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 terhadap *game* sejenis serta analisis berbagai kebutuhan baik fungsional maupun non-fungsional.

3.1.1 Analisis Masalah

Sejarah di Indonesia memiliki potensi yang luar biasa untuk dilestarikan dan dikenang serta di dukung oleh pengenalan yang baik. Saat ini, banyak media yang mampu untuk memperkenalkan peristiwa sejarah. Pertempuran lima hari di Semarang merupakan pertempuran terhebat pada masa transisi. Pertempuran ini terjadi pada tanggal 15 oktober 1945 sampai 20 oktober 1945. Peristiwa gugurnya pahlawan Indonesia yang berjuang melawan tentara Jepang menjadi bukti kegigihan para pejuang Indonesia untuk mempertahankan wilayahnya dari Jepang. Pertempuran ini merupakan salah satu petempuran yang terjadi sesudah kemerdekaan penuh dengan nilai sejarah yang besar dan patut untuk di ingat. Dengan di aplikasikannya ke dalam bentuk *game*, sejarah pertempuran lima hari di Semarang yang diharapkan bisa lebih mengenal dan mengenang peristiwa yang terjadi di dalamnya, serta menanamkan rasa cinta dan kebanggaan kepada tanah air suatu bangsa. Game merupakan salah satu alternatif untuk mengenalkan sejarah dan budaya, karna dengan bermain *game* diharapkan bisa lebih mengenal dari segi kronologis cerita, tokoh pejuang yang terlibat, tempat atau lokasi peristiwa kejadian serta bisa merasakan langsung berada di dalam permainan tersebut.

Game yang akan di bangun mempunyai grafis desain 3D dengan *genre First Person Shooter*. Dalam *game* 3D pengguna dapat melihat sudut pandang tiga ratus enam puluh derajat pemandangan yang ada di dalam *game* dan untuk memberikan nilai lebih di dalam *game* diperlukan model dan desain yang menarik dan artistik. *First Person Shooter* atau disingkat FPS merupakan *ganre* sebuah

game yang mengambil sudut orang pertama (*first person*). Dengan fokus sudut pandang orang pertama pemain dapat merasakan seperti berada dalam realitas lingkungan *game* yang dibangun [2].

Berdasarkan uraian latar belakang masalah di atas, maka dapat dirumuskan sebuah masalah yaitu Bagaimana membangun suatu *game* dengan *genre* FPS (*First Person Shooter*) dan desain grafis 3D yang dapat memberikan informasi tentang cerita sejarah Pertempuran Lima Hari di Semarang.

3.1 Analisis Game Sejenis

Pada analisis *game* sejenis, akan dilakukan observasi terhadap *game* yang memiliki kesamaan *gameplay* dan *genre* dengan *game* yang akan dibangun. Tujuan dari analisis ini yaitu mendapatkan konsep permainan dan perbandingan dari *game* yang telah ada.

Dalam pembahasan *game* sejenis ini, akan dibahasa *game* sejenis yang dilihat dari jenis *genre game*, maupun dari kesamaan alur cerita yang diambil. *Game* yang akan menjadi acuan disini adalah *game* Palagan Ambarawa.

Game Palagan Ambarawa

a. Pengenalan

Game Palagan Ambarawa, merupakan *game* edukasi yang mempunyai *genre first person shooter* (FPS), yang dibuat oleh beberapa mahasiswa ITS Surabaya. *Game* ini dibuat oleh 10 orang mahasiswa ITS Surabaya yang mempunyai tugas masing-masing yaitu 3 orang sebagai programer, 3 orang sebagai desin *game* 3D dan 4 orang sebagai desain konten.

Game Palagan Ambarawa merupakan *game* yang menceritakan Pertempuran Ambarawa dan Hari Infanteri, yang dimulai dengan kedatangan tentara Sekutu untuk membebaskan tawanan perang di Magelang, Jawa Tengah, pada 20 Oktober 1945. Saat itu Sekutu malah mempersenjatai para tawanan. Pertempuran Tentara Keamanan Rakyat (TKR) melawan tentara Sekutu.

Pada gambar 3.1 merupakan tampilan awal pada *game* Palagan Ambarawa.



Gambar. 3.1 Game Palagan Ambarawa

Dalam game ini player berperan sebagai sudirman yaitu pejuang yang berusaha untuk mengalahkan serangan tentara sekutu yang dipersenjatai perlataan yang canggih, dan player diharuskan melewati beberapa misi seperti merebut kembali ambarawa dari tangan sekutu.

Pada gambar 3.2 merupakan salah satu *level* yang ada di *game* palagan ambarawa.



Gambar. 3.2 Salah Satu *Level* Pada *Game* Palagan Ambarawa

a. *Gameplay*

Pada *game* Palagan Ambarawa, pemain harus menyelesaikan 4 level yaitu :

1. Pertempuran Magelang.

   Sebelum memulai pertempuran, pemain harus mencari komandan di gedung TKR, kemudian misi pertama di level 1 ini adalah pemain harus menyusup ke markas sekutu untuk mencari informasi penyerangan.

2. Insiden Desa Jambu.

   Pemain harus bertahan dari serangan pasukan sekutu, sampai menemukan jalan keluar untuk keluar dari pertempuran

3. Serangan Benteng Ambarawa.

   Pemain menyerang benteng Ambarawa dan memukul mundur pasukan sekutu menuju ambarawa, yang mengakibatkan tewasnya Letkol Isdiman di dalam peperangan.

4. Pertempuran Ambarawa.

   Pemain menghadapi tentara sekutu di Ambarawa dengan tujuan menguasai wilayah ambarawa dan mengusir tentara sekutu.

b. Skenario *Game*

Skenario dalam permainan *game* Palagan Ambarawa disusun oleh beberapa hal sebagai berikut :

1. Alur Permainan

Alur permainan pada *game* palagan ambarawa adalah sebagai berikut :

a. Pemain berada depan gedung markas TKR.
b. Pemain mencari pimpinan TKR di dalam gedung.
c. Pemain berada di magelang dan dipenuhi oleh musuh.
d. Pemain harus membunuh semua musuh untuk melanjutkan ke level selanjutnya.
e. Pemain harus menghindari dan mematikan musuh ketika berada di arena permainan.

f. Jika pemain terkena tembakan oleh musuh, maka darah / *health point* akan berkurang dan apabila habis maka permainan diulangi ke awal permainan.

Secara garis besar perbandingan *game* sejenis diatas dijelaskan dengan tabel 3.1.

Tabel 3. 1 Perbandingan *Game* Sejenis

	***Game* Palagan Ambarawa**	***Game* yang Akan Dibangun**
Genre Game	*First Person Shooter*	*First Person Shooter*
Desain Grafis	3 Dimensi (3D)	3 Dimensi (3D)
Storyline* dan *Gameplay	Menceritakan seorang pejuang yang melawan pasukan sekutu, di bawah instruksi Komandan TKR dan *gameplay* yang ada di *game* ini yaitu harus menyelesaikan misi yang ada di setiap levelnya.	Menceritakan seorang pejuang Indonesia yang mengusir pasukan Jepang dari wilayah Semarang dan *gameplay* yang ada di *game* ini yaitu harus menyelesaikan misi yang ada di setiap levelnya.
Level	Game ini terdiri dari 4 level dan misi di setiap levelnya yaitu : -Level 1 : mengusir tentara NICA dan mengambil alih kota Magelang dari NICA. -Level 2 : mengambil alih desa jambu dari tentara NICA. -Level 3 : masuk ke dalam benteng ambarawa dan menguasai ruang radio. -Level 4 : mengusir tentara NICA dari ambarawa.	Game ini terdiri dari 2 level dan misi di setiap levelnya yaitu : - Level 1 : misi yang ada di level ini adalah mencari sebuah gudang dan mengambil alih gudang dari tentara jepang. - Level 2 : melakukan pertempuran melawan tentara jepang dan mencari bendera merah putih di lawang sewu untuk menyelesaikan permainan.
Struktur Menu	*Game* ini terdiri dari beberapa menu diantaranya *new game*, *load game*, *exit game*, *resume game*, *save game* dan *exit game*.	*Game* ini terdiri daribeberapa menu diantaranya mulai permainan, petunjuk permainan, keluar permainan, kembali dan menu utama.
Karakter Musuh	Pergerakan karakter musuh yang ada di *game* ini adalah berjalan, berlari, menembak dan memukul.	Pergerakan karakter musuh yang ada di *game* pertempuran lima hari di semarang adalah berjalan, berlari

		dan menembak.
Kecerdasan Musuh	Keputusan musuh yang ada di *game* palagan ambarawa yaitu hanya bisa menyerang saja.	Kecerdasan musuh yang ada di dalam ini menggunakan *fuzzy logic* dimana mempunyai tiga keputusan yaitu menyerang, bertahan dan melarikan diri.
Karakter Pemain	Pergerakan karakter pemain yang ada di *game* palagan ambarawa adalah berjalan, berlari, menunduk, melompat, mengambil darah, mengambil peluru, mengambil senjata, menembak dan melempar bom.	Pergerakan karakter pemain yang ada di *game* pertempuran lima hari di semarang adalah berjalan, berlari, melompat, menembak, mengambil darah, membuang senjata dan mengambil senjata.
Tampilan Permainan	Tampilan permainan yang ada di *game* ini yaitu darah, peluru, nama senjata dan lives.	Tampilan Permainan yang ada di *game* pertempuran lima hari di semarang yaitu darah, nama senjata, peluru dan map.
Komponen Game	Pistol, machinegun, bom, rumah, darah, gedung, pohon.	Pistol, machinegun, darah, gudang gula, pohon, rumah, jembatang, gedung lawang sewu.
Konten Edukasi	Mengenalkan nama para pahlawan, tempat terjadinya peristiwa dan awal terjadinya pertempuran Palagan Ambarawa.	Mengenalkan nama para pahlawan, tempat terjadinya peristiwa dan awal terjadinya pertempuran 5 hari di semarang.
Berbasis	*Desktop*	*Desktop*

3.1.3 Analisis *Game* Yang Dibangun

Game Pertempuran Lima Hari di Semarang yang akan dibangun yaitu *Game* yang bergenre *first person shooter*, *Game* ini memiliki grafis 3D (3 Dimensi). *Game* ini mempunyai alur cerita yang sama seperti sejarah pertempuran lima hari di semarang.

Game dibangun dengan konsep sudut pandang orang pertama agar permainan yang ditampilkan terkesan lebih nyata, seolah-olah pemain terlibat langsung ke dalam arena permainan

Berikut ini adalah fitur-fitur yang terdapat di dalam game:

1. Grafik game 3D
2. *Single Player*
3. *Game* di angkat dari cerita salah satu sejarah pertempuran di Indonesia
4. Game ini memiliki 2 *level*
5. Game di bangun dengan genre *first person shooter*
6. Memiliki objek-objek gedung bersejarah dimana info dari objek gedung tersebut bisa ditampilkan pada saat permainan berlangsung

3.1.3.1 Storyline

Game yang dibangun *single player* atau dengan satu pemain yang dijalankan seorang *actor* yang bernama Sudanco Mirza Midharta. komandan Sudanco Mirza Midharta adalah pemimpin pasukan yang menggabungkan TKR (Tentara Keamanan Rakyat) dan pejuang-pejuang lain yang memerangi tentara Jepang. *Actor* tersebut harus menyelesaikan 2 *level* yang tiap-tiap *level* memiliki rintangan yang berbeda-beda.

Level 1, insiden ini bermula ketika 675 tentara Jepang dari Irian Barat yang sedang menuju Batavia dimana tentara jepang tersebut kehabisan persediaan logistik dalam perjalanan menuju Batavia dan transit di kota Semarang untuk menambah persediaan logistik. Transitnya 675 tentara jepang di Semarang membuat TKR (Tentara Keamanan Rakyat) dan para pejuang berjaga-jaga di sekitar gudang gula cepiring dan dalam keadaan siap tempur. Berontaknya 400 tentara jepang dan di rubahnya gudang gula menjadi gudang penyimpanan senjata menyebabkan kemarahan TKR dan pejuang indonesia hingga terjadi pertempuran. Dipukul mundurnya tentara Jepang yang berada di gudang gula cepiring dan melarikan diri menuju jatingaleh untuk bergabung dengan tentara jepang/pasukan kidobutai yang di pimpin oleh Mayor Kido untuk menyerang pusat kota semarang. Pada insiden ini *actor* diharuskan mengalahkan tentara jepang yang berada di sekitar gudang gula dan menguasai gudang gula tersebut dari tangan jepang.

Level 2, Kaburnya tentara Jepang dari gudang gula cepiring untuk bergabung dengan tentara jepang yang berada di jatingaleh dan dibunuhnya Dr Karyadi oleh tentara jepang menjadi kemarahan TKR (Tentara Keamanan Rakyat) dan pejuang

Indonesia maka terjadilah pertempuran yang terjadi di kota Semarang. Pada insiden ini *actor* akan menghadapi tentara Jepang dengan tujuan menguasai wilayah Semarang dari tangan Jepang dan mengusir tentara Jepang dari kota semarang.

3.1.3.2 Gameplay

Tugas utama pemain dalam memainkan *game* ini adalah menyelesaikan misi di setiap *level* yang dimainkan, setiap *level* yang dimainkan memiliki tingkat kesulitan yang berbeda-beda, di setiap *level* pemain akan diberikan beberapa petunjuk berupa *icon* yang apabila di ambil akan mengeluarkan informasi berupa gambar dan suara. Petunjuk ini berguna apabila pemain kesulitan mencari informasi tentang apa yang harus dilakukan ketika memainkan *game* ini. Selain mendapatkan petunjuk berupa *icon* yang berisi informasi, pemain diberikan juga 2 *item* senjata yang berbeda, senjata yang digunakan adalah senjata pistol dan senjata laras panjang yang keduanya memiliki peluru terbatas. Ketika berada di dalam arena permainan, pemain akan dihadapkan dengan beberapa musuh yang terdapat di beberapa tempat yang berbeda.

a. *Level* 1

Misi yang harus diselesaikan adalah sebagai berikut :

1. Pemain harus mencari gudang gula cepiring
2. Pemain mengambil petunjuk permainan untuk menyelesaikan misi
3. Ketika pemain kehabisan peluru, pemain harus mencari senjata yang berada di dalam arena permainan tersebut
4. Apabila pemain kehabisan darah, maka pemain bisa mengambil *icon* darah
5. Melakukan pertempuran melawan tentara jepang dan mengambil alih gudang yang di kuasai oleh tentara jepang

b. *Level* 2

Misi yang harus diselesaikan adalah sebagai berikut :

1. Pemain berada di pusat kota semarang
2. Pemain mengambil petunjuk permainan untuk menyelesaikan misi

3. Ketika pemain kehabisan peluru, pemain harus mencari senjata yang berada di dalam arena permainan tersebut
4. Apabila pemain kehabisan darah, maka pemain bisa mengambil *icon* darah
5. Melakukan pertempuran secara besar-besaran terhadap tentara Jepang dengan tujuan untuk merebut wilayah kota Semarang yang dikuasai oleh tentara Jepang dan mengambil bendera Merah Putih untuk menyelesaikan permainan.

