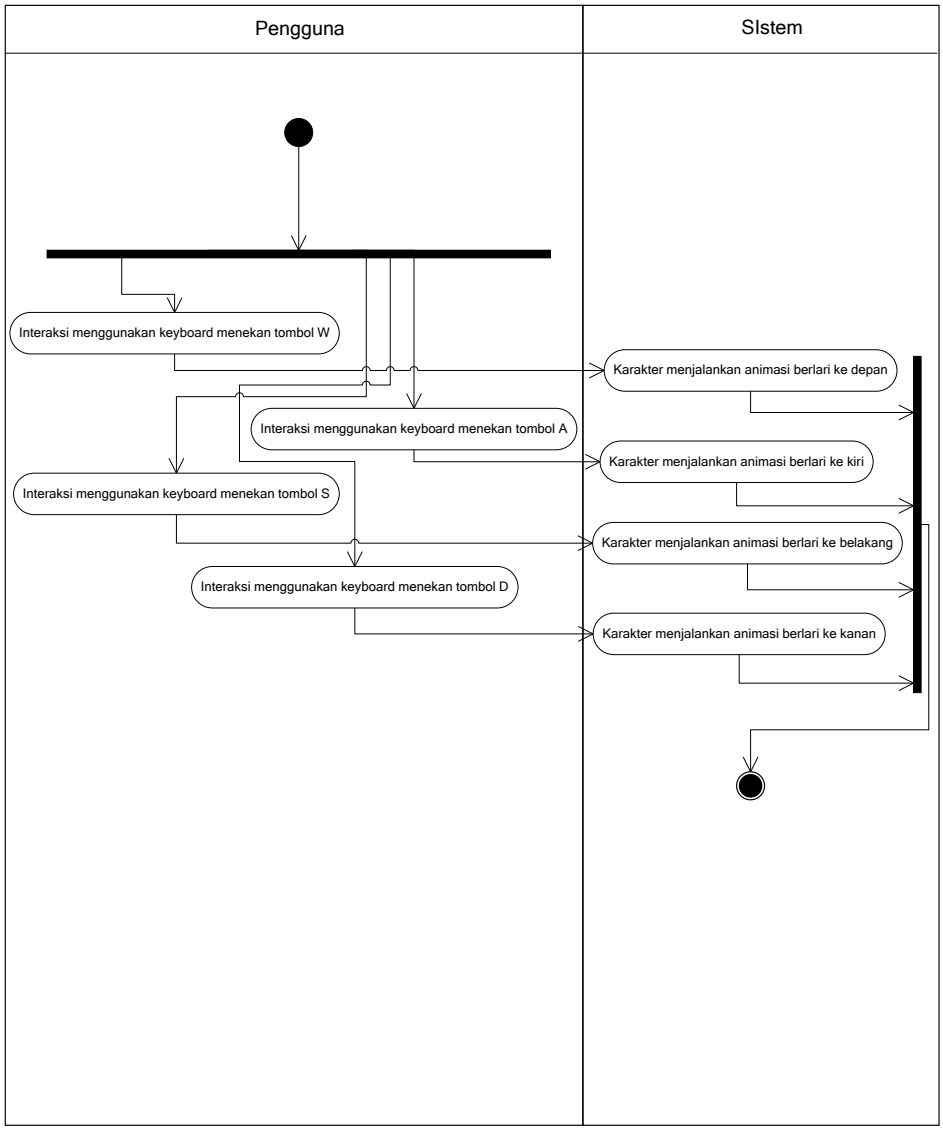


Gambar III. 24 *Activity diagram* Memilih Berjalan

Tabel III. 32 Penjelasan Proses Memilih Berjalan

Deskripsi	Proses untuk menggerakan karakter atau tokoh utama yang dimainkan dalam permainan menggunakan tombol *keyboard.*
Normal Flow	1. Pengguna menggerakkan karakter dengan interaksi menggunakan *keyboard* menekan tombol CTRL dan W. 2. Sistem menggerakkan karakter menjalankan animasi berjalan ke depan. 3. Pengguna menggerakkan karakter dengan interaksi menggunakan *keyboard* menekan tombol CTRL dan A. 4. Sistem menggerakkan karakter menjalankan animasi berjalan ke kiri. 5. Pengguna menggerakkan karakter dengan interaksi menggunakan *keyboard* menekan tombol CTRL dan S. 6. Sistem menggerakkan karakter menjalankan animasi berjalan ke belakang. 7. Pengguna menggerakkan karakter dengan interaksi menggunakan *keyboard* menekan tombol CTRL dan D. 8. Sistem menggerakkan karakter menjalankan animasi berjalan ke kanan.

7. *Activity Diagram* Proses Memilih Berlari



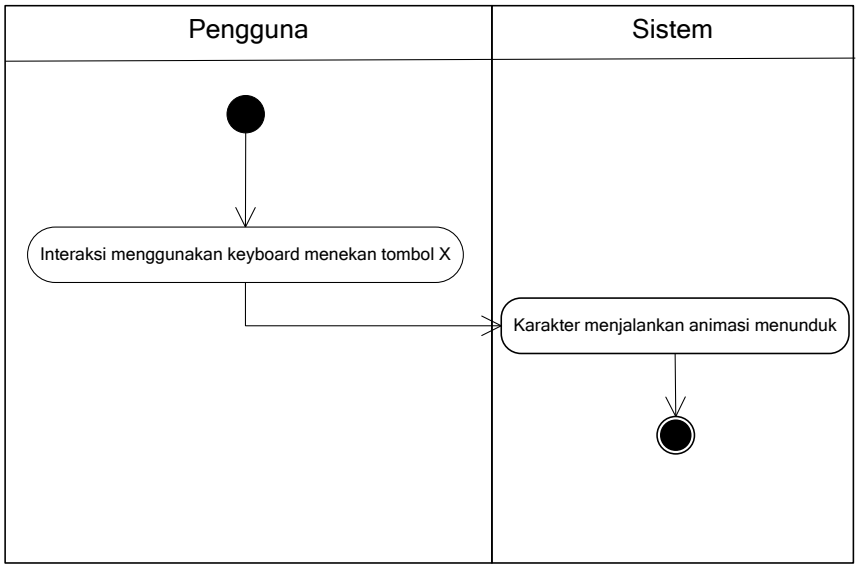
Gambar III. 25 *Activity diagram* Memilih Berlari

Tabel III. 33 Penjelasan Proses Memilih Berlari

Deskripsi	Proses untuk menggerakan karakter atau tokoh utama yang dimainkan dalam permainan menggunakan tombol *keyboard.*
Normal Flow	1. Pengguna menggerakkan karakter dengan interaksi menggunakan *keyboard* menekan tombol W.

	2. Sistem menggerakkan karakter menjalankan animasi berlari ke depan. 3. Pengguna menggerakkan karakter dengan interaksi menggunakan *keyboard* menekan tombol A. 4. Sistem menggerakkan karakter menjalankan animasi berlari ke kiri. 5. Pengguna menggerakkan karakter dengan interaksi menggunakan *keyboard* menekan tombol S. 6. Sistem menggerakkan karakter menjalankan animasi berlari ke belakang. 7. Pengguna menggerakkan karakter dengan interaksi menggunakan *keyboard* menekan tombol D. 8. Sistem menggerakkan karakter menjalankan animasi berlari ke kanan.

8. *Activity Diagram* Proses Memilih Menunduk

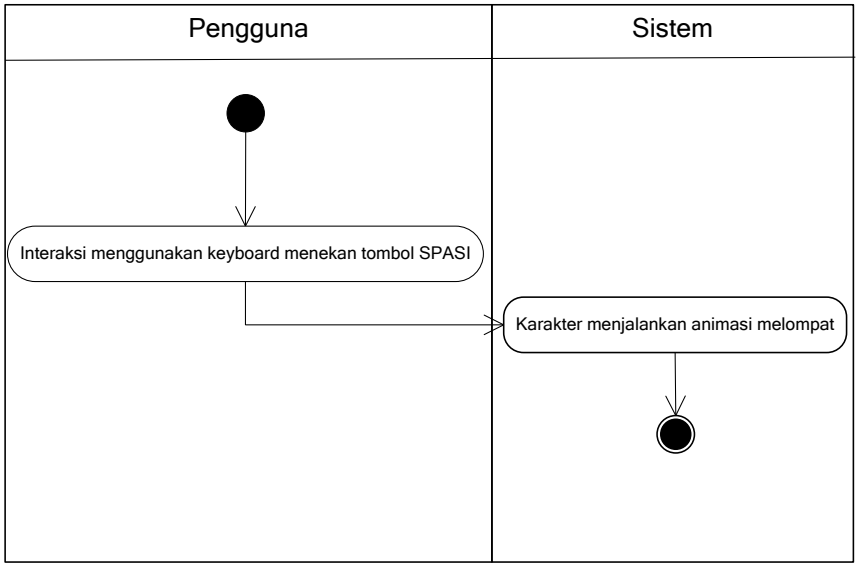


Gambar III. 26 *Activity diagram* Memilih Menunduk

Tabel III. 34 Penjelasan Proses Memilih Menunduk

Deskripsi	Proses untuk menggerakan karakter atau tokoh utama yang dimainkan dalam permainan menggunakan tombol *keyboard.*
Normal Flow	1. Pengguna menggerakkan karakter dengan interaksi menggunakan *keyboard* menekan tombol X. 2. Sistem menggerakkan karakter menjalankan animasi menunduk.