3.1.3.3 Analisis Tingkat Kesulitan

Tingkat kesulitan akan berbeda di setiap levelnya dilihat dari jumlah musuh, kekuatan musuh, dan selang waktu setiap tembakan yang dilakukan musuh. Di bawah ini adalah perbandingan tingkat kesulitan tiap *level*, dapat dilihat pada tabel 3.2.

Tabel 3.2 Tingkat Kesulitan Tiap *Level*

Keterangan	***Level 1***	***Level 2***
Jumlah musuh	25	30
Darah	100	100
Kecepatan musuh	5 pixel	10 pixel
Shoot range	15 pixel	20 pixel
Attack range	30 pixel	40 pixel
Shoot angle	5 pixel	8 pixel

3.1.3.4 Analisis Scoring

Pada *game* pertempuran lima hari di Semarang ini tidak memiliki *score* satu-satunya penilaian yang ada adalah sukses atau tidaknya misi dalam setiap *level*.

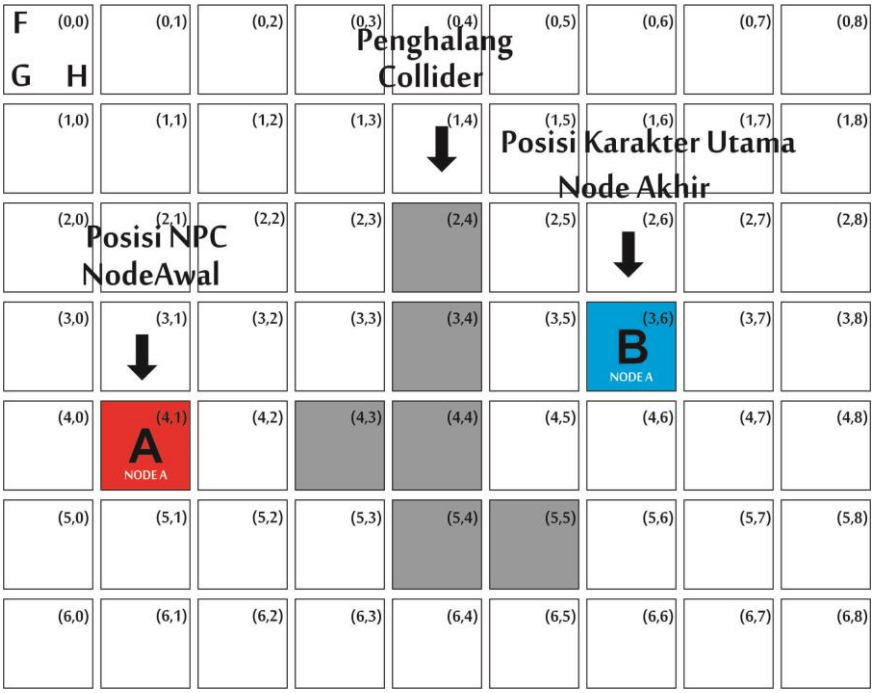
3.2 Analisis Metode

Pada Analisis Metode akan dijelaskan tentang algoritma A* *(A Star)* dan *Fuzzy Logic* yang di gunakan di dalam *game*.

3.2.1 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 first person shooter* 3D sejarah Pertempuran Lima Hari di Semarang lebih jelasnya sebagai berikut.



Gambar 3.3 Ilustrasi A*

Pada gambar 3.3 diasumsikan bahwa wilayah pencarian dibagi-bagi menjadi kotak-kotak atau disebut *node*, *node* yang berwarna merah atau node (4,1) merupakan *node* A (posisi NPC) dan *node* yang berwarna biru atau *node* (3,6) merupakan *node* B (posisi karakter utama), sedangkan *node* yang berwarna abu-abu {(2,4), (3,4), (4,3), (4,4), (5,4), (5,5)} 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:

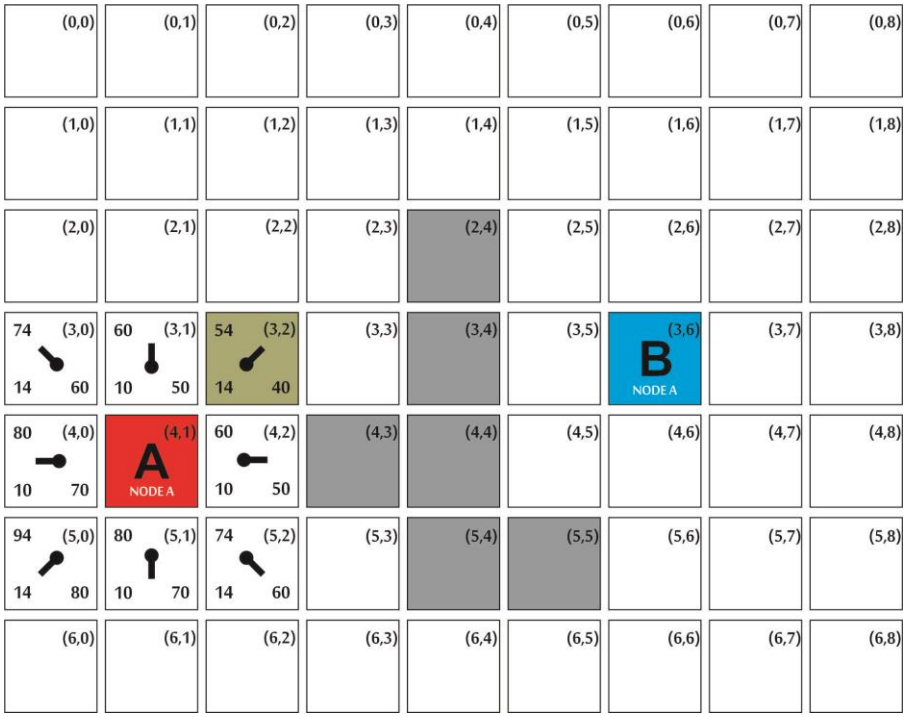
H = 10*(abs(*current*X-*target*X) + abs(*current*Y-*target*Y)) (3.1)

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.



Gambar 3.4 Ilustrasi Iterasi Pertama

Pada 3.4 merupakan iterasi pertama *node* (4,1) dimasukkan ke dalam *open list*, dan selanjutnya memeriksa *neighbor node* atau *node* yang berdekatan dengan *node* (4,1) yaitu *node* {(3,0), (3,1), (3,2), (4,0), (4,2), (5,0), (5,1), (5,2)} dan dimasukkan ke dalam *open list.* Hapus *node* (4,1) dari *open list* dan masukkan ke dalam *closed list.* Lalu dari *node-node* yang berada di dalam *open list* pilih *node* dengan nilai F yang paling rendah, yaitu *node* (3,2) dengan nilai F = 54 hasil dari penjumlahan nilai G=14 dan nilai H=40, setelah itu diberikan tanda dengan kotak berwarna coklat.

Iterasi pertama menghasilkan *node-node* di dalam *open list* dan *closed list* dijelaskan pada tabe 3.3.

Tabel 3.3 Tabel Perhitungan Iterasi Pertama

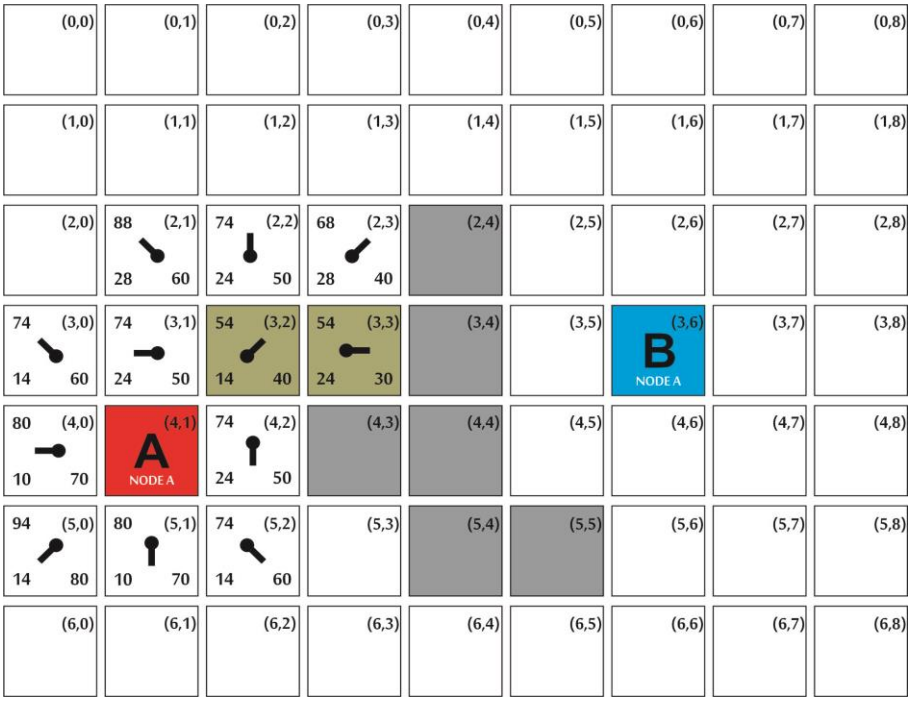
Open list	*Parent*	*Closed list*	*Current node*
(3,0)	(4,1)	(4,1)	(3,2)
(3,1)	(4,1)		
(3,2)	(4,1)		
(4,0)	(4,1)		
(4,2)	(4,1)		
(5,0)	(4,1)		

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

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 G dan H serta merubah arah *pointer,* jika tidak maka jangan lakukan apa-apa.

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



Gambar 3.5 Ilustrasi Iterasi Kedua

Pada gamber 3.5 terlihat ada 6 *node* yang masuk ke dalam *open list* yaitu *node* {(2,1), (2,2), (2,3), (3,1), (3,3), (4,2)}. Sedangkan untuk *node* (4,1), (3,2) merupakan *node* c*losed list* dan *node* (4,3) merupakan penghalang atau *collider.* Lalu dari *node-node* yang berada di dalam *open list* pilih *node* dengan nilai F yang paling rendah, yaitu *node* (3,3) dengan nilai F = 54 hasil dari penjumlahan nilai G=24 dan nilai H=30, setelah itu diberikan tanda dengan kotak berwarna coklat.

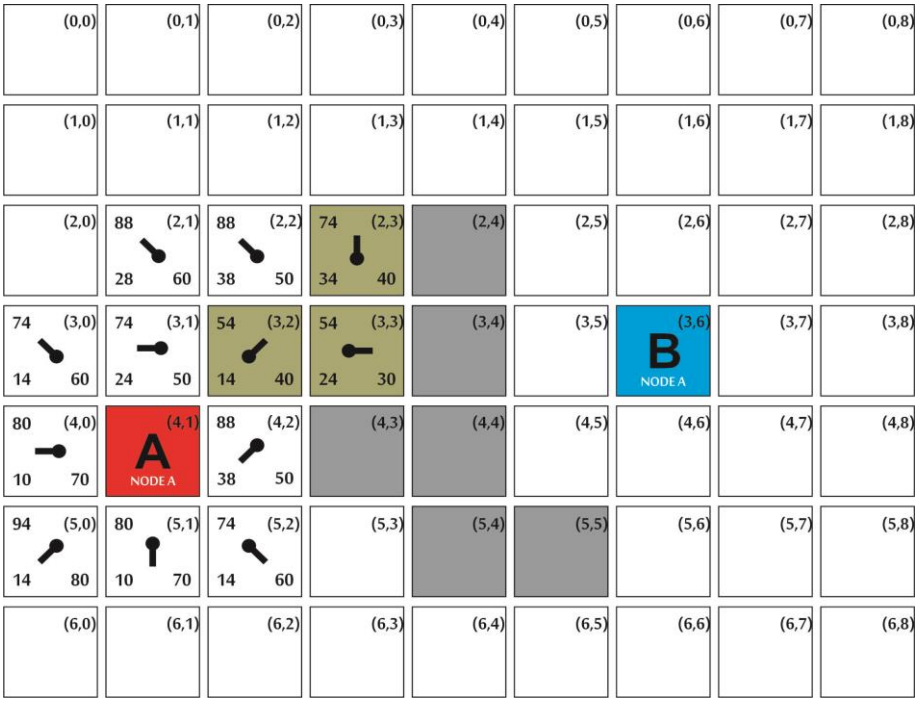
Iterasi kedua menghasilkan *node-node* di dalam *open list* dan *closed list* dijelaskan pada tabel 3.4.