9. *Activity Diagram* Proses Memilih Melompat

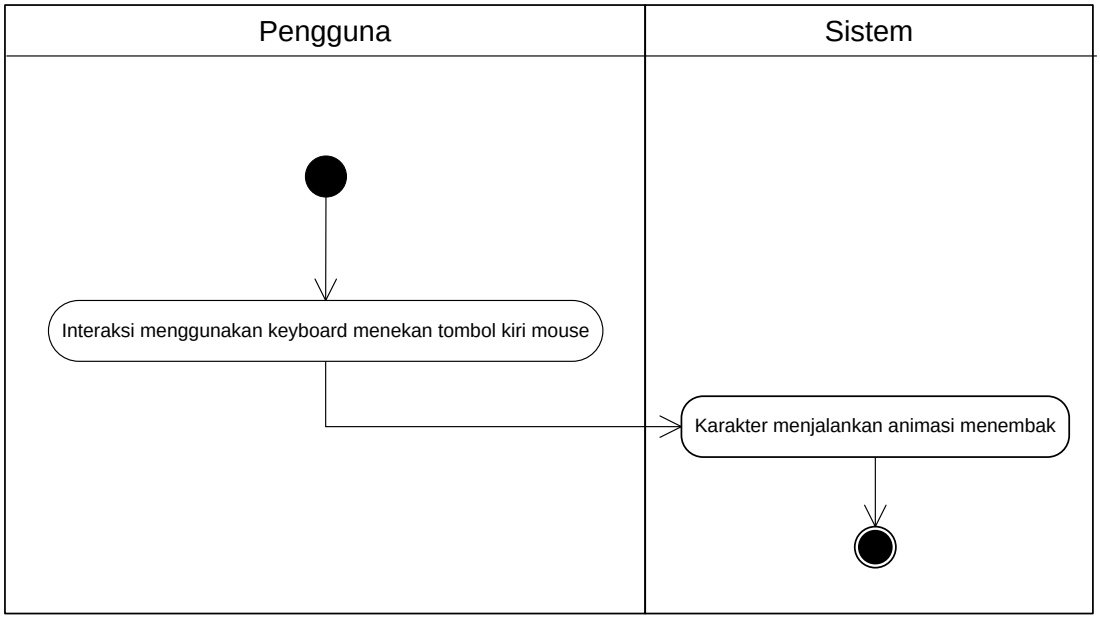


Gambar III. 27 *Activity diagram* Memilih Melompat

Tabel III. 35 Penjelasan Proses Memilih Melompat

Deskripsi	Proses untuk menggerakan karakter atau tokoh utama yang dimainkan dalam permainan menggunakan tombol *keyboard.*
Normal Flow	1. Pengguna menggerakkan karakter dengan interaksi menggunakan *keyboard* menekan tombol SPASI. 2. Sistem menggerakkan karakter menjalankan animasi melompat.

10. *Activity Diagram* Proses Memilih Menembak

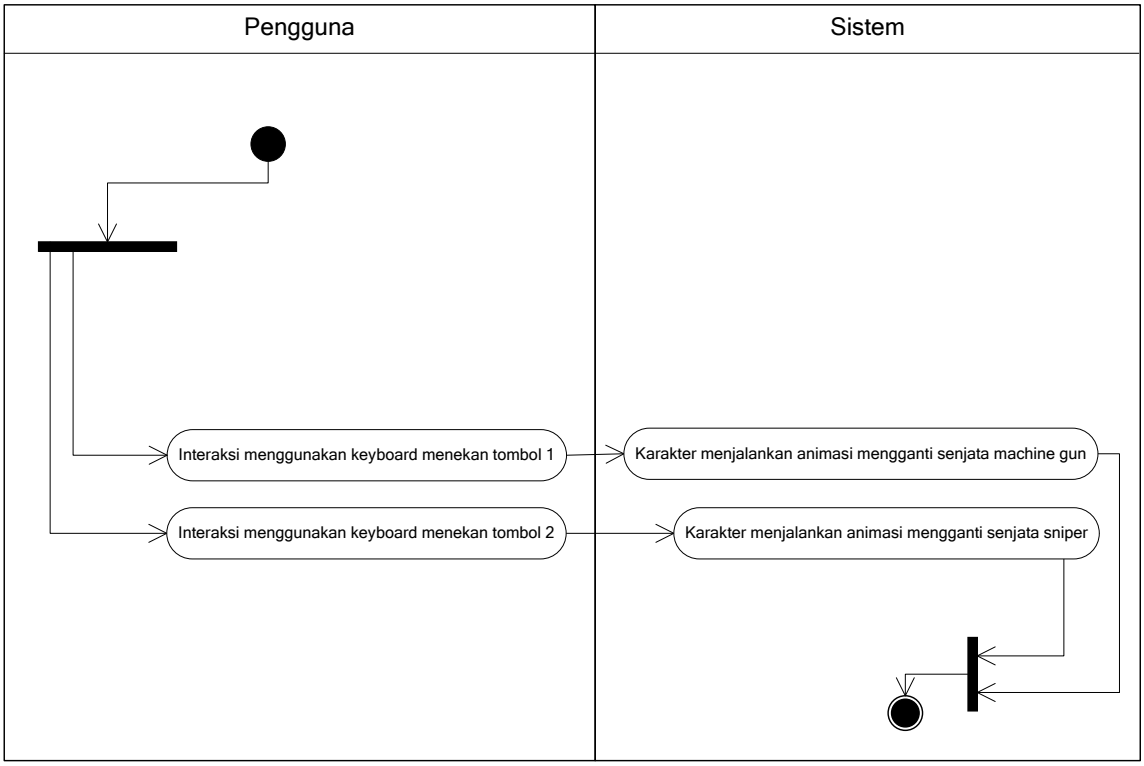


Gambar III. 28 *Activity diagram* Memilih Menembak

Tabel III. 36 Penjelasan Proses Memilih Menembak

Deskripsi	Proses untuk menggerakan karakter atau tokoh utama yang dimainkan dalam permainan menggunakan tombol *keyboard.*
Normal Flow	1. Pengguna menggerakkan karakter untuk animasi menembak dengan interaksi menggunakan tombol kiri *mouse*. 2. Sistem menggerakkan karakter menjalankan animasi menembak.

11. *Activity Diagram* Proses Memilih Mengganti Senjata



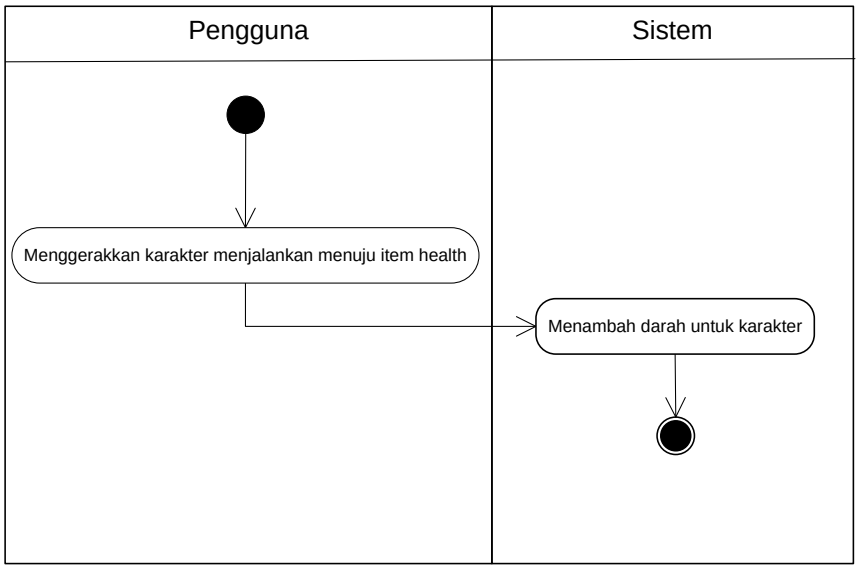
Gambar III. 29 *Activity diagram* Memilih Mengganti Senjata

Tabel III. 37 Penjelasan Proses Memilih Mengganti Senjata

Deskripsi	Proses mengganti senjata dengan pilihan senjata *machine gun* atau senjata *sniper*
Normal Flow	1. Pengguna menggerakkan karakter menjalankan animasi mengganti senjata dengan interaksi menggunakan *keyboard* menekan tombol 1 2. Sistem menampilkan karakter menjalankan animasi mengganti senjata *machine gun* 3. Pengguna menggerakkan karakter menjalankan animasi mengganti senjata dengan interaksi menggunakan *keyboard* menekan tombol 2.

	4. Sistem menampilkan karakter menjalankan animasi mengganti senjata *sniper*.

12. *Activity Diagram* Proses Memilih Menambah Darah

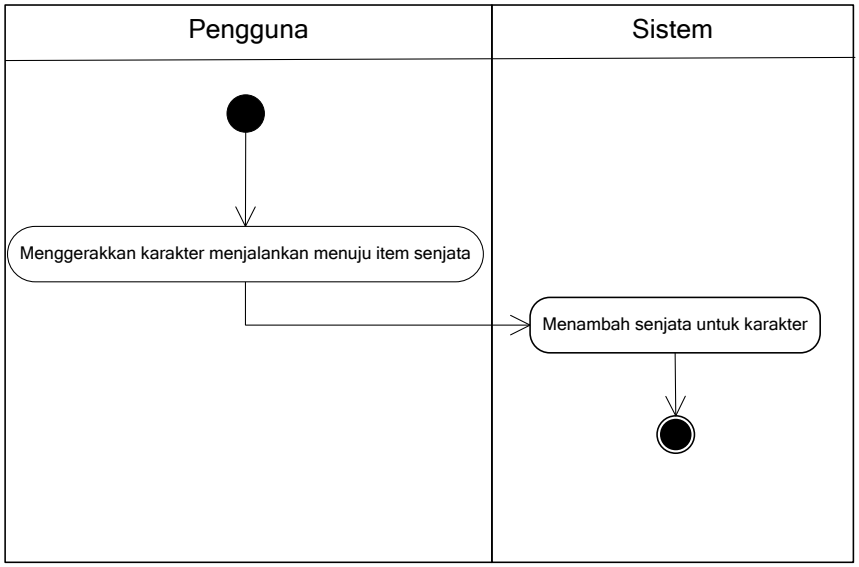


Gambar III. 30 *Activity diagram* Memilih Menambah Darah

Tabel III. 38 Penjelasan Proses Memilih Menambah Darah

Deskripsi	Proses untuk menambah darah untuk karakter..
Normal Flow	1. Pengguna menggerakkan karakter dan menjalankan menuju *item health.* 2. Sistem menambah darah untuk karakter.