Tabel 3.4 Tabel Perhitungan Iterasi Kedua

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

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

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



Gambar 3.6 Ilustrasi Iterasi Ketiga

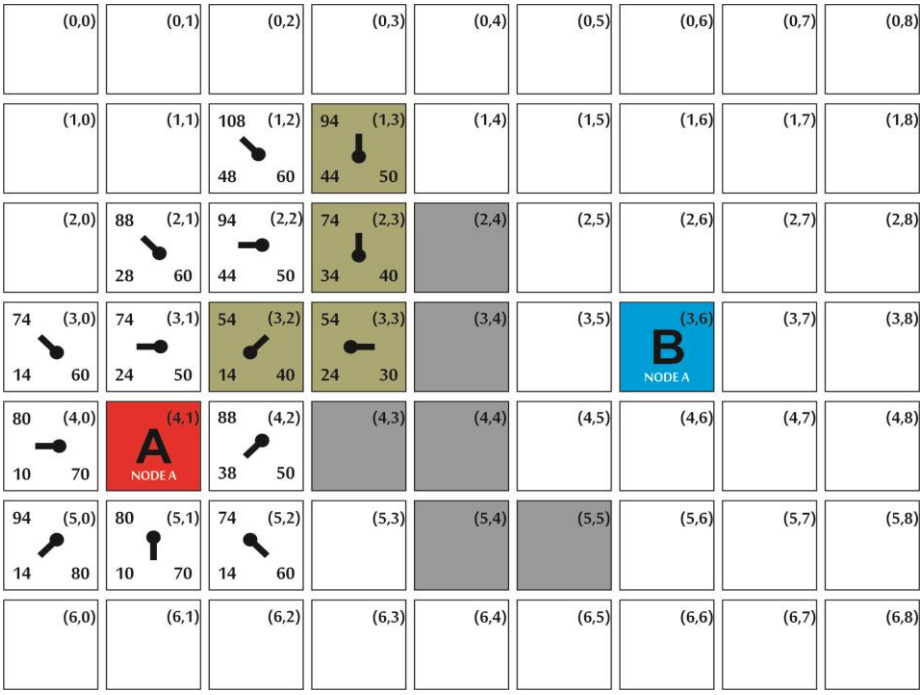
Pada gamber 3.6 terlihat ada 2 *node* yang masuk ke dalam *open list* yaitu *node* {(2,2), (2,3)}. Sedangkan untuk *node* (3,2) merupakan *node* c*losed list* dan *node* (2,4), (3,4), (4,3), (4,4) merupakan penghalang atau *collider*. karena karakter tidak akan bisa bergerak secara *diagonal* menembus dinding atau *collision* sehingga *node* (4,2) otomatis menjadi *unwalkable*. Lalu dari *node-node* yang berada di dalam *open list* pilih *node* dengan nilai F yang paling rendah, yaitu *node* (2,3) dengan nilai F = 74 hasil dari penjumlahan nilai G=34 dan nilai H=40, setelah itu diberikan tanda dengan kotak berwarna coklat.

Iterasi ketiga menghasilkan *node-node* di dalam *open list* dan *closed list* dijelaskan pada table 3.5.

Tabel 3.5 Tabel Perhitungan Iterasi Ketiga

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

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



Gambar 3.7 Ilustrasi Iterasi Keempat

Pada gambar 3.7 diatas *node* (1,4) tidak dihitung atau tidak masuk ke dalam *open list*, karena karakter tidak akan bisa bergerak secara *diagonal* menembus dinding atau *collision* sehingga *node* (1,4) otomatis menjadi *unwalkable*.

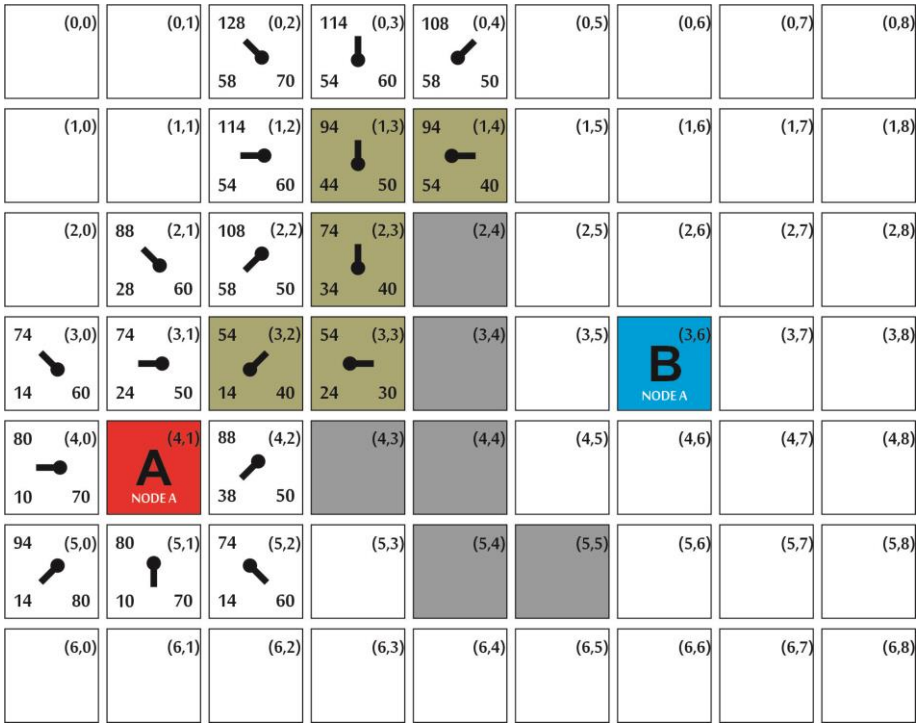
Pada *node* (2,2) dan (1,3) mempunyai nilai yang sama yaitu dengan nilai F=94 hasil dari penjumlahan nilai G=44 dan nilai H=50, dapat dilihat pada gambar (3,7) di atas, maka sistem akan memilih *node* (1,3) karena yang terakhir masuk ke dalam *open list*. Yang berarti *current node* selanjutnya adalah *node* (1,3).

Iterasi keempat menghasilkan *node-node* di dalam *open list* dan *closed list* dijelaskan pada tabel 3.6.

Tabel 3.6 Tabel Perhitungan Iterasi Keempat

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

Perhitungan pada iterasi kelima dengan *current node* berada pada *node* (1,3).



Gambar 3.8 Ilustrasi Iterasi Kelima

Pada gamber 3.8 terlihat ada 6 *node* yang masuk ke dalam *open list* yaitu *node* {(0,2), (0,3), (0,4), (1,2), (1,4), (2,2)}. Sedangkan untuk *node* (2,3) merupakan *node* c*losed list* dan *node* (2,4) merupakan penghalang atau *collider.* Lalu dari *node-node* yang berada di dalam *open list* pilih *node* dengan nilai F yang paling rendah, yaitu *node* (1,4) dengan nilai F = 94 hasil dari penjumlahan nilai G=54 dan nilai H=40, setelah itu diberikan tanda dengan kotak berwarna coklat.

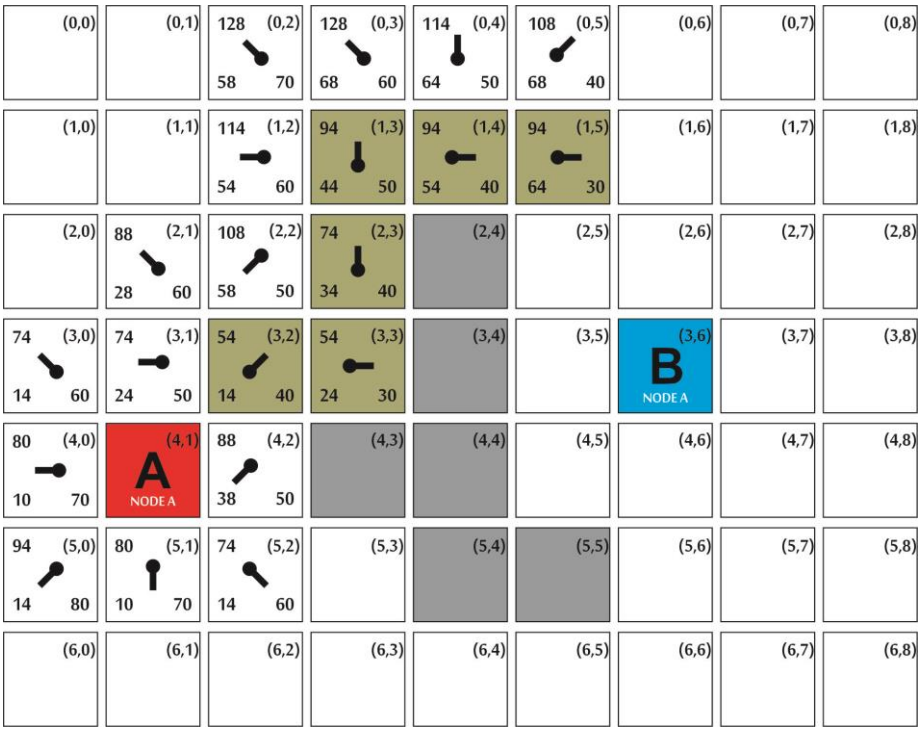
Iterasi keempat menghasilkan *node-node* di dalam *open list* dan *closed list* dijelaskan pada tabel 3.7.

Tabel 3.7 Tabel Perhitungan Iterasi Kelima

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

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

Perhitungan pada iterasi keenam dengan *current node* berada pada *node* (1,4).



Gambar 3.9 Ilustrasi Iterasi Keenam

Pada gamber 3.9 terlihat ada 4 *node* yang masuk ke dalam *open list* yaitu *node* {(0,3), (0,4), (0,5), (1,5)}. Sedangkan untuk *node* (1,3),(2,3) merupakan *node* c*losed list* dan *node* (2,4) merupakan penghalang atau *collider*. karena karakter tidak akan bisa bergerak secara *diagonal* menembus dinding atau *collision* sehingga *node* (2,5) otomatis menjadi *unwalkable*. Lalu dari *node-node* yang berada di dalam *open list* pilih *node* dengan nilai F yang paling rendah, yaitu

node (1,5) dengan nilai F = 94 hasil dari penjumlahan nilai G=64 dan nilai H=30, setelah itu diberikan tanda dengan kotak berwarna coklat

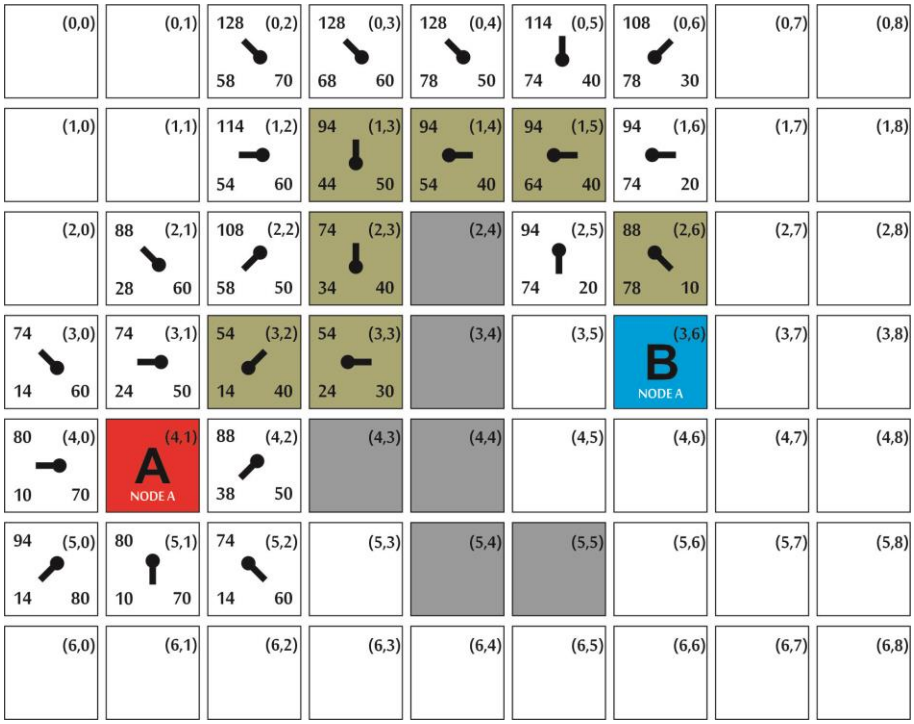
Iterasi keenam menghasilkan *node-node* di dalam *open list* dan *closed list* dijelaskan pada tabel 3.8.

Tabel 3.8 Tabel Perhitungan Iterasi Keenam

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

Penentuan jalan terbaik dengan melakukan pemeriksaan dimulai dari *target* atau *node* B, dalam ilustrasi adalah kotak berwarna biru. Dengan mengikuti arah *pointer* yang berarti akan membawanya kembali ke *node* awal, maka akan didapatkan jalan terbaiknya, Maka jika diimplementasikan pada *game*, nantinya musuh / NPC akan melewati halangan dan akhirnya akan menemukan rute terpendek ke tujuan, lebih jelasnya seperti gambar berikut.

Perhitungan pada iterasi ketujuh dengan *current node* berada pada *node* (1,5)



Gambar 3.10 Ilustrasi Iterasi Ketujuh

Pada gamber 3.10 Penentuan jalan terbaik dengan melakukan pemeriksaan dimulai dari *target* atau *node* B, terlihat ada 6 *node* yang masuk ke dalam *open list* yaitu *node* {(0,4), (0,5), (0,6), (1,6), (2,5), (2,6)}. Sedangkan untuk *node* (1,4) merupakan *node* c*losed list* dan *node* (2,4) merupakan penghalang atau *collider*. Lalu dari *node-node* yang berada di dalam *open list* pilih *node* dengan nilai F yang paling rendah, yaitu *node* (2,6) dengan nilai F = 88 hasil dari penjumlahan nilai G=78 dan nilai H=10, setelah itu diberikan tanda dengan kotak berwarna coklat.

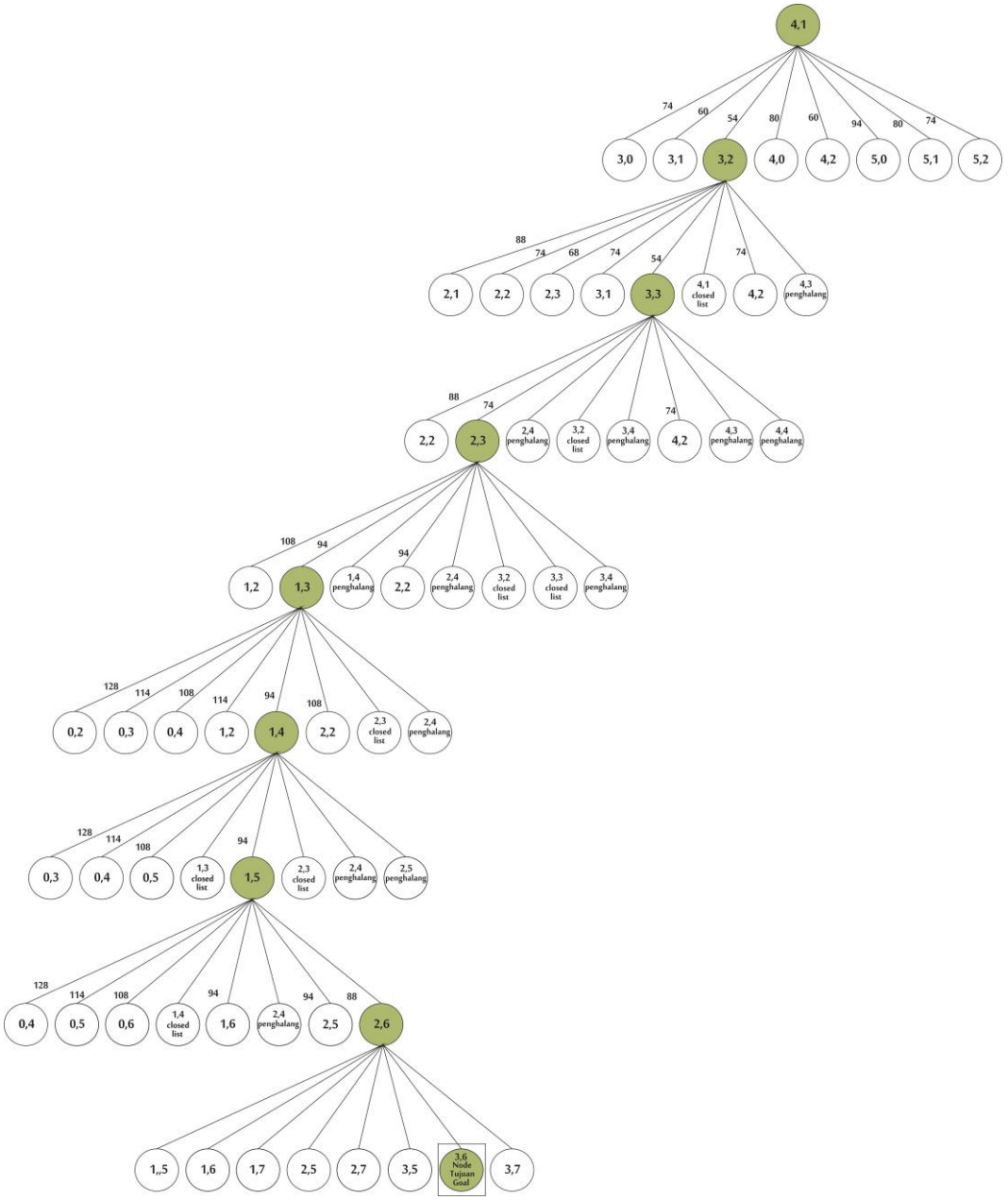
Iterasi ketujuh menghasilkan *node-node* di dalam *open list* dan *closed list* dijelaskan pada tabel 3.9.

Tabel 3.9 Tabel Perhitungan Iterasi Ketujuh

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

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

Langkah-langkah perhitungan dari contoh diatas dapat diilustrasikan ke dalam *tree* dapat dilihat pada gambar 3.11



Gambar 3.11 Ilustrasi *Tree* Pencarian Jalan Terbaik

3.2.2 Analisis *Fuzzy Logic*

Fuzzy Logic (logika fuzzy) di dalam *game* akan digunakan untuk menentukan variasi perilaku yang akan dihasilkan oleh NPC (*Non Playable Character*) untuk memberikan *abilities* dalam menentukan keputusan untuk menyerang, bertahan dan melarikan diri. Cara kerja *Fuzzy Logic* di dalam *game* pertempuran lima hari di Semarang lebih jelasnya sebagai berikut.

Logika fuzzy yang digunakan adalah metode sugeno. Analisis *fuzzy logic* tahap-tahapnya adalah sebagai berikut.

3.2.2.1 Variabel *Fuzzy*

Didalam *game* ini digunakan 3 variabel dalam fungsi *fuzzy*, variabel tersebut yaitu:

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

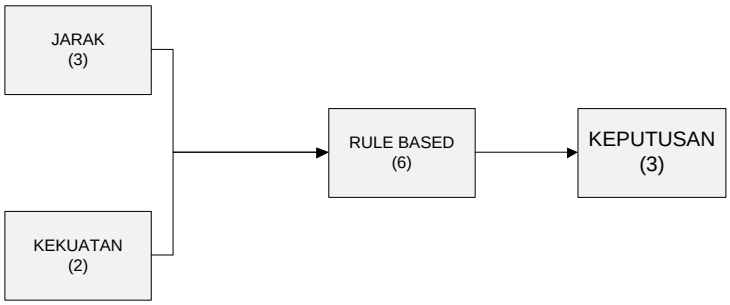
3.2.2.2 Nilai linguistik

Dari tiga variabel yang digunakan, maka nilai linguistiknya sebagai berikut:

1. Variabel Jarak, terbagi menjadi 3 *input* yaitu : Dekat, Sedang, Jauh.
    1. Dekat
    2. Sedang
    3. Jauh
2. Variabel Kekuatan, terbagi menjadi 2 *input* yaitu : Lemah dan Kuat.
    1. Lemah
    2. Kuat
3. Variabel Keputusan, terbagi menjadi 3 *output* yaitu :
    1. Menyerang
    2. Bertahan
    3. Melarikan Diri.

3.2.2.3 Fuzzyfikasi

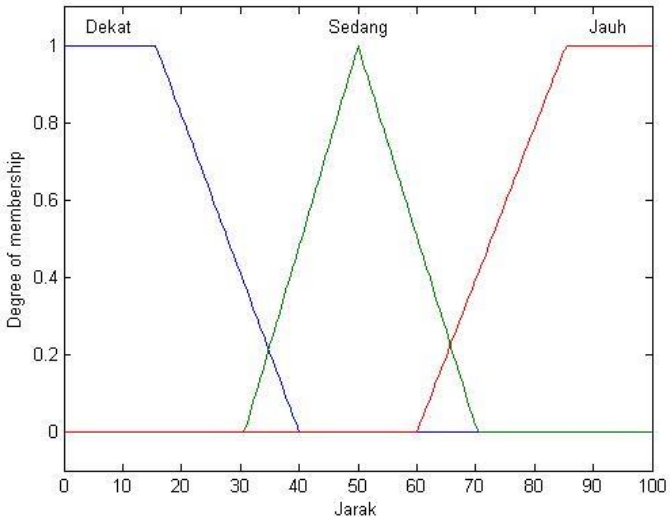
Fuzzyfikasi merupakan proses memetakan nilai *crisp* (numerik) ke dalam himpunan *fuzzy* dan menentukan derajat keanggotaannya. Secara garis besar pemetaan nilai *crisp* ke dalam himpunan *fuzzy* dijelaskan dengan gambar 3.12.



Gambar 3.12 *Fuzzy Interface System*

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 3.13 Grafik *input* jarak

Pada gambar 3.13 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. Perhitungan nilai

fuzzyfikasi didapatkan dari beberapa fungsi, fungsi yang digunakan pada Variable Jarak ada 2 yaitu fungsi Trapesium dan Fungsi Segitiga.

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

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

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

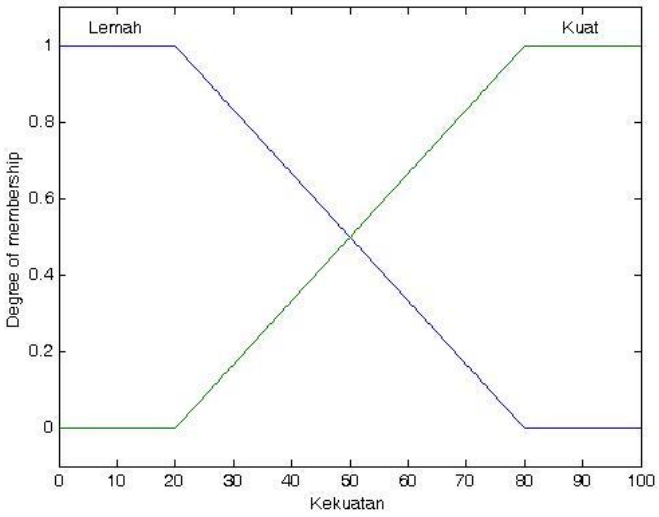
$$\mu Sedang[x]\begin{cases}0; x \leq 30\ atau\ x \geq 70 \\ \dfrac{(x-30)}{20}; 30 \leq x \leq 50 \\ \dfrac{(70-x)}{20}; 50 \leq x \leq 70\end{cases} \quad (3.3)$$

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

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

2. Variabel Kekuatan, terbagi menjadi 2 input yaitu : Lemah dan Kuat. *Range* nilai variabel Kekuatan diantara 0 – 100 akan dijelaskan sebagai berikut :

    1. Lemah = 0 - 80
    2. Kuat = 20 – 100



Gambar 3.14 Grafik *Input* Kekuatan

Pada gambar 3.14 Menunjukan sebuah grafik trapesium Kekuatan yang mempunyai range nilai dari 0 – 100, setiap nilai linguistik dari variabel kekuatan seperti lemah dan kuat mempunyai nilai *fuzzyfikasi* yang berbeda-beda.

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

$$\mu Lemah[y]\begin{cases}1; y \leq 20\\ \dfrac{80-y}{60}\\ 0; y \geq 80\end{cases} ; 20 \leq y \leq 80 \qquad (3.5)$$

b. Variabel linguistik Kuat yang memiliki *range* nilai antara 20 – 100 nilai *fuzzyfikasi*nya dihitung berdasarkan fungsi trapesium, berikut persamaannya :

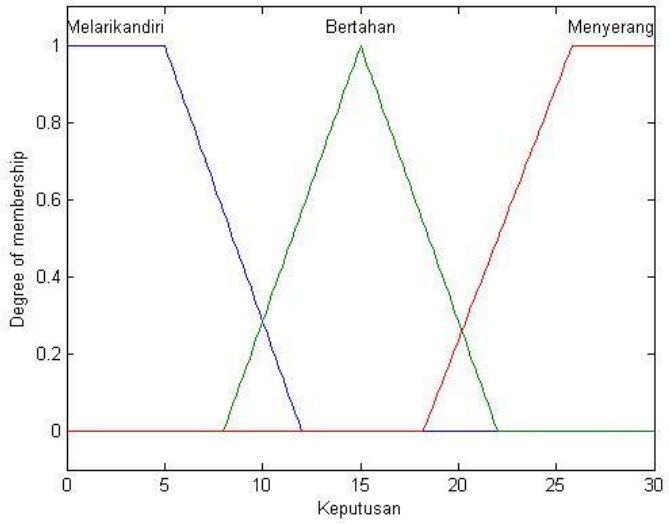
$$\mu Kuat[y]\begin{cases}0; y \leq 20\\ \dfrac{y-20}{60}; 20 \leq y \leq 80\\ 1; y \geq 80\end{cases} \quad (3.6)$$

3. Variabel Keputusan, yaitu Menyerang, Bertahan dan Melarikan Diri. *Range* nilai variabel keputusan diantara 0-30, akan dijelaskan sebagai berikut :

   1. Melarikan diri = 0 - 12

   2. Bertahan = 8 - 22

   3. Menyerang = 18 - 30



Gambar 3.15 Grafik Keputusan

Pada gambar 3.15 Menunjukan sebuah grafik keputusan yang mempunyai *range* nilai dari 0 – 30, setiap nilai lingustik seperti melarikan diri, bertahan dan menyerang mempunyai nilai *fuzzyfikasi* yang berbeda-beda.

a. Variabel linguistik Melarikan Diri yang memiliki *range* nilai antara 0 – 12 nilai *fuzzyfikasi*nya dihitung berdasarkan fungsi trapesium, berikut persamaannya :

$$\mu MelarikanDiri[z]\begin{cases}1; z \leq 6\\ \dfrac{12-z}{6}; 6 \leq z \leq 12\\ 0; z \geq 12\end{cases} \quad (3.7)$$

b. Variabel linguistik Bertahan yang memiliki *range* nilai antara 8 – 22 nilai *fuzzyfikasi*nya dihitung berdasarkan fungsi segitiga, berikut persamaannya :

$$\mu Bertahan[x]\begin{cases} 0; z \leq 8 \; atau \; z \geq 22 \\ \dfrac{(z-8)}{7}; 8 \leq z \leq 15 \\ \dfrac{(22-z)}{7}; 15 \leq z \leq 22 \end{cases} \quad (3.8)$$

c. Variabel linguistik Menyerang yang memiliki *range* nilai antara 18 – 30 nilai *fuzzyfikasi*nya dihitung berdasarkan fungsi trapesium, berikut persamaannya :

$$\mu Menyerang[z]\begin{cases} 0; z \leq 18 \\ \dfrac{z-18}{6}; 18 \leq z \leq 24 \\ 1; z \geq 24 \end{cases} \quad (3.9)$$

3.2.2.4 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* Kuat) *then* (Perilaku NPC *is* Menyerang)
3. *If* (Jarak *is* Sedang) *and* (Kekuatan *is* Lemah) *then* (Perilaku NPC *is* Melarikan diri)
4. *If* (Jarak *is* Sedang) *and* (Kekuatan *is* Kuat) *then* (Perilaku NPC *is* Menyerang)
5. *If* (Jarak *is* Jauh) *and* (Kekuatan *is* Lemah) *then* (Perilaku NPC *is* Bertahan)

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

3.2.2.5 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*).

$$z^* = \frac{\sum \mu i.zi}{\sum \mu i} \qquad (3.10)$$

z*=titik pusat nilai *crisp*

μi =derajat keanggotaan masing-masing variabel

zi = nilai domain pada variabel linguistik z.