13. *Activity Diagram* Proses Memilih Menambah Senjata

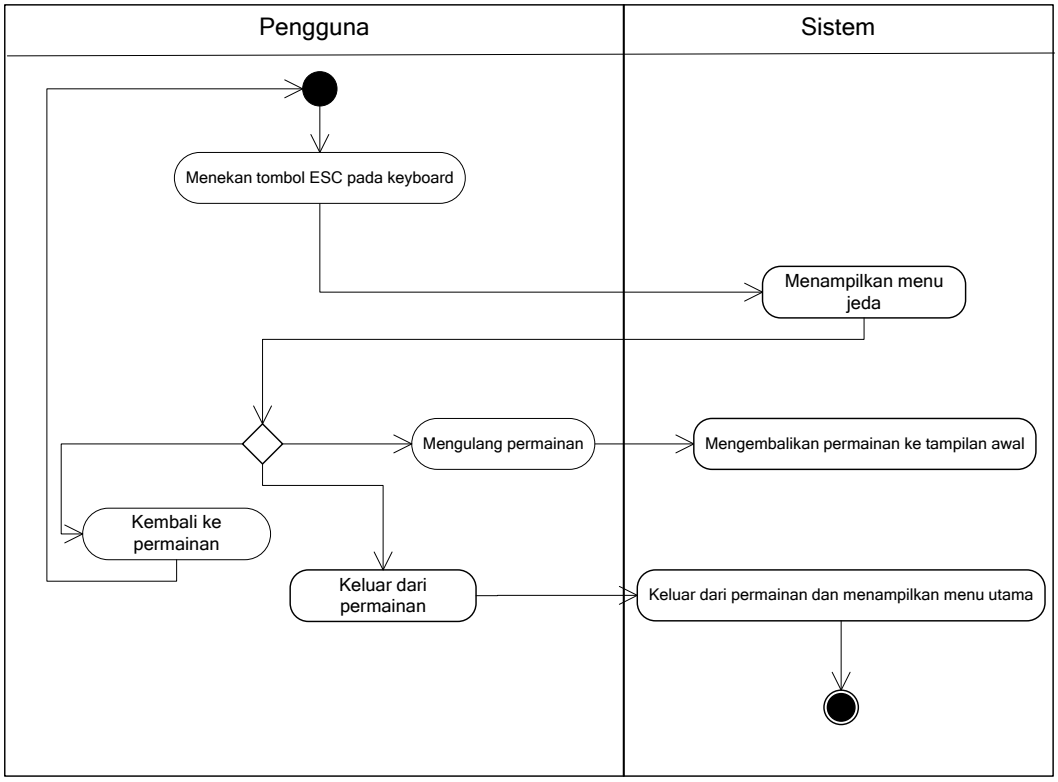


Gambar III. 31 *Activity diagram* Memilih Menambah Senjata

Tabel III. 39 Penjelasan Proses Memilih Menambah Senjata

Deskripsi	Proses untuk menambah senjata untuk karakter..
Normal Flow	3. Pengguna menggerakkan karakter dan menjalankan menuju *item* senjata. 4. Sistem menambah senjata untuk karakter.

14. *Activity Diagram* Proses Memilih Memberhentikan Permainan



Gambar III. 32 *Activity diagram* Memilih Memberhentikan Permainan

Tabel III. 40 Penjelasan Proses Memilih Memberhentikan Permainan

Deskripsi	Proses ketika pemain ingin melihat petunjuk permainan dan informasi kontrol atau keluar dari permainan
Normal Flow	1. Pengguna menekan tombol ESC pada *keyboard*. 2. Sistem menampilkan menu jeda. 3. Pengguna memilih *menu* kembali ke permainan. 4. Sistem mengembalikan permainan ke tampilan pengguna sebelumnya 5. Pengguna memilih *menu* mengulang permainan. 6. Sistem mengembalikan permainan ke tampilan awal permainan.

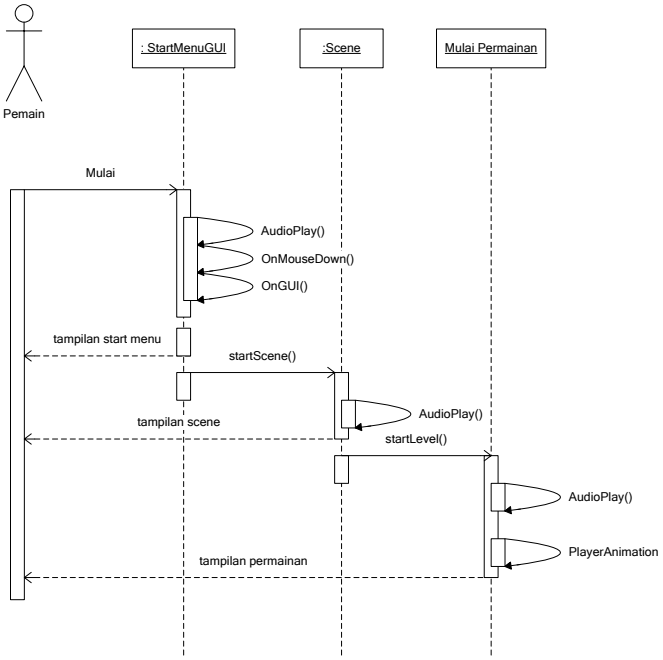
	7. Pengguna memilih *menu* keluar dari permainan. 8. Sistem keluar dari permainan dan menampilkan *menu* utama.

III.4.4 *Sequence Diagram*

Sequence diagram menggambarkan interaksi antar objek di dalam dan di sekitar sistem (termasuk pengguna, display, dan sebagainya) berupa message yang digambarkan terhadap waktu. *Sequence diagram* terdiri atas dimensi vertikal (waktu) dan dimensi horizontal (objek-objek yang terkait).

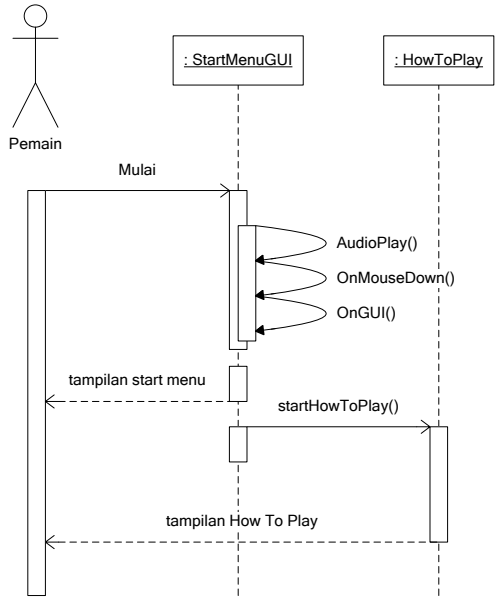
Sequence diagram biasa digunakan untuk menggambarkan skenario atau rangkaian langkah-langkah yang dilakukan sebagai respon dari sebuah *event* untuk menghasilkan *output* tertentu. Diawali dari apa yang men-*trigger* aktivitas tersebut, proses dan perubahan apa saja yang terjadi secara *internal* dan *output* apa yang dihasilkan.

1. *Sequence Diagram* Memulai Permainan



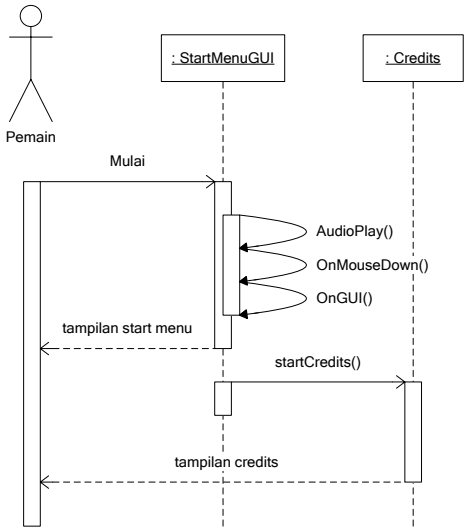
Gambar III. 33 *Sequence Diagram* Memulai Permainan

2. *Sequence Diagram* Melihat Tampilan *How To Play*



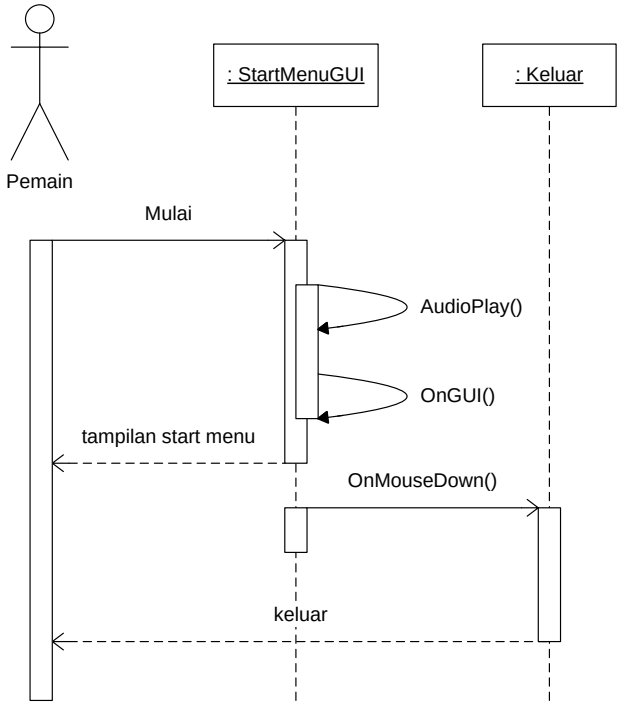
Gambar III. 34 *Sequence Diagram* Melihat Tampilan *How To Play*