Contoh Kasus

Misal, apabila jarak memiliki nilai [35] dan Kekuatan memiliki nilai [75], maka tahapan-tahapan atau proses agregasi yang menentukan keluaran sesuai dengan *rules* yang sudah dibuat akan dijelaskan seperti berikut :

1. Fuzzifikasi

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

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

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

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

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

1. Fungsi keanggotaan dekat dengan nilai jarak [35] = 0,2
2. Fungsi keanggotaan sedang dengan nilai jarak [35] = 0,25
3. Fungsi keanggotaan jauh dengan nilai jarak [35] = 0

Perhitungan *Fuzzyfikasi* untuk Kekuatan dari setiap variabel lingustik (Lemah dan Kuat) apabila nilai Kekuatan [75] adalah sebagai berikut :

$$\mu Lemah[75] = \frac{80 - 75}{60} = 0{,}08$$

$$\mu Kuat[75] = \frac{75 - 20}{60} = 0{,}91$$

Dari hasil perhitungan untuk mencari nilai fungsi kenaggotaan menggunakan fungsi trapesium, didapatkan nilai fungsi keanggotaan dari masing-masing variabel lingustik yaitu :

1. Fungsi keanggotaan lemah dengan nilai kekuatan [75] = 0,08
2. Fungsi keanggotaan kuat dengan nilai kekuatan [75] = 0,91

2. Implikasi

Setelah nilai derajat keanggotaan dari variabel jarak dan variable kekuatan didapatkan, maka nilai-nilai derajat tersebut harus dibandingkan sesuai dengan *Rule* dari variabel jarak dan variabel kekuatan, tujuannya untuk mencari nilai minimum untuk setiap *rule* yang dihitung.

a. *Rule* 1

If (Jarak *is* Dekat) *and* (Kekuatan *is* Lemah) *then* (Perilaku NPC *is* Melarikan diri)

→ Jarak (Dekat) = 0.2 dan Kekuatan (Lemah) = 0.08

Min (0.2 , 0.08) = 0.08

b. *Rule* 2

If (Jarak *is* Dekat) *and* (Kekuatan *is* Kuat) *then* (Perilaku NPC *is* Menyerang)

→ Jarak (Dekat) = 0.2 dan Kekuatan (Kuat) = 0.91

Min (0.2 , 0.91) = 0.2

c. *Rule* 3

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

→ Jarak (Sedang) = 0.25 dan Kekuatan (Lemah) = 0.08

Min (0.25 , 0.08) = 0.08

d. *Rule* 4

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

→ Jarak (Sedang) = 0.25 dan Kekuatan (Kuat) = 0.91

Min (0.25 , 0.91) = 0.25

e. *Rule* 5

If (Jarak *is* Jauh) *and* (Kekuatan *is* Lemah) *then* (Perilaku NPC *is* Bertahan)

→ Jarak (Jauh) = 0 dan Kekuatan (Lemah) = 0.08

Min (0 , 0.08) = 0

f. *Rule* 6

If (Jarak *is* Jauh) *and* (Kekuatan *is* Kuat) *then* (Perilaku NPC *is* Menyerang)

→ Jarak (Jauh) = 0 dan Kekuatan (Kuat) = 0.91

Min (0 , 0.91) = 0

3. Defuzzy

Setelah nilai derajat keanggotaan didapatkan dengan cara implikasi, maka langkah berikutnya adalah menentukan variabel linguistik z dari setiap rule.

a. *Rule* 1 = Melarikan Diri

$$\frac{12 - z}{6} = 0.08$$

$Z = 12 – 0.48 = 11{,}52$

b. *Rule* 2 = Menyerang

$$\frac{z - 18}{6} = 0.2$$

Z = 18 – 1,2 = 16,8

c. *Rule* 3 = Melarikan Diri

$$\frac{12 - z}{6} = 0.08$$

Z= 12 – 0.48 = 11.52

d. *Rule* 4 =Menyerang

$$\frac{z - 18}{6} = 0{,}25$$

Z = 18 – 1,5 = 16,5

e. *Rule* 5 = Bertahan

$$\frac{z - 8}{7} = 0$$

Z = 8 – 0 = 8

f. *Rule* 6 = Menyerang

$$\frac{z - 18}{6} = 0$$

Z = 18 – 0 = 18

Setelah mendapatkan nilai dari setiap *rules* maka langkah selanjutnya adalah menentukan *Defuzzyfikasi* untuk proses pemetaan besaran dari himpunan *fuzzy set* yang dihasilkan ke dalam bentuk nilai *crisp* dengan metode rata-rata (*average*) untuk menentukan variabel keputusan.

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

$$\mathbf{Z}^* = \frac{0{,}08 \text{ x } 11{,}5 + 0{,}2 \text{ x } 16{,}8 + 0{,}08 \text{ x } 11{,}5 + 0{,}25 \text{ x } 16{,}5 + 0 \text{ x } 8 + 0 \text{ x } 18}{0{,}08 + 0{,}2 + 0{,}08 + 0{,}25 + 0 + 0} = \frac{9{,}32}{0{,}61}$$

Z* = 15,27

Jadi nilai yang dihasilkan berdasarkan variabel keputusan antara *range* 0-12 adalah melarikan diri, *range* 8-20 adalah bertahan dan *range* 18-30 adalah menyerang. Sesuai dengan contoh kasus yang dihasilkan dari nilai jarak [35] dan

Kekuatan [75] dengan nilai z* yang didapatkan yaitu **15,27** jadi keputusannya adalah **bertahan**.

3.3 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.

3.3.1 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 digunakan untuk pembangunan *Game* ini adalah :

1. Sistem operasi *Windows 7*.
2. *Unity* versi 4.0
3. *3ds Max* 2011
4. *Mono Develop 2.8.2*

3.3.2 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* pertempuran lima hari di semarang 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 512 MB.

5. Monitor dengan resolusi 1024 x 768 pixels.
6. *Speaker*.
7. *Keyboard* dan *mouse.*

Spesifikasi minimum perangkat keras bagi pengguna untuk dapat memainkan *game* pertempuran lima hari di semarang dengan baik 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.*

3.3.3 Analisis Pengguna

Pada analisis pengguna ini akan mencakup analisis beberapa parameter terhadap calon pemain dari applikasi.

Game yang akan dibangun ini dapat dimainkan oleh berbagai kalangan akan tetapi karena game ini mengangkat salah satu cerita tentang perjuangan rakyat semarang, pertimbangan tersebut membuat *game* ini ditujukan secara khusus untuk siswa Sekolah Menengah Atas kelas XI, karena pada kelas XI materi tentang sejarah pertempuran lima hari di Semarang.

Ada 2 hal yang akan memudahkan pemain ketika memainkan *game* ini yaitu pengetahuan dan pengalaman, terutama pengetahuan dan pengalaman dalam memainkan *game first person shooter*.

3.3.3.1 *User Knowledge and Experience*

Berikut ini klasifikasi *knowledge adn experience* dari pengguna aplikasi dapat dilihat pada tabel 3.10.

Tabel 3.10 Analisis Klasifikasi *Knowledge and Experience*

Education Level	Bisa digunakan oleh berbagai kalangan, seperti pelajar, mahasiswa hingga masyarakat awam
Reading Level	Bisa digunakan oleh berbagai *level* pendidikan dengan reading *level* yang sedang
Typing Skills	*Average* (40 wpm) Tidak memerlukan *typing skills* yang tinggi
Computer Literacy	*Moderate* (menengah)
Task Experience	Bisa digunakan oleh dengan pengalaman penggunaan computer dan *game* yang sedang
System Experience	Bisa digunakan oleh dengan pengalaman penggunaan komputer dan *game* yang sedang
Application Experience	Bisa digunakan dalam semua sistem operasi *Windows xp / vista/ 7*
Native Language	Menggunakan satu bahasa, yakni Indonesia
Use Of Other System	Bisa di jalankan tanpa perlu *install* applikasi lain

3.3.3.2 *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*, dapat dilihat pada tabel 3.11.

Tabel 3.11 Analisis *Users Physical Characteristic*

Age	15 tahun ke atas
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.

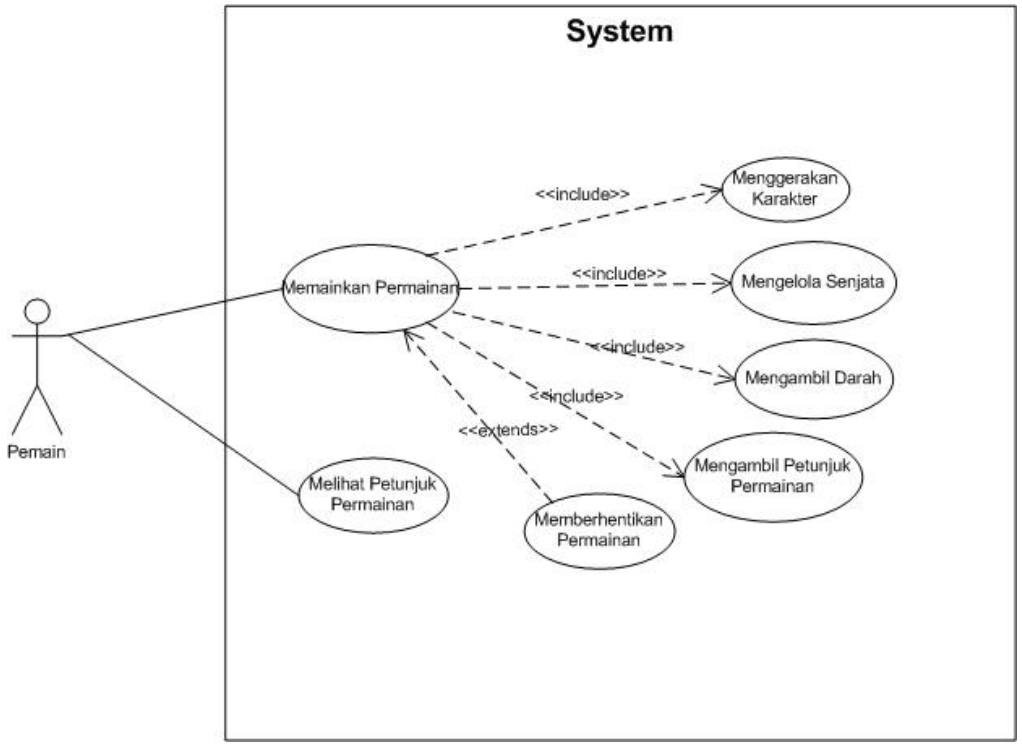
3.4 Analisis Kebutuhan Fungsional

Tahap pertama yang dilakukan dalam melakukan analisis berorientasi objek menggunakan UML adalah menentukan aktor atau pengguna sistem. Kata aktor dalam konteks UML, menampilkan peran (*roles*) pengguna atau sesuatu diluar sistem yang dikembangkan yang dapat berupa perangkat keras, end user, sistem yang lain , dan sebagainya.

3.4.1 *Use Case Diagram*

Use Case atau diagram use case merupakan pemodelan untuk kelakuan (behavior) sistem informasi yang akan dibuat. Use case mendeskripsikan sebuah interaksi antara satu atau lebih aktor dengan sistem informasi yang akan dibuat .

Pada gambar 3.16 merupakan perancangan use case yang terdapat pada *game first person shooter* sejarah pertempuran lima hari di Semarang.



Gambar 3.16 ***Use case Diagram***

3.4.1.1 Definisi *Actor*

Definisi *Actor* berfungsi untuk menjelaskan *actor* yang terdapat pada *use case diagram*. Definisi *actor* diterangkan pada tabel 3.12 berikut.

Tabel 3.12 ***Definisi Actor***

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

3.4.1.2 Definisi *Use Case*

Definisi *use case* berfungsi untuk menjelaskan proses yang terdapat pada setiap *use case* dapat dilihat pada tabel 3.13.

Tabel 3.13 Definisi ***Use Case***

No	***Use Case***	**Deskripsi**
1	Memainkan Permainan	Untuk memainkan misi-misi yang ada di dalam permainan
2	Melihat Petunjuk Permainan	Untuk melihat cara bermain atau informasi kontrol
3	Menggerakan Karakter	Proses untuk menggerakan karakter dengan interaksi *keyboard*

No	*Use Case*	Deskripsi
4	Mengelola Senjata	Proses untuk menukar senjata dan mengganti senjata apabila peluru habis
5	Mengambil Darah	Proses untuk menambah darah / *health* ketika darah tersisa sedikit lagi
6	Mengambil Petunjuk Permainan	Proses untuk pemain menyelesaikan misi permainan
7	Memberhentikan Permainan	Proses untuk memberhentikan permainan untuk sementara.

3.4.1.3 *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*.

3.4.1.3.1 *Scenario Use Case* Memainkan Permainan

Skenario *Use Case* memainkan permainan berfungsi untuk memainkan misi-misi yang ada di dalam permainan, skenario *Use Case* memainkan permainan diterangkan pada tabel 3.14.

Tabel 3.14 Ske*nario Use Case* Memainkan Permainan

Identifikasi		
Nomor	1	
Nama	Memainkan Permainan	
Tujuan	Untuk melihat scene atau cerita permainan dan masuk ke dalam permainan	
Deskripsi		
Tipe	*Primary Actor*	
Aktor	Pemain	
Skenario Utama		
Kondisi awal	Pemain berada di menu utama	
Aksi Aktor		**Reaksi Sistem**
1. Memilih menu mulai permainan		

	2. Menampilkan scene video level 1
	3. Menampilkan informasi misi
4. Menekan tombol mainkan sekarang	
	5. Sistem menampilkan permainan level 1
6. Pemain gagal menyelesaikan misi dengan membunuh 25 musuh	
	7. Sistem menampilkan permainan level 1
8. Pemain berhasil menyelesaikan misi dengan membunuh 25 musuh	
	9. Sistem menampilkan *scene* permainan selesai
10. Menekan spasi pada *keyboard*	
	11. Sistem menampilkan *scene* level 2
	12. Sistem menampilkan informasi misi
13. Menekan tombol mainkan sekarang	
	14. Sistem menampilkan permainan level 2
15. Pemain gagal menyelesaikan misi dengan membunuh 30 musuh	
	16. Sistem menampilkan permainan level 2
17. Pemain berhasil menyelesaikan misi dengan membunuh 30 musuh	
	18. Memperbolehkan mengambil bendera untuk mengakhiri permainan
	19. Menampilkan *scene* permainan selesai
Kondisi Akhir	Jika pemain gagal menyelesaikan misi sistem menampilkan permainan sebelumnya dan jika pemain berhasil menyelesaikan misi, sistem menampilkan *scene* permainan selesai

3.4.1.3.2 *Scenario Use Case* Melihat Petunjuk Permainan

Skenario*Use Case* melihat petunjuk permainan berfungsi untuk menampilkan cara bermain dapat dilihat pada tabel 3.15.