3. *Sequence Diagram* Melihat Tampilan *Credits*



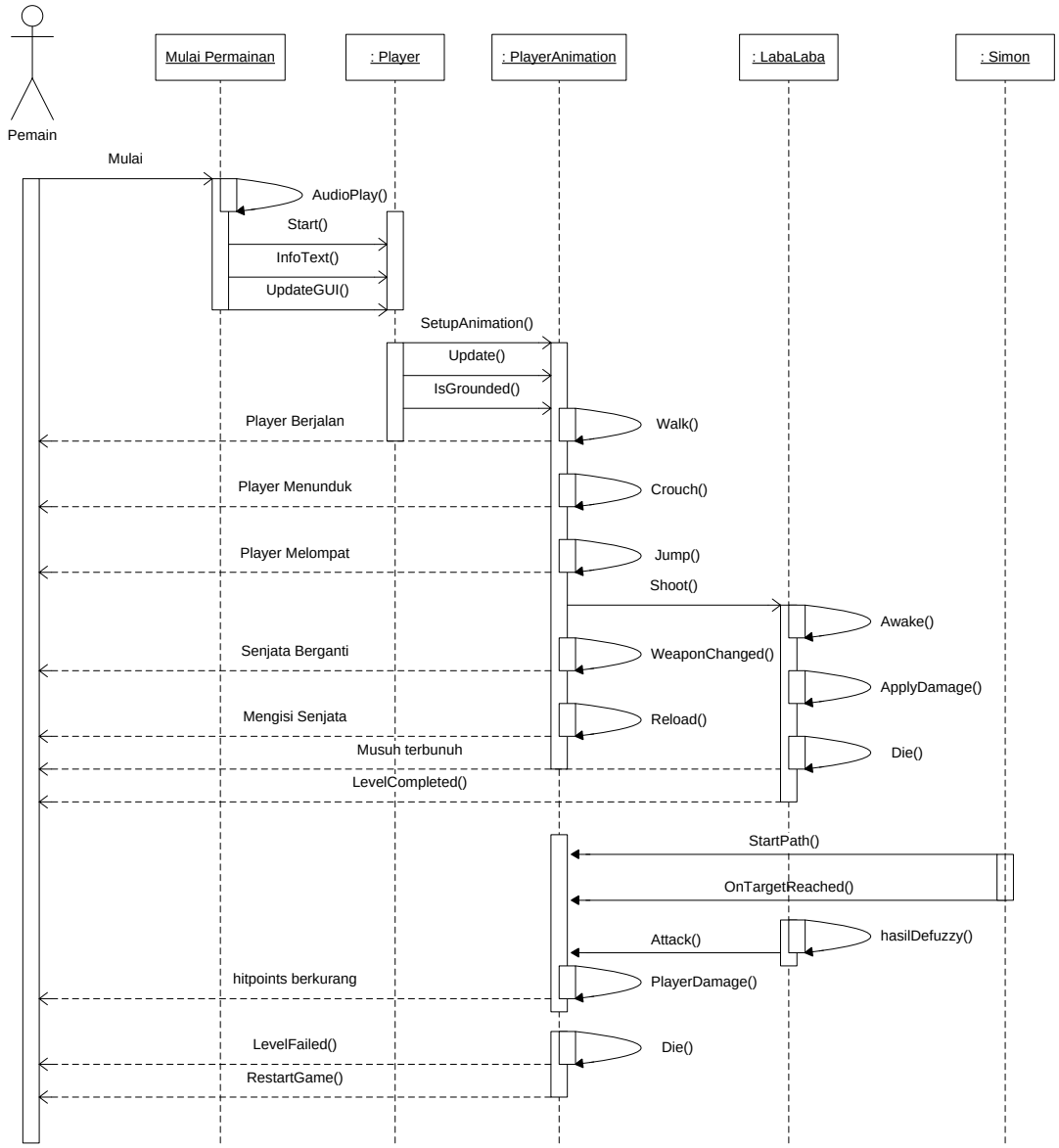
Gambar III. 35 *Sequence Diagram* Melihat Tampilan *Credits*

4. *Sequence Diagram* Keluar Dari Permainan



Gambar III. 36 *Sequence Diagram* Keluar Dari Permainan

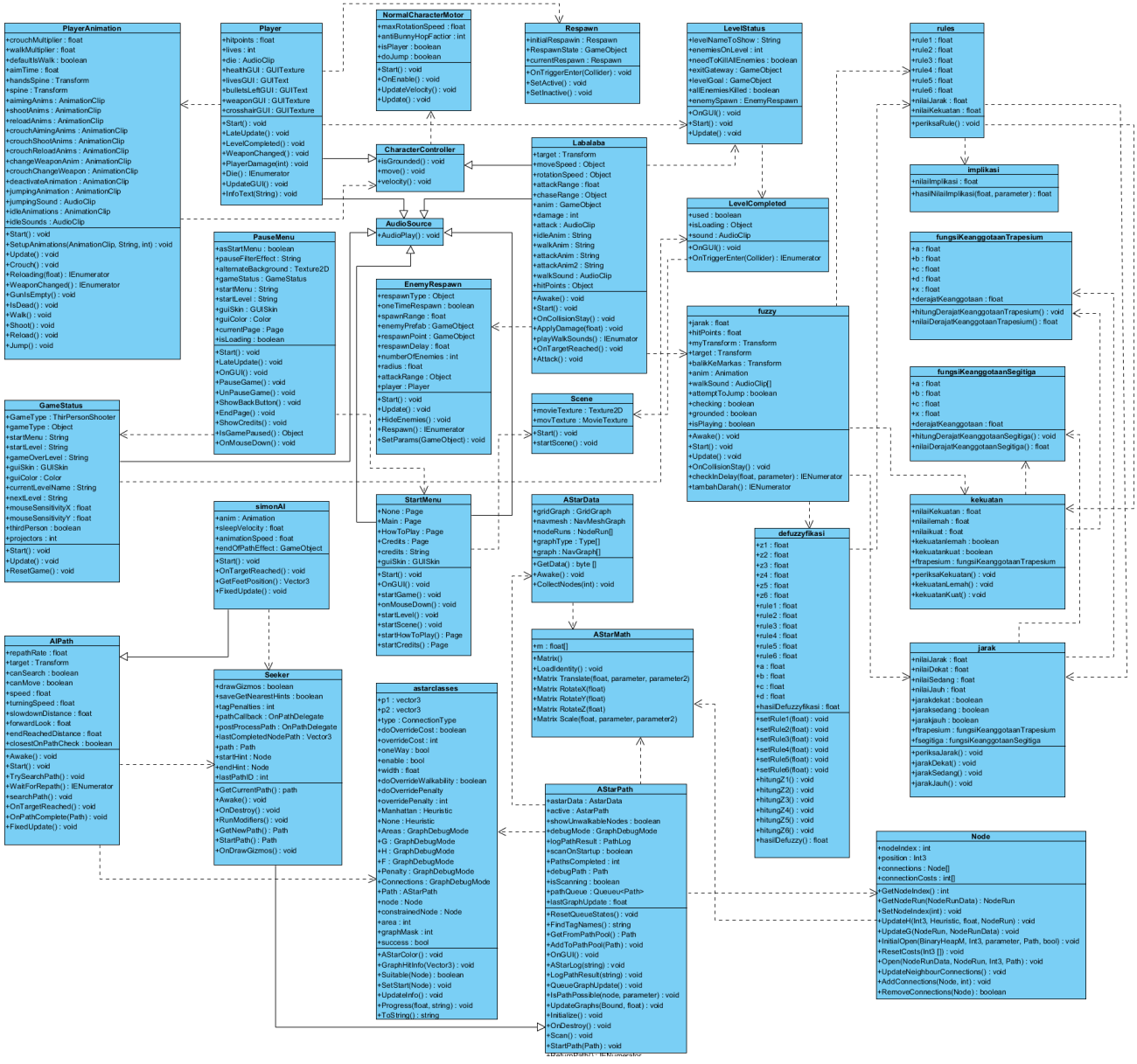
5. *Sequence Diagram* Memainkan Permainan



Gambar III. 37 *Sequence Diagram* Memainkan Permainan

III.4.5 *Class Diagram*

Class diagram menggambarkan struktur sistem dari segi pendefinisian kelas-kelas yang dibuat untuk membangun sistem. Berikut adalah *class diagram* pada game *Action* 3D *The Walker*.



Gambar III. 38 *Class Diagram*

III.5 Perancangan Sistem

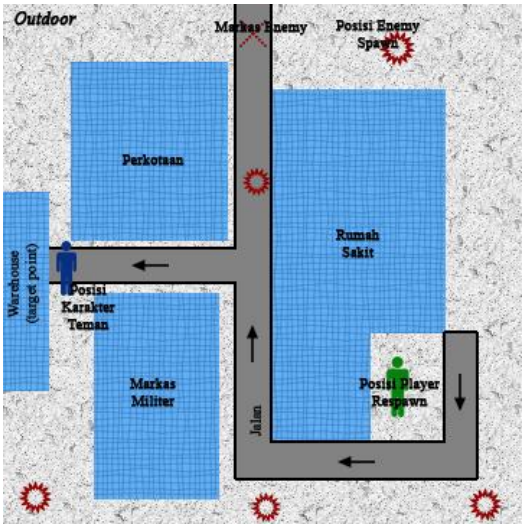
Perancangan adalah suatu bagian dari metodologi pengembangan suatu perangkat lunak yang dilakukan setelah tahapan analisis untuk memberikan gambaran secara terperinci.

Perancangan sistem merupakan tahapan dari siklus pengembangan sistem yang didefinisikan dari kebutuhan-kebutuhan fungsional dan persiapan untuk rancang bangun implementasi yang menggambarkan bagaimana suatu sistem dibentuk, yang dapat berupa penggambaran, perancangan, dan pembuatan sketsa atau pengaturan dari beberapa elemen yang terpisah kedalam satu kesatuan yang utuh dan berfungsi, juga menyangkut konfigurasi dari komponen-komponen perangkat keras dan perangkat lunak dari suatu sistem.

III.5.1 *Storyboard*

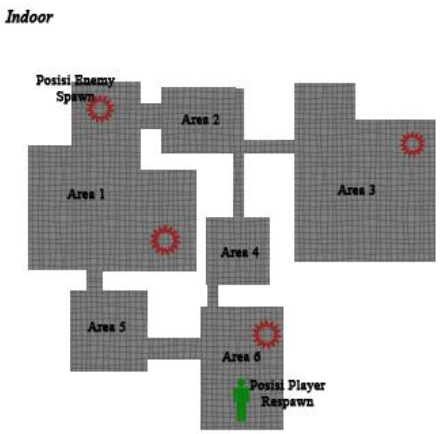
Storyboard adalah sketsa gambar yang disusun berurutan sesuai dengan naskah, dengan menggunakan *storyboard* dapat menyampaikan ide cerita kepada orang lain dengan lebih mudah karena dengan storyboard dapat mengimajinasikan khayalan yang sesuai dengan gambar-gambar yang ada sehingga dapat menghasilkan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

Berikut beberapa *storyboard* dari *game action* 3D *The Walker* yaitu.



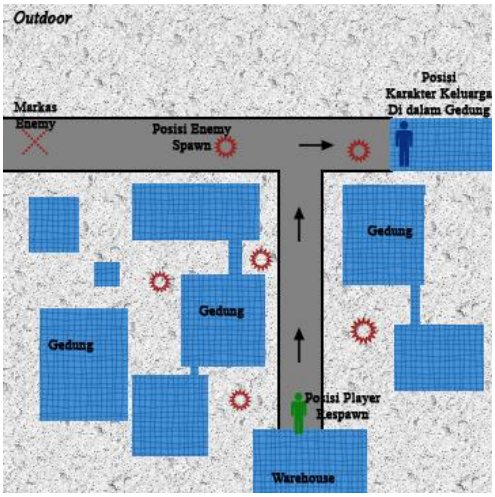
Gambar III. 39 *Storyboard Level* 1

Pada *level* 1 posisi *player* atau karakter utama terletak disebelah kanan bawah dari *map*, sedangkan *target point* terletak disebelah kiri ditengah-tengah map. Dimana *target point* terletak didepan sebuah gudang atau *warehouse* dan didepannya juga terdapat satu orang karakter teman dan *sentry gun* yang menunggu *player* datang. Lambang bulat berwarna merah adalah posisi *enemy spawn* atau *spot-spot* tempat munculnya *enemy*.