Tabel 3.15 Ske*nario Use Case* Melihat Petunjuk Permainan

Identifikasi	
Nomor	2
Nama	Melihat Petunjuk Permainan
Tujuan	Melihat petunjuk dan cara bermain atau informasi kontrol

Deskripsi		
Tipe	*Primary Actor*	
Aktor	Pemain	
Skenario Utama		
Kondisi awal	Pemain berada di *menu* utama	
Aksi Aktor		**Reaksi Sistem**
1. Pemain melihat menu cara bermain		
		2. Sistem menampilkan halaman petunjuk bermain
Kondisi Akhir	Menampilkan halaman petunjuk bermain atau informasi kontrol	

3.4.1.3.3 *Scenario Use Case* Menggerakan Karakter

Skenario*Use Case* pergerakkan karakter berfungsi untuk menampilkan interaksi antara *actor* dengan *use case* pergerakkan karakter, tabel menggerakan karakter dapat dilihat pada tabel 3.16.

Tabel 3.16 Skenario *Use Case* Menggerakan Karakter

Identifikasi		
Nomor	3	
Nama	Menggerakan Karakter	
Tujuan	Untuk menggerakan karakter yang dimainkan	
Deskripsi		
Tipe	*Primary Actor*	
Aktor	Pemain	
Skenario Utama		
Kondisi awal	Karakter diam	
Aksi Aktor		**Reaksi Sistem**
1. Pengguna menggerakkan karakter dengan interaksi menggunakan *keyboard* menekan tombol W		
		2. Sistem menggerakkan karakter menjalankan animasi berjalan maju ke depan
3. Pengguna menggerakkan karakter dengan interaksi menggunakan *keyboard* menekan tombol S		
		4. Sistem menggerakkan karakter menjalankan

	animasi berjalan lurus mundur ke belakang
5. Pengguna menggerakkan karakter dengan interaksi menggunakan *keyboard* menekan tombol D	
	6. Sistem menggerakkan karakter menjalankan animasi berjalan ke kanan
7. Pengguna menggerakkan karakter dengan interaksi menggunakan *keyboard* menekan tombol A	
	8. Sistem menggerakkan karakter menjalankan animasi berjalan ke kiri
9. Pengguna menggerakkan karakter dengan interaksi menggunakan *keyboard* menekan tombol Shift dan W secara bersamaan	
	10. Sistem menggerakkan karakter menjalankan animasi berlari ke depan
11. Pengguna menggerakkan karakter dengan interaksi menggunakan *keyboard* menekan tombol Shift dan S secara bersamaan	
	12. Sistem menggerakkan karakter menjalankan animasi berlari mundur ke belakang
13. Pengguna menggerakkan karakter dengan interaksi menggunakan *keyboard* menekan tombol Shift dan D secara bersamaan	
	14. Sistem menggerakkan karakter menjalankan animasi berlari ke kanan
15. Pengguna menggerakkan karakter dengan interaksi menggunakan *keyboard* menekan tombol Shift dan A secara bersamaan	
	16. Sistem menggerakkan karakter menjalankan animasi berlari ke kiri
Kondisi Akhir	Karakter bergerak sesuai dengan inputan pemain

3.4.1.3.4 *Scenario Use Case* Mengelola Senjata

Skenario*Use Case* mengelola sanjata berfungsi untuk mengganti atau membuang senjata yang digunakan oleh pemain, tabel mengelola senjata dapat dilihat pada tabel 3.17.

Tabel 3.17 Skenario *Use Case* Mengolah Senjata

Identifikasi		
Nomor	4	
Nama	Mengelola senjata	
Tujuan	Untuk mengganti dan mengambil senjata	
Deskripsi		
Tipe	*Primary Actor*	
Aktor	Pemain	
Skenario Utama		
Kondisi awal	Pemain memainkan permainan	
Aksi Aktor		**Reaksi Sistem**
1. Menekan tombol G pada *keyboard*		
		2. Karakter pemain membuang senjata
3. Pemain menggerakkan karakter dan menjalankan menuju *item* senjata.		
		4. Karakter menggunakan senjata
5. Menekan tombol 1 pada *keyboard*		
		6. Sistem menampilkan karakter dengan senjata pistol
7. Menekan tombol 2 pada *keyboard*		
		8. Sistem menampilkan karakter dengan senjata machinegun
Kondisi Akhir	Sistem menampilkan karakter menjalankan animasi membuang senjata dan mengganti senjata pistol dan senjata machinegun sesuai inputan pemain.	

3.4.1.3.5 *Scenario Use Case* Mengambil Darah

Skenario*Use Case* mengambil darah berfungsi untuk menambah darah karakter apabila tinggal sedikit lagi, tabel mengambil darah dapat dilihat pada tabel 3.18.

Tabel 3.18 Skenario *Use Case* Mengambil Darah

Identifikasi		
Nomor	5	
Nama	Mengambil Darah	
Tujuan	Untuk menambahkan darah pemain	
Deskripsi		
Tipe	*Primary Actor*	
Aktor	Pemain	
Skenario Utama		
Kondisi awal	Darah pemain berkurang	
Aksi Aktor		**Reaksi Sistem**
1. Pemain menggerakkan karakter dan menjalankan menuju *item health* / darah.		
		2. Sistem menambah darah untuk karakter
		3. Sistem melakukan pengecekan apabila darah pemain penuh, sistem tidak mengalami perubahan apapun
		4. Apabila darah pemain tidak penuh, sistem menampilkan *bar* darah pemain yang bertambah dan menghilangkan *icon* darah yang telah diambil oleh pemain.
Kondisi Akhir	Sistem menampilkan darah karakter bertambah dan *icon* darah menghilang. Apabila darah karakter dalam keadaan penuh darah tidak bertambah dan icon darah tidak menghilang.	

3.4.1.3.6 *Scenario Use Case* Mengambil Petunjuk Permainan

Skenario *Use Case* mengambil petunjuk misi berfungsi untuk menampilkan misi yang ada di dalam *game*, tabel menampilkan petunjuk informasi dapat dilihat pada tabel 3.19.

Tabel 3.19 Skenario *Use Case* Mengambil Petunjuk Permainan

Identifikasi	
Nomor	6
Nama	Mengambil petunjuk permainan
Tujuan	Untuk menampilkan misi yang ada pada *game*
Deskripsi	
Tipe	*Primary Actor*
Aktor	Pemain
Skenario Utama	
Kondisi awal	Pemain mengambil petunnjuk
Aksi Aktor	**Reaksi Sistem**
1. Pemain mengambil petunjuk	
	2. Sistem menampilkan informasi / petunjuk permainan
Kondisi Akhir	Sistem menampilkan informasi / petunjuk permainan

3.4.1.3.7 *Scenario Use Case* Memberhentikan Permainan

Skenario *Use Case* memberhentikan permainan berfungsi untuk menampilkan menu jeda yang ada di dalam *game*, tabel memberhentikan permainan dapat dilihat pada tabel 3.20.

Tabel 3.20 Skenario *Use Case* Memberhentikan permainan

Identifikasi	
Nomor	7
Nama	Memberhentikan permainan
Tujuan	Memberhentikan permainan untuk sementara.
Deskripsi	
Tipe	*Primary Actor*
Aktor	Pemain
Skenario Utama	
Kondisi awal	Pemain memainkan *game*
Aksi Aktor	**Reaksi Sistem**
1. Pengguna menekan tombol ESC pada *keyboard*.	
	2. Sistem menampilkan menu jeda
3. Pemain memilih *menu* kembali ke	

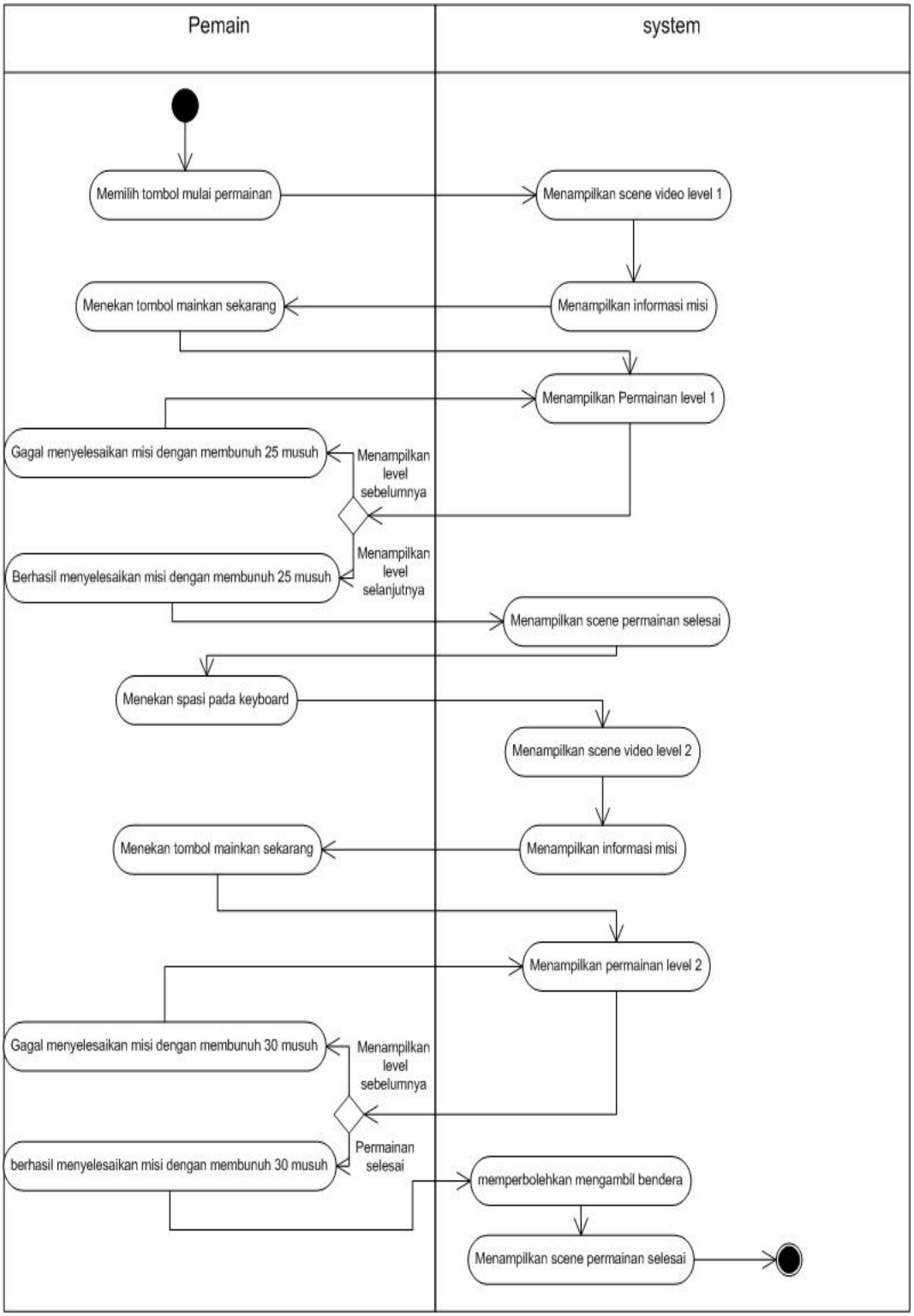
permainan.	
	4. Sistem mengembalikan permainan ke tampilan pengguna sebelumnya.
5. Pemain memilih menu utama	
	6. Sistem menampilkan *menu* utama.
Kondisi Akhir	Sistem memberhentikan permainan mengeluarkan menu jeda. Jika pengguna memilih kembali ke permainan, maka sistem mengembalikan tampilan pengguna sebelumnya. Jika pemain memilih menu utama dari permainan, maka sistem keluar dari permainan dan menampilkan *menu* utama.

3.4.2 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*)

3.4.2.1 *Activity Diagram* Memainkan Permainan

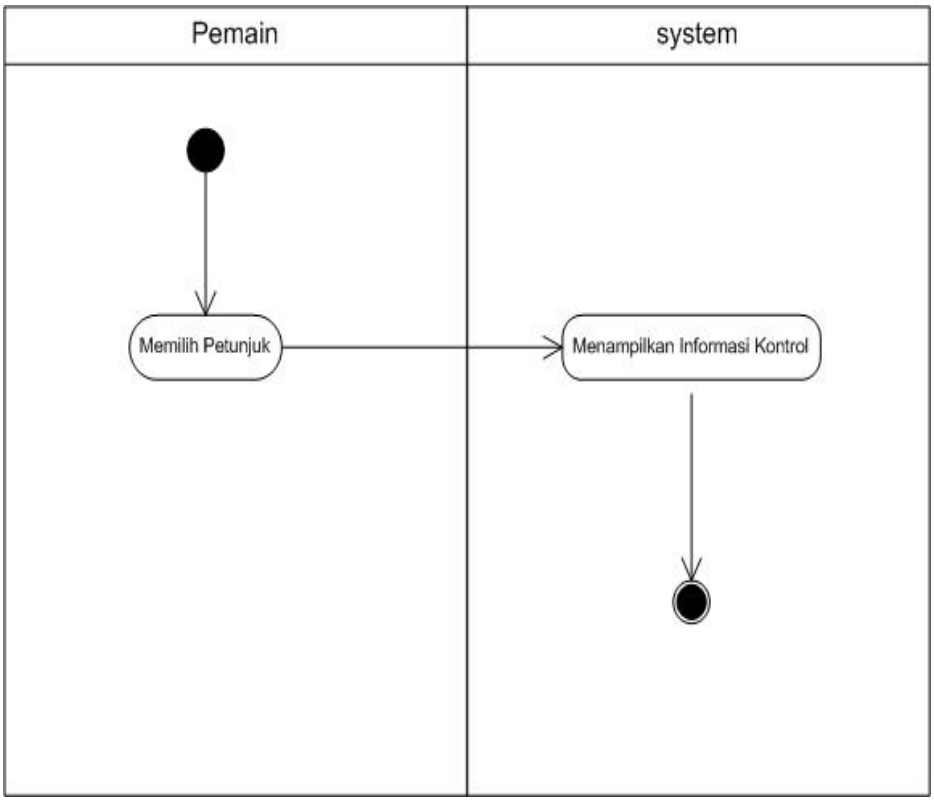
Activity diagram memainkan permainan merupakan aktivitas diagram untuk menyajikan seluruh konten permainan yang ada di dalam *game activity diagram* menampilkan penyajian permainan dapat dilihat pada gambar 3.17.



Gambar 3.17 *Activity Diagram* Memainkan Permainan

3.4.2.2 *Activity Diagram* Melihat Petunjuk Permainan

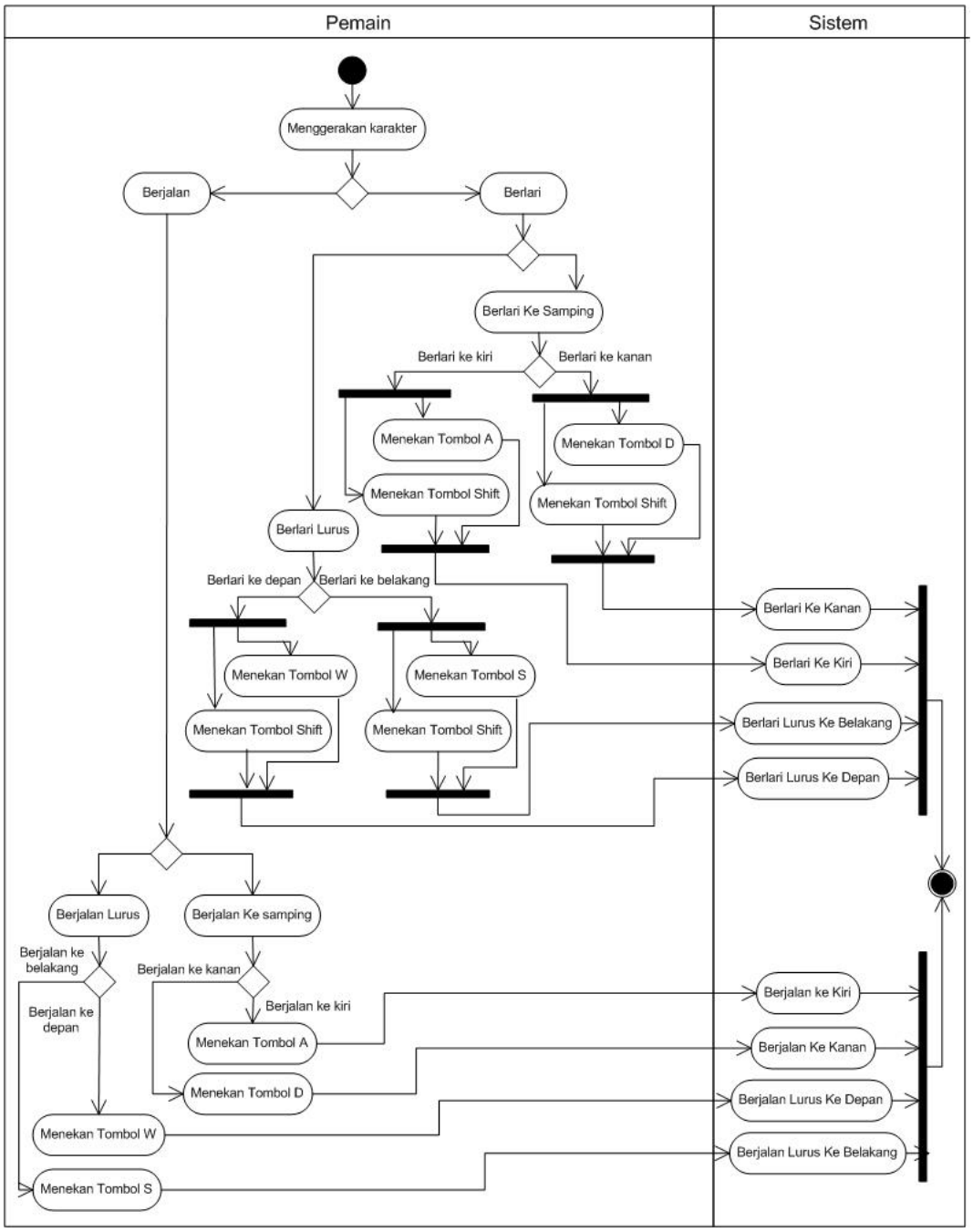
Activity diagram melihat petunjuk permainan digunakan untuk menampilkan cara bermain yang ada di dalam *game. Activity diagram* melihat petunjuk permainan bisa dilihat pada gambar 3.18.



Gambar 3.18 *Activity Diagram* Melihat Petunjuk Permainan

3.4.2.3 *Activity Diagram* Menggerakan Karakter

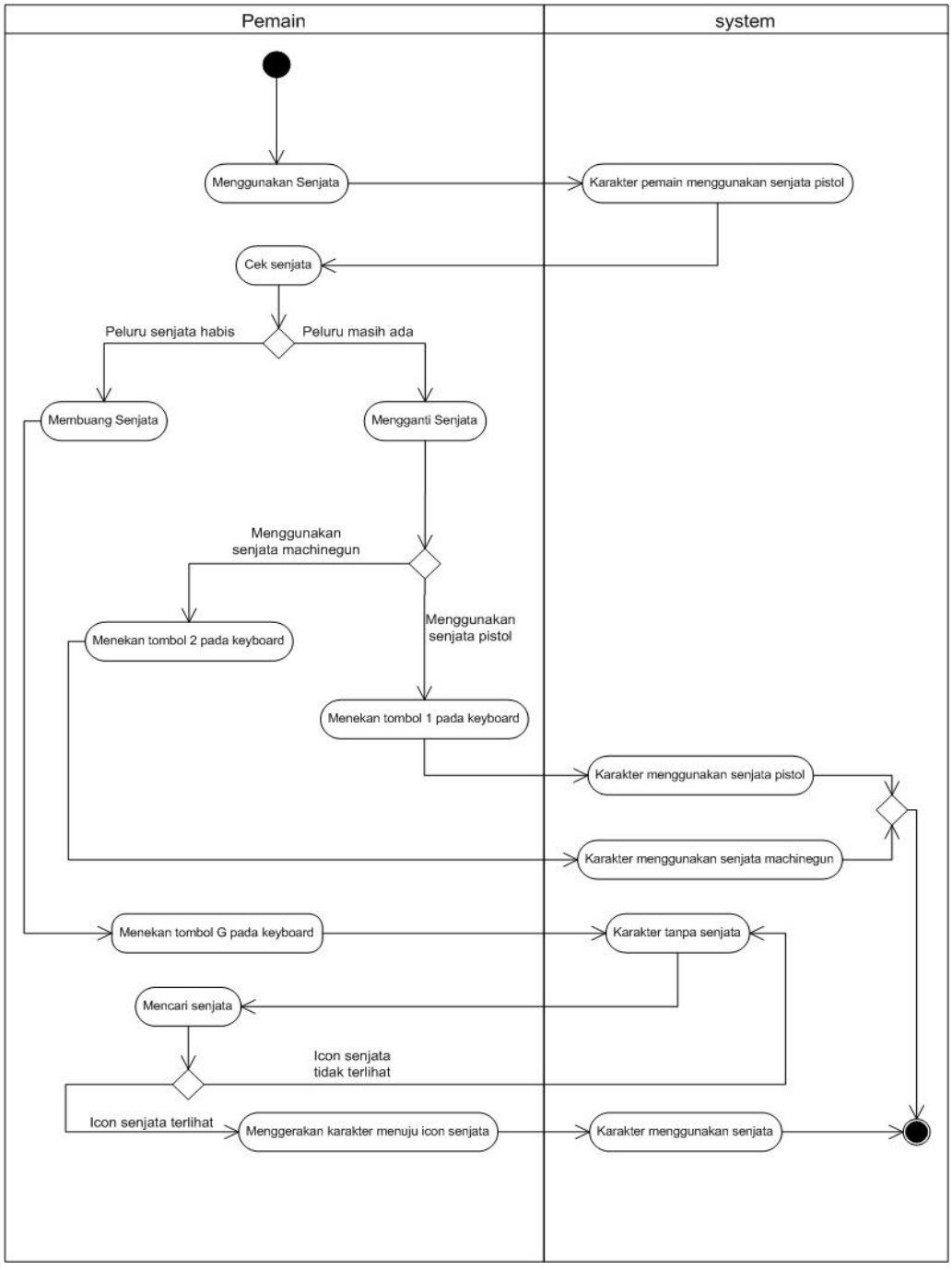
Activity diagram pergerakkan karakter digunakan untuk menampilkan menggerakkan karakter utama yang ada di dalam *game*. *Activity diagram* pergerakkan karakter bisa dilihat pada gambar 3.19.



Gambar 3.19 *Activity Diagram* Menggerakan Karakter

3.4.2.4 *Activity Diagram* Mengelola Senjata

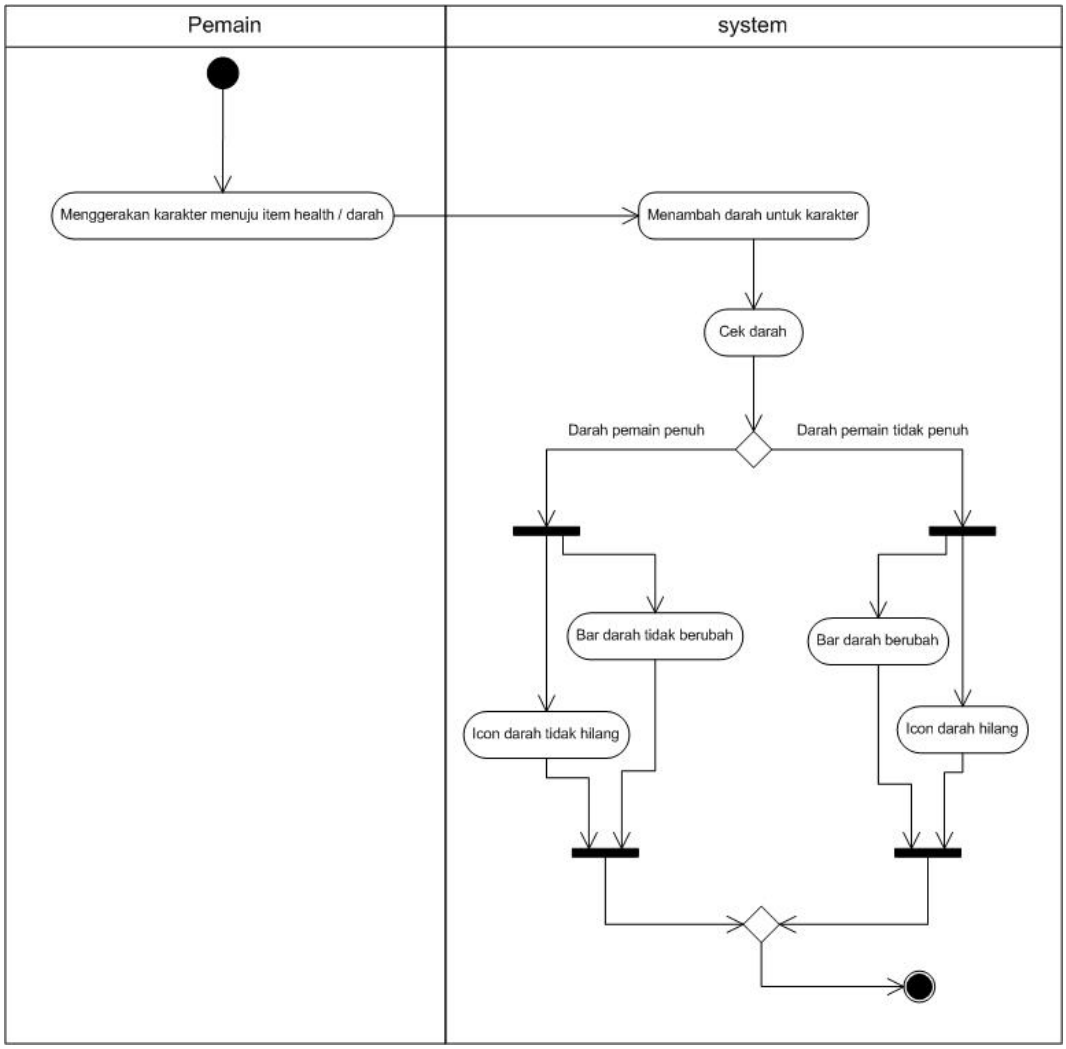
Activity diagram mengelola senjata digunakan untuk menampilkan senjata yang digunakan karakter utama yang ada di dalam *game*. *Activity diagram* mengelola senjata bisa dilihat pada gambar 3.20.



Gambar 3.20 *Activity Diagram* Mengelola Senjata

3.4.2.5 *Activity Diagram* Mengambil Darah

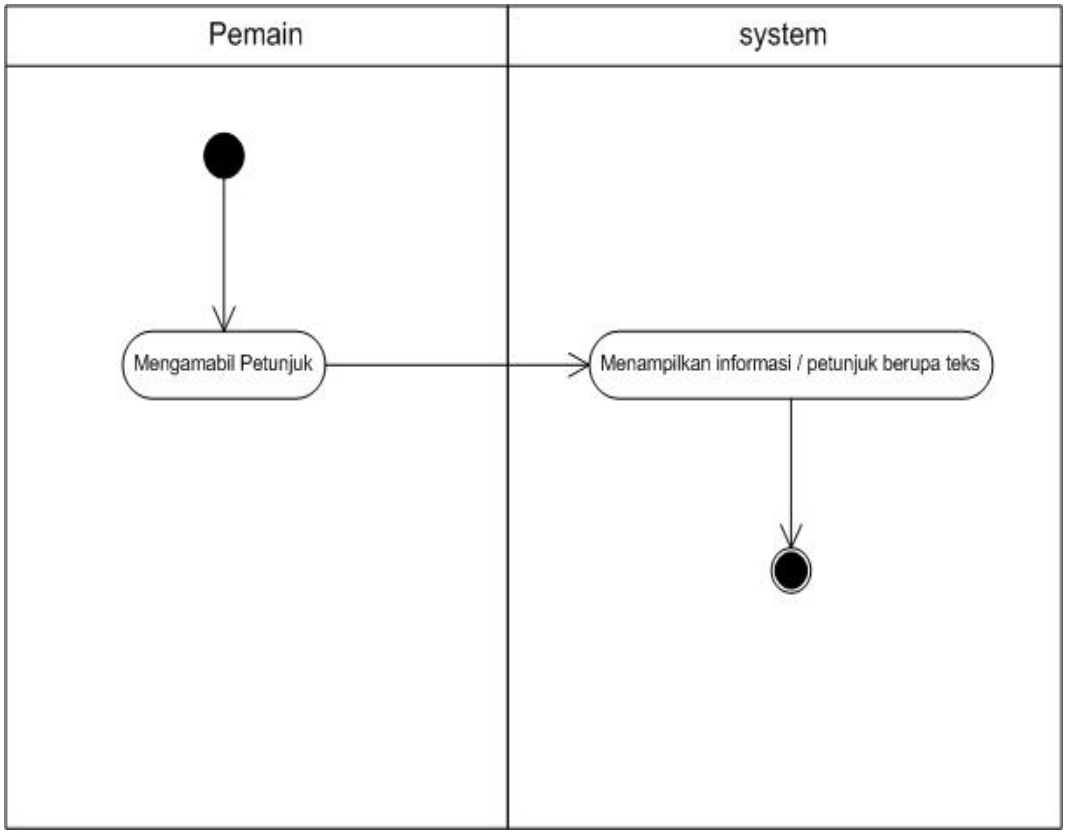
Activity diagram mengambil darah digunakan menambah bar darah karakter utama yang ada di dalam *game. Activity diagram* mengelola senjata bisa dilihat pada gambar 3.21.