Gambar III. 40 *Storyboard Level* 2

Level 2 terletak di dalam gudang atau *warehouse* yang didalamnya terdapat beberapa area seperti gambar III.40 diatas.



Gambar III. 41 *Storyboard Level* 3

Pada *level* 3, karakter utama akan keluar dari gudang atau *warehouse* dan mencari karakter keluarganya yang berada di dalam sebuah gedung seperti terlihat pada gambar III.41 diatas.

III.5.2 Pengenalan Karakter

1. Karakter Utama

Karakter utama dalam *game* ini bernama *Rick* seorang tentara amerika yang dirancang untuk menghindari dan melawan para laba-laba. *Rick* memiliki *hitpoints* 100 yang berarti memiliki 100 kekuatan atau darah, artinya *hitpoint* akan berkurang sesuai dengan *damage* yang dihasilkan oleh musuh atau *enemy*. Mempunyai *lives* sejumlah 5 yang berarti apabila mati akan balik lagi ketempat *respawn*-nya sejumlah lima kali. *Rick* juga memiliki *abilities* atau kemampuan seperti berlari, berjalan, menunduk, melompat, dan menembak, juga dilengkapi dengan persenjataan berupa satu senjata *machine gun* dan satu senjata *sniper*. Dengan penjelasan berupa gambar dan tabel sebagai berikut:



Gambar III. 42 Karakter Utama

Tabel III. 41 Properties Karakter Utama

Karakter Utama	
Nick Name	Rick
***Hit Points* (HP)**	100
Lives	5
Abilities	Berlari, Berjalan, Menunduk, Melompat, Menembak
Guns	*Machine gun, Sniper*

2. Senjata / *Guns*

Senjata yang digunakan *Rick* ada dua macam, yaitu senjata jenis *machine gun* dan senjata jenis *sniper*, yang keduanya memiliki peluru *(bullet)* yang terbatas. *Machine gun* memiliki *damage* 20 yang berarti akan mengurangi *hitpoints enemy* sejumlah 20, sedangkan *sniper* memiliki *damage* 50. Untuk *machine gun* memiliki 5 *pack* yang di dalamnya berisi 30 butir peluru. Sedangkan untuk *sniper* memiliki 5 *pack* yang di dalamnya berisi 5 butir peluru. Sedangkan untuk *reload time* berarti waktu yang diperlukan untuk melakukan *reload* atau mengisi ulang senjata.

Berikut ini adalah penjelasan perbedaan dari kedua senjata yang digunakan:



Gambar III. 43 Senjata *Machine Gun*



Gambar III. 44 Senjata *Sniper*

Tabel III. 42 Properties Senjata

Senjata		
	Machine Gun	***Sniper***
Damage	20	50
Bullet	30 x 5	5 x 5
Reload Time	0.5 second	1 second

3. Musuh atau *Enemy*

Musuh yang terdapat di dalam *game* ini berupa laba-laba yang berukuran lebih besar dari laba-laba biasanya. Terdapat dua jenis laba-laba di dalam *game*. Yang pertama adalah laba-laba berwarna hitam dengan AI atau kecerdasan buatan biasa dengan *abilities* yang kurang hanya bisa berlari dan menyerang. Sedangkan, yang kedua laba-laba berwarna merah dengan AI atau kecerdasan buatan menggunakan *fuzzy logic* dengan *abilities* bisa menyerang secara brutal, menyerang secara biasa, melarikan diri menuju markasnya, bertahan, dan mengisi darah atau *hitpoints* nya. Perbedaan menyerang secara brutal dengan yang biasa terletak pada *damage* dan *speed* yang dihasilkan. *Damage* nilai *random* antara 20 – 30 untuk menyerang biasa dan nilai 30 untuk menyerang brutal, artinya akan mencapai nilai maksimum 30 apabila menyerang brutal, dan *damage* yang dihasilkan artinya akan mengurangi *hitpoints player* sesuai jumlah yang dihasilkan. Sedangkan *speed* 3 untuk menyerang biasa dan bertambah satu menjadi 4 untuk menyerang brutal, artinya akan melebihi *speed* dari *player* yaitu 3.5.

Berikut ini adalah penjelasan perbedaan dari kedua *enemy*:



Gambar III. 45 Laba-laba Hitam

Tabel III. 43 Properties Musuh Laba-Laba Hitam

Laba-Laba Hitam	
***Hit Points* (HP)**	100
Lives	0
Abilities	Berlari, Menyerang
Damage	Nilai *random* antara 20 - 30
Speed	3



Gambar III. 46 Laba-laba Merah

Tabel III. 44 Properties Musuh Laba-Laba Merah

Laba-Laba Merah	
***Hit Points* (HP)**	100
Lives	0
Abilities	Berlari, menyerang secara brutal, menyerang biasa, bertahan, melarikan diri ke markasnya, mengisi *hitpoints*

	sendiri.
Damage	Nilai *random* antara 20 – 30 untuk menyerang biasa. Nilai 30 untuk menyerang secara brutal.
Speed	3 untuk menyerang biasa, 4 untuk menyerang secara brutal.

4. Karakter Teman atau *Ally*

Karakter teman yang terdapat di dalam *game* ini berjumlah tiga, yang pertama karakter *Shane* yang berada di *level* 1, yang kedua karakter *Simon* yang berada di *level* 3, yang ketiga karakter *Sentry Gun* yang berada di *level* 1. Dijelaskan sebagai berikut:



Gambar III. 47 Karakter *Shane*

Tabel III. 45 Properties Karakter *Shane*

Karakter Teman (Shane)	
Nick Name	Shane
***Hit Points* (HP)**	100
Lives	0
Abilities	Menembak
Guns	*Machine gun*



Gambar III. 48 Karakter *Simon*

Tabel III. 46 Properties Karakter *Simon*

Karakter Teman (Simon)	
Nick Name	Simon
***Hit Points* (HP)**	100
Lives	0
Abilities	Berlari
Guns	-



Gambar III. 49 Karakter *Sentry Gun*

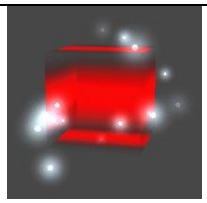
Tabel III. 47 Properties Karakter *Sentry Gun*

Karakter Teman (Sentry Gun)	
Name	Sentry Gun
***Hit Points* (HP)**	100
Lives	0
Abilities	Menembak
Damage	10

5. *Items*

Berikut ini adalah *items-items* yang terdapat di dalam *game* dan yang berhubungan dengan jalan cerita dari *game* itu sendiri beserta penjelasannya.

Tabel III. 48 Tabel Penjelasan *Items*

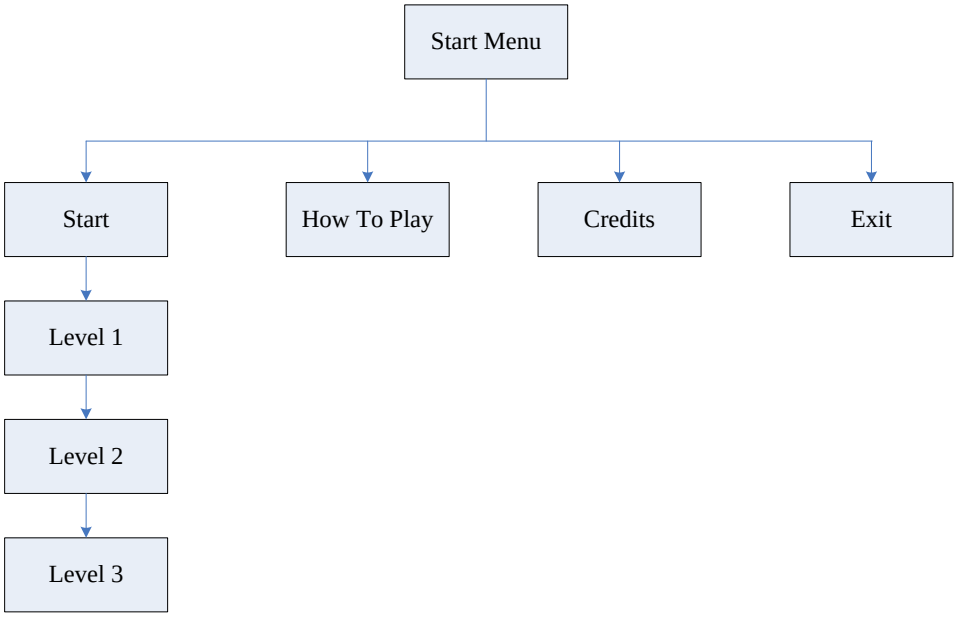
Gambar *Items*	**Nama *Items***	**Keterangan**
	*Sparkle*	*Target Point*, menunjukkan tempat yang harus dicapai untuk menyelesaikan misi.
	*Health*	*Item* yang bisa diambil oleh karakter utama untuk menambah darah atau *hitpoints*. Satu kali ambil bertambah darah 50 *hitpoints*.
	*Player Respawn*	Tempat karakter utama hidup kembali, apabila *hitpoints* = 0 dan *lives* lebih dari 0.

III.6 Perancangan Arsitektur

Setelah melakukan perancangan pada sistem yang dibangun, maka dilakukanlah perancangan arsitektur. Perancangan arsitektur yang telah dibuat meliputi beberapa perancangan diantaranya struktur menu, perancangan antarmuka, dan jaringan semantik.

III.6.1 Perancangan Struktur Menu

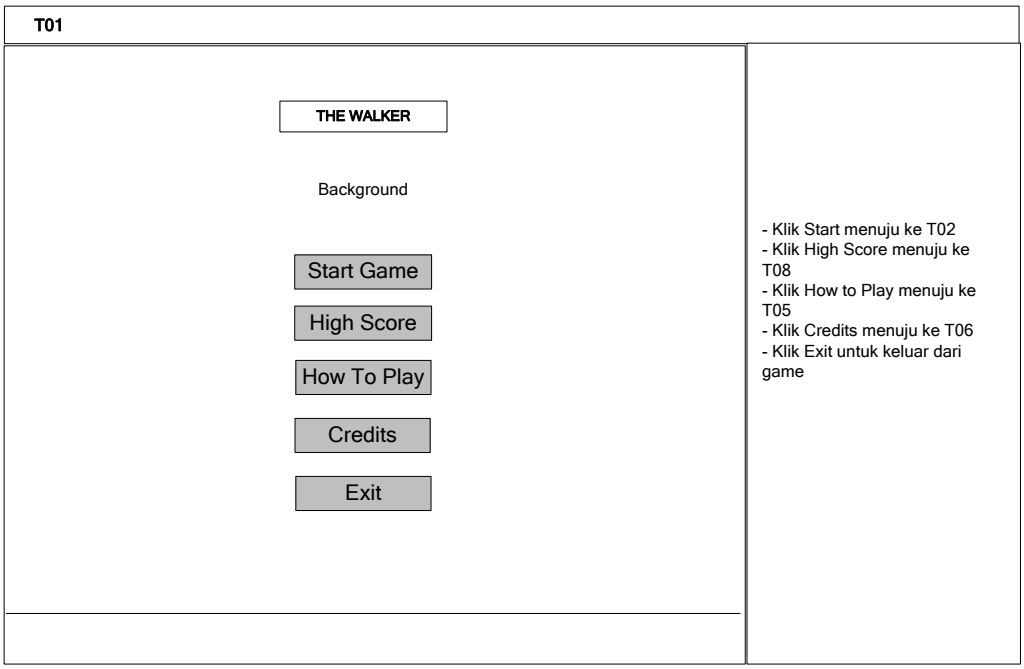
Perancangan struktur menu dirancang untuk menyediakan fungsi-fungsi yang akan digunakan dalam *game action* 3D *The Walker*.



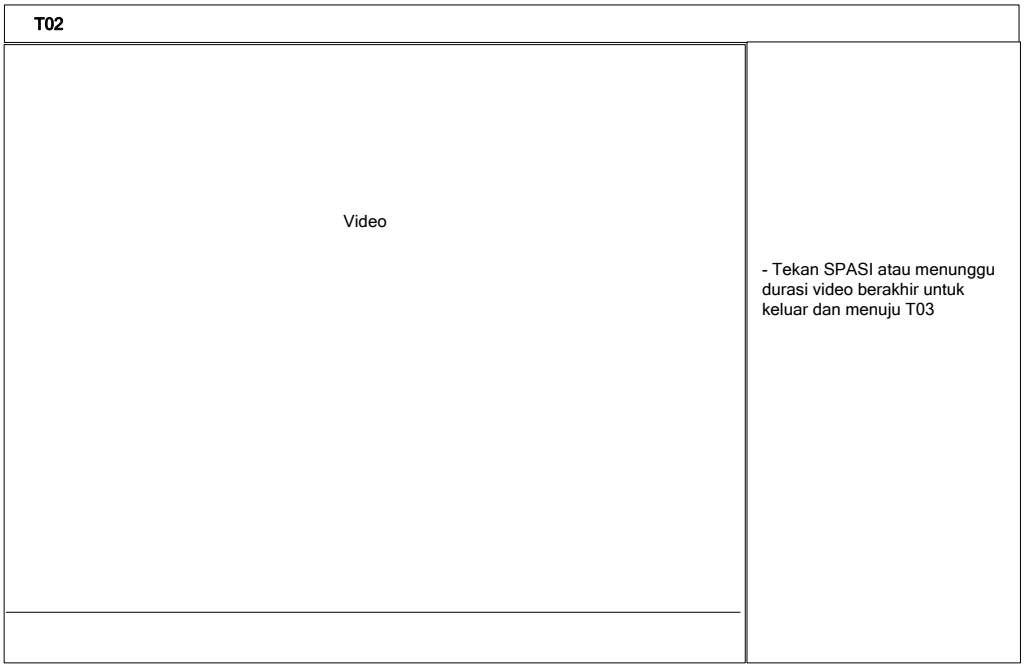
Gambar III. 50 Struktur Menu

III.6.2 Perancangan Antar Muka

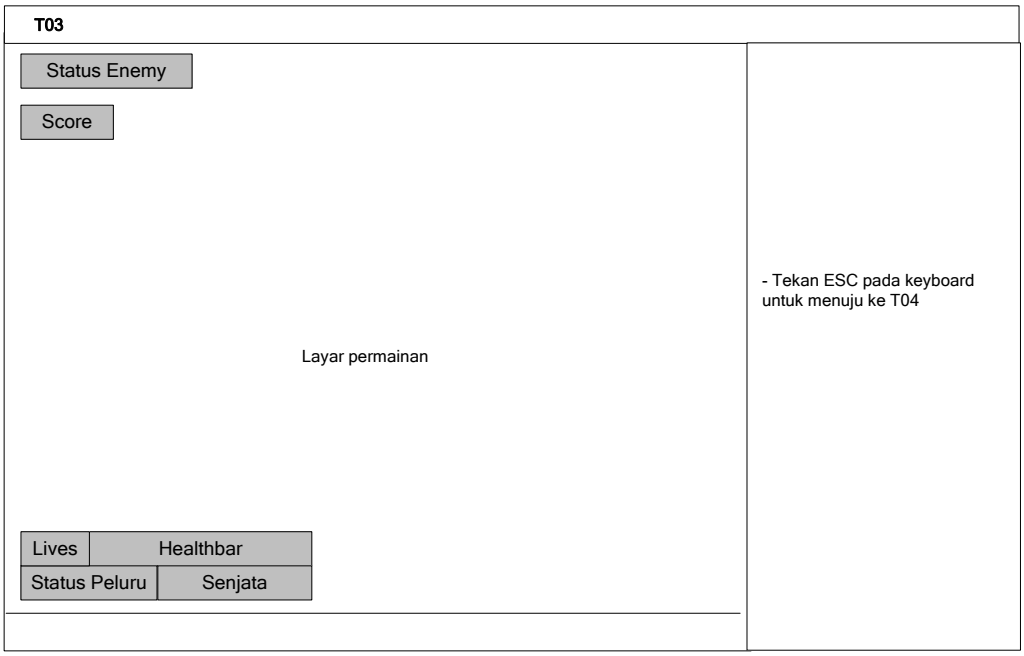
Perancangan antar muka dibutuhkan untuk mawakili keadaan dari aplikasi yang akan dibangun. Berikut ini beberapa perancangan antar muka dari *game* yang akan dibangun yaitu.



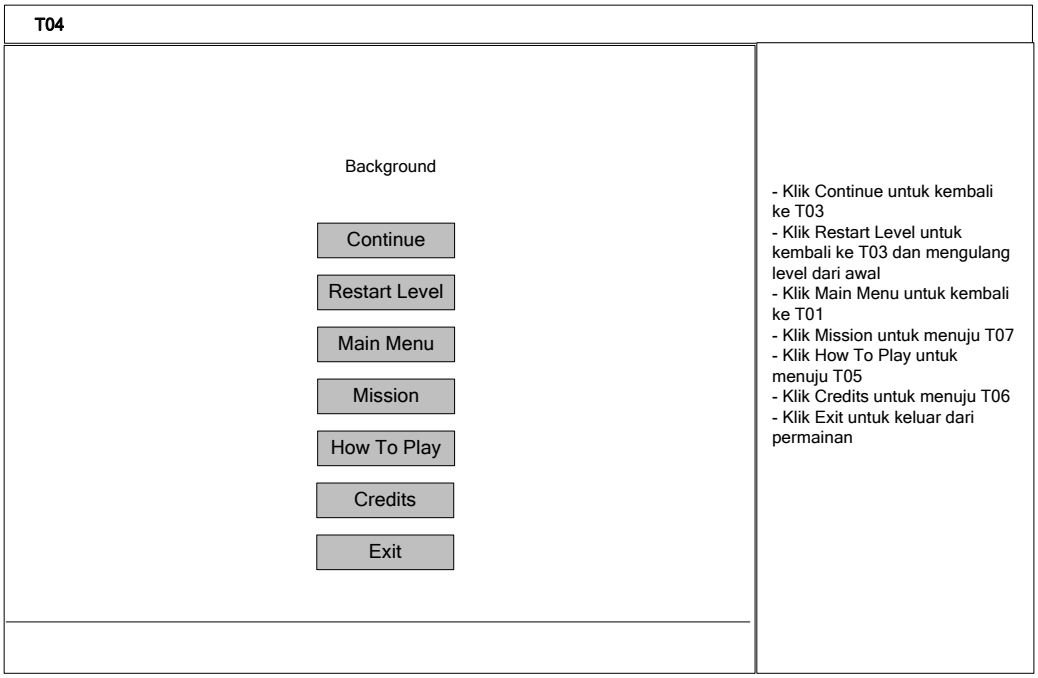
Gambar III. 51 Antar Muka Menu Utama



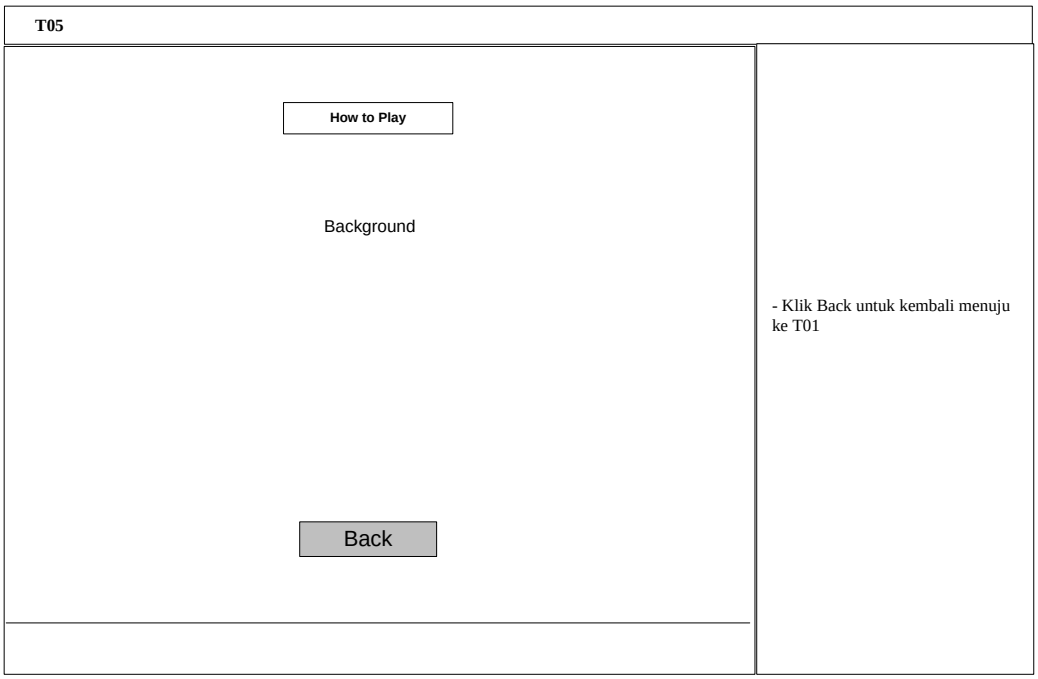
Gambar III. 52 Antar Muka *Scene Video*



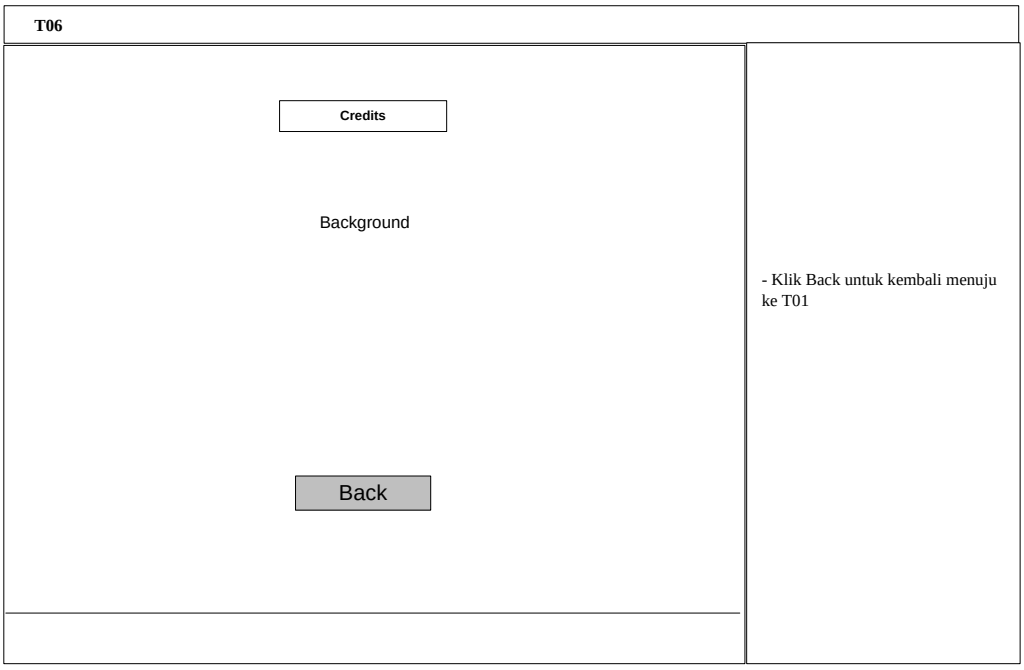
Gambar III. 53 Antar Muka Permainan



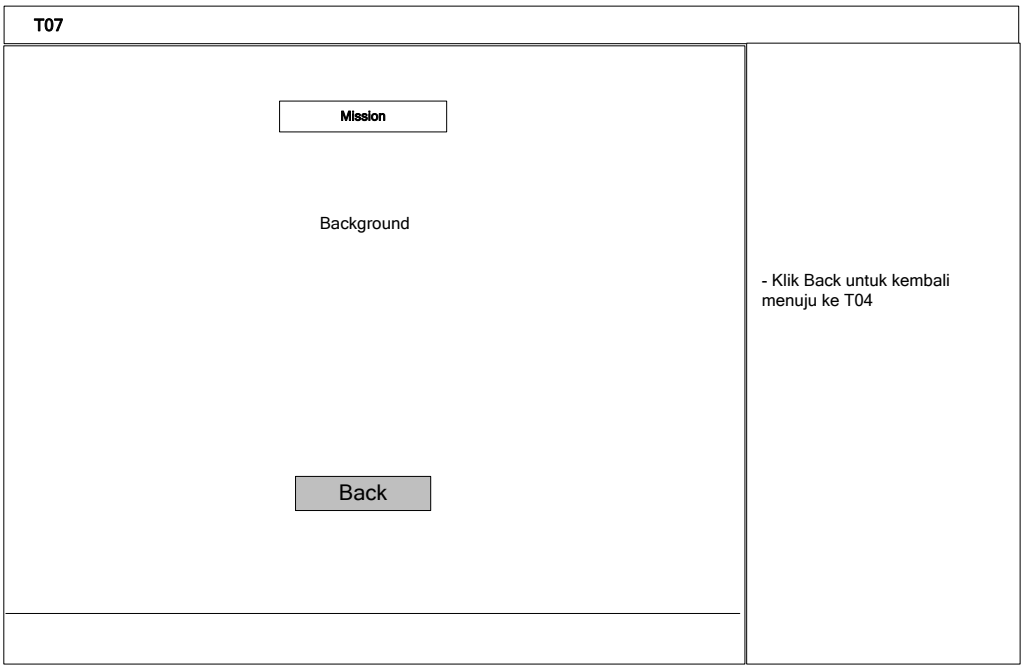
Gambar III. 54 Antar Muka *Pause Menu*



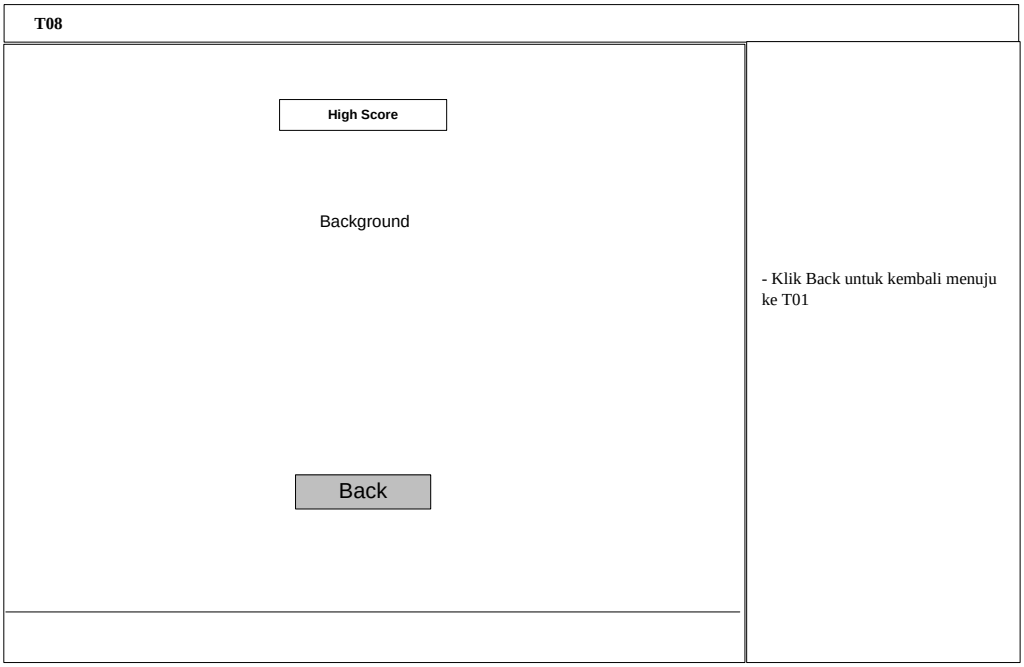
Gambar III. 55 Antar Muka *How To Play*



Gambar III. 56 Antar Muka *Credits*



Gambar III. 57 Antar Muka *Mission*

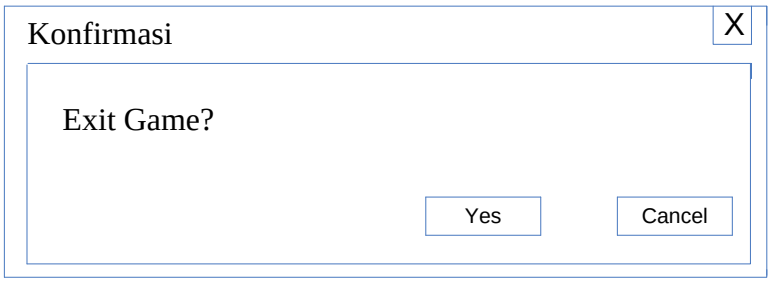


Gambar III. 58 Antar Muka *High Score*

III.6.3 Perancangan Pesan

Perancangan pesan muncul sebagai informasi kepada pengguna dalam mengoperasikan sistem.

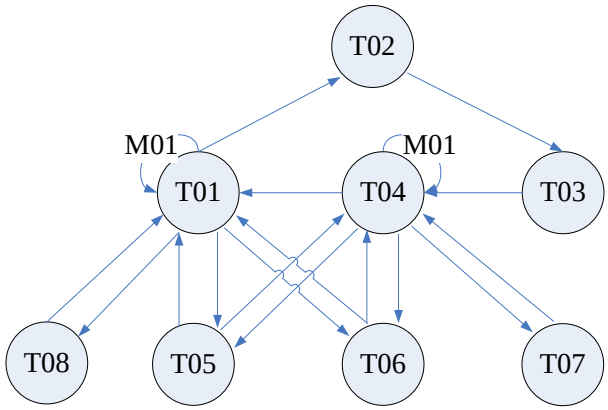
M01:



Gambar III. 59 Perancangan Pesan M01

III.6.4 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.



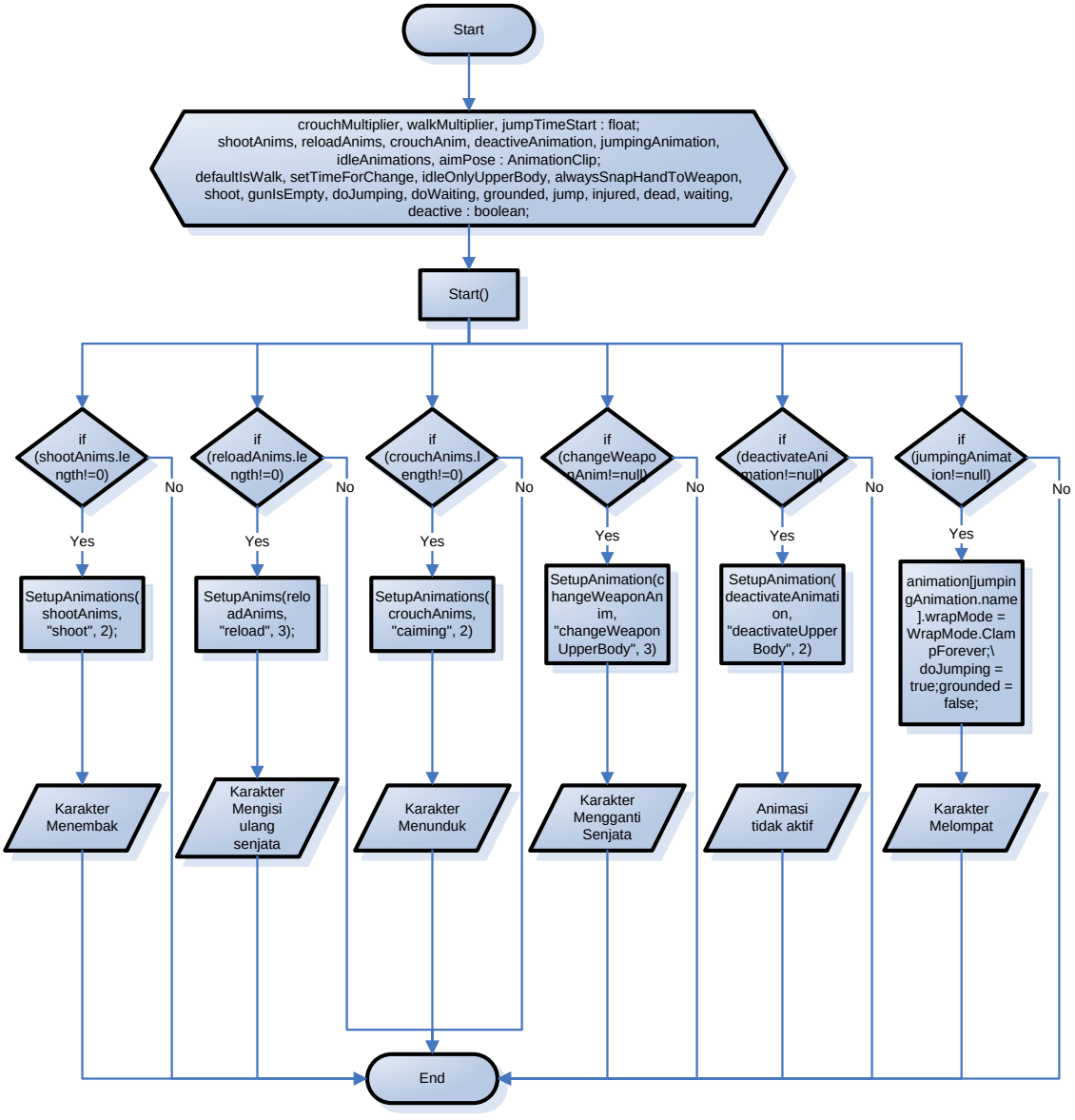
Gambar III. 60 Jaringan Semantik

III.7 Perancangan *Method*

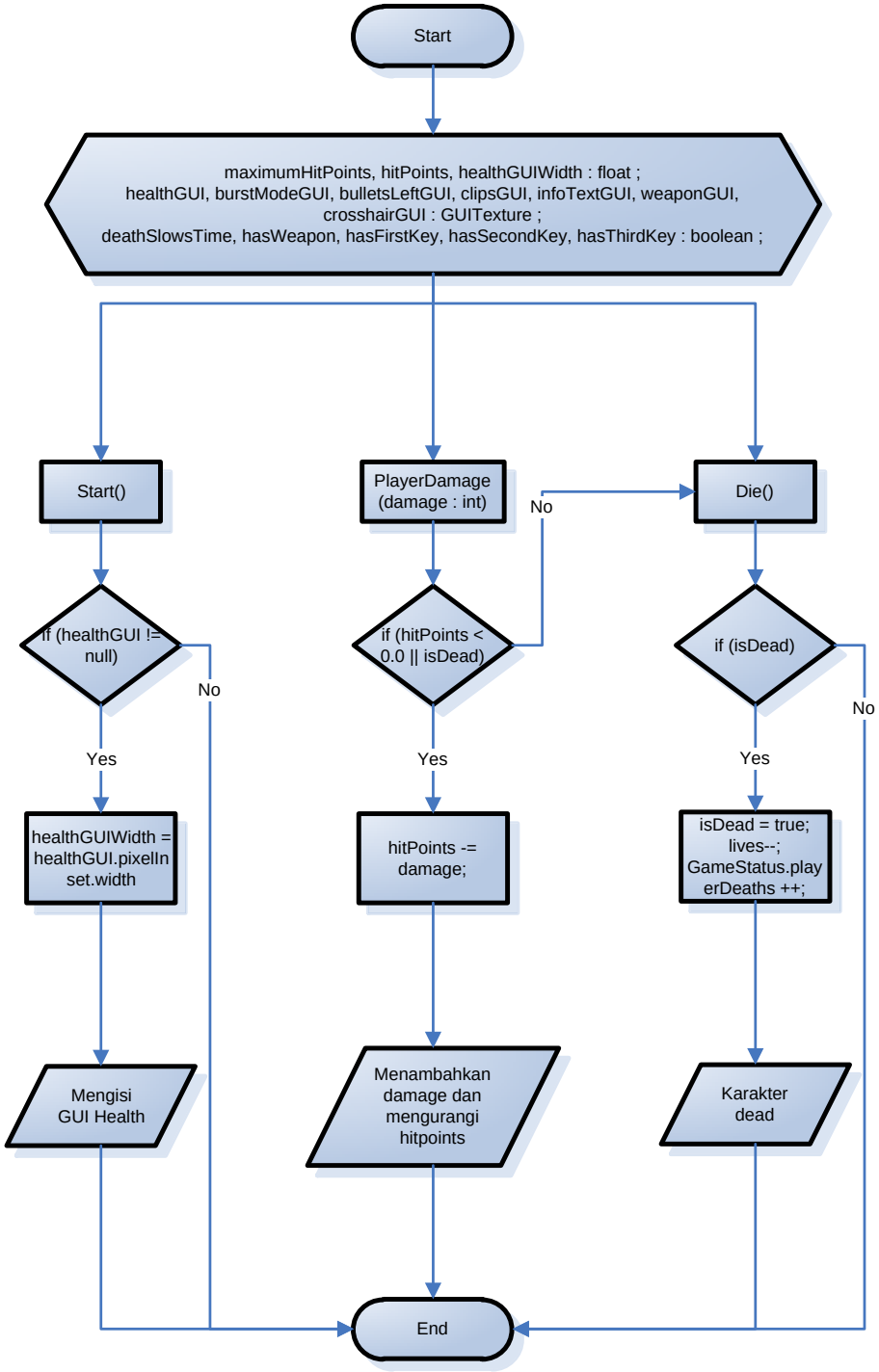
Perancangan *method* merupakan perancangan yang berfungsi untuk mendeskripsikan *method-method* yang berada di dalam aplikasi. *Method* dapat

dipanggil dengan menyertakan variabel, baik hanya satu variabel, banyak variabel atau bahkan tidak ada sama sekali .Adapun *method-method* yang terdapat dalam *game action* 3D *The Walker* yang akan dibangun dijelaskan melalui *flowchart* sebagai berikut.

III.7.1 *Method Player*

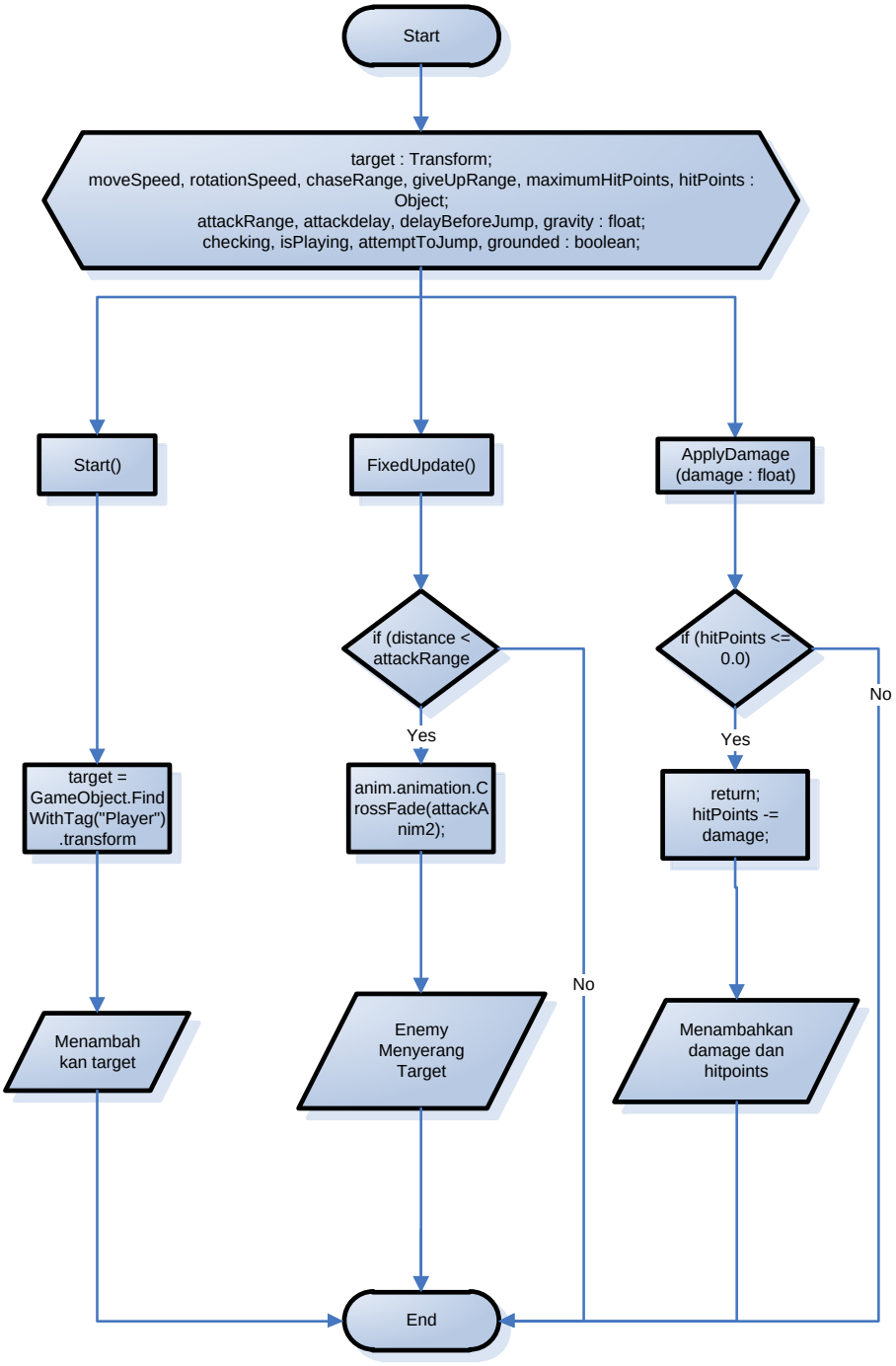


Gambar III. 61 *Method Animation Player*



Gambar III. 62 *Method Player*

III.7.2 *Method Enemy*



Gambar III. 63 *Method Enemy*