Gambar 3.21 *Activity Diagram* Mengambil Darah

3.4.2.6 *Activity Diagram* Mengambil Petunjuk Permainan

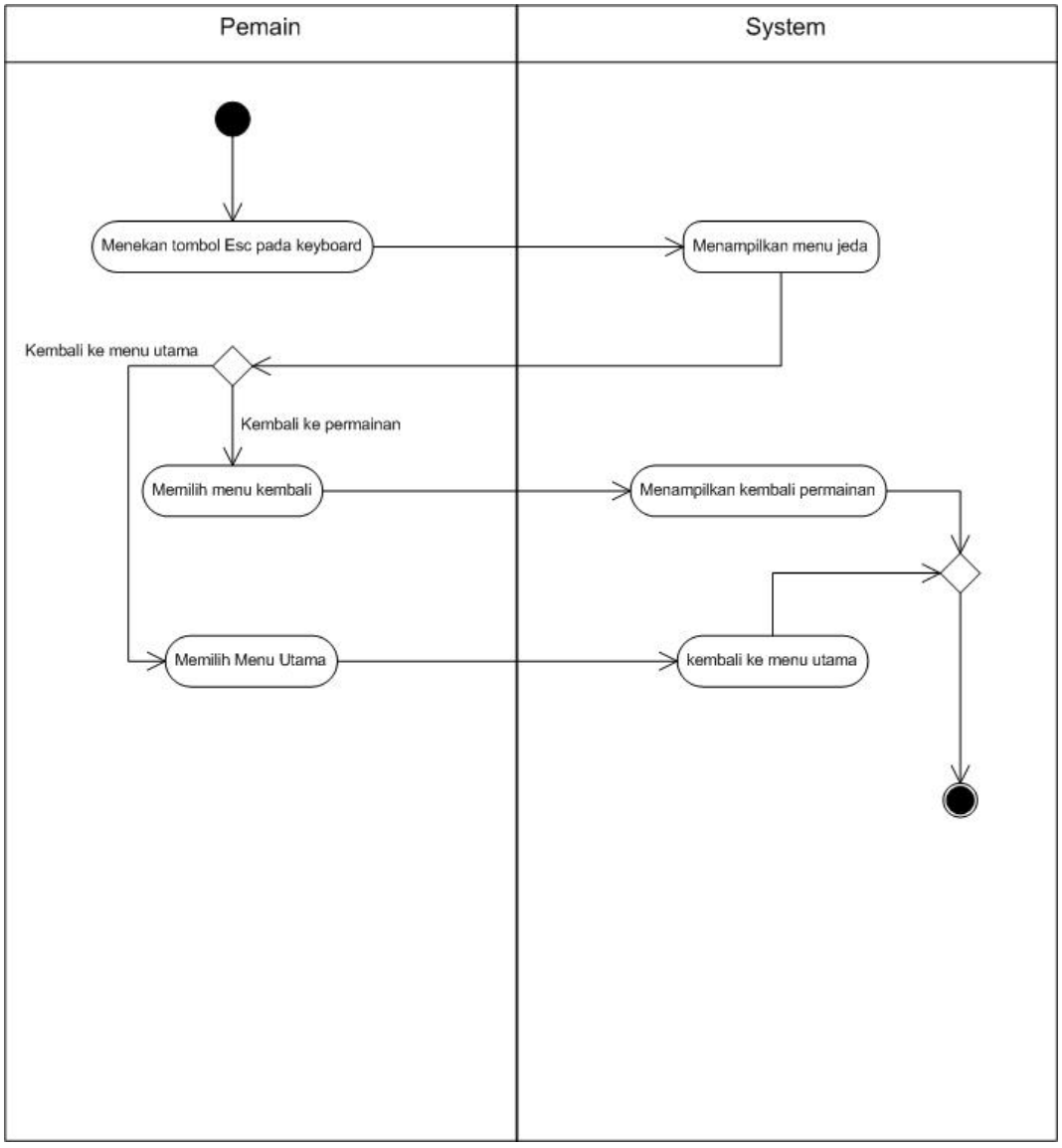
Activity diagram mengambil petunjuk permainan digunakan untuk menampilkan misi yang harus dijalankan ketika *player* berada didalam arena *game. Activity diagram* perhitungan skor bisa dilihat pada gambar 3.22.



Gambar 3.22 *Activity Diagram* Mengambil Petunjuk Permainan

3.4.2.7 *Activity Diagram* Memberhentikan Permainan

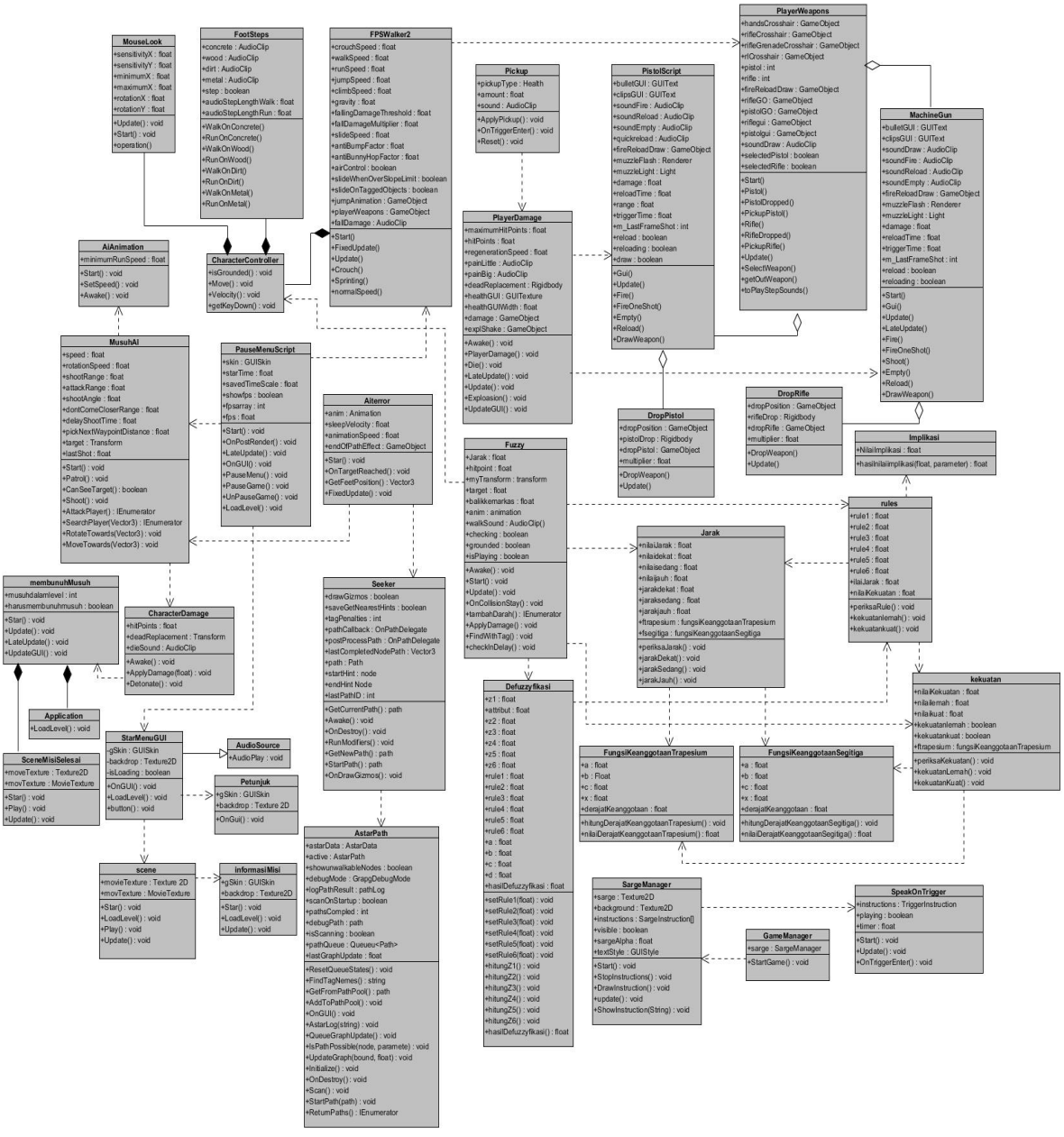
Activity diagram memberhentikan permainan digunakan untuk menampilkan menu jeda, di sini pemain dapat memilih dua menu apakah akan melanjutkat permainan atau kembali ke menu utama berada didalam arena *game. Activity diagram* perhitungan skor bisa dilihat pada gambar 3.23.



Gambar 3.23 *Activity Diagram* Memberhentikan Permainan

3.4.3 Class Diagram

Class diagram menggambarkan struktur sistem dari segi pendefinisian kelas-kelas yang dibuat untuk membangun sistem. Pada gambar 3.24 merupakan class diagram *game* Pertempuran Lima Hari di Semarang :



Gambar 3.24 *Class diagram* Pertempuran lima hari di semarang

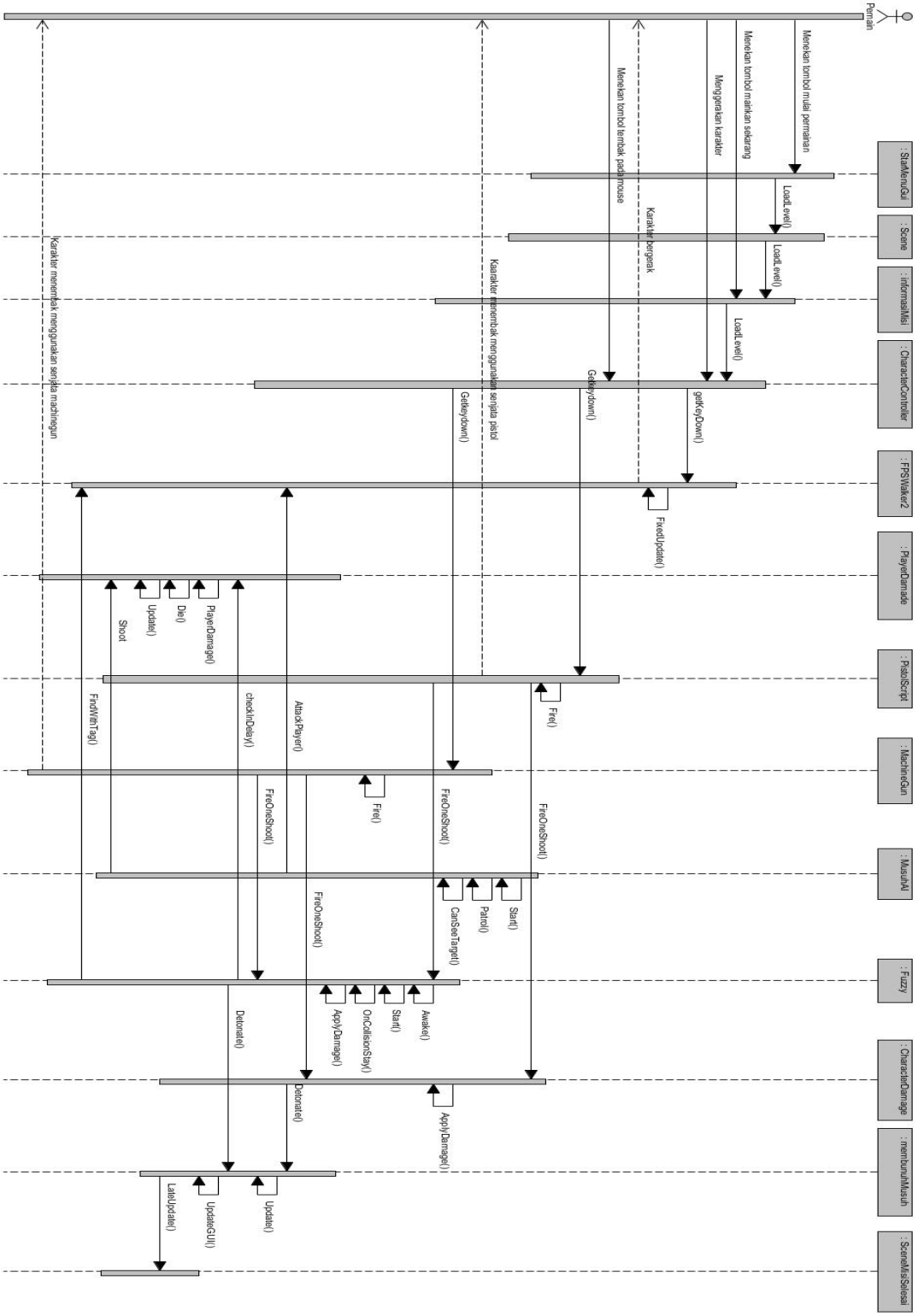
3.4.4 Sequence Diagram

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 menggambarakan 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.

3.4.4.1 Sequence Diagram Memainkan Permainan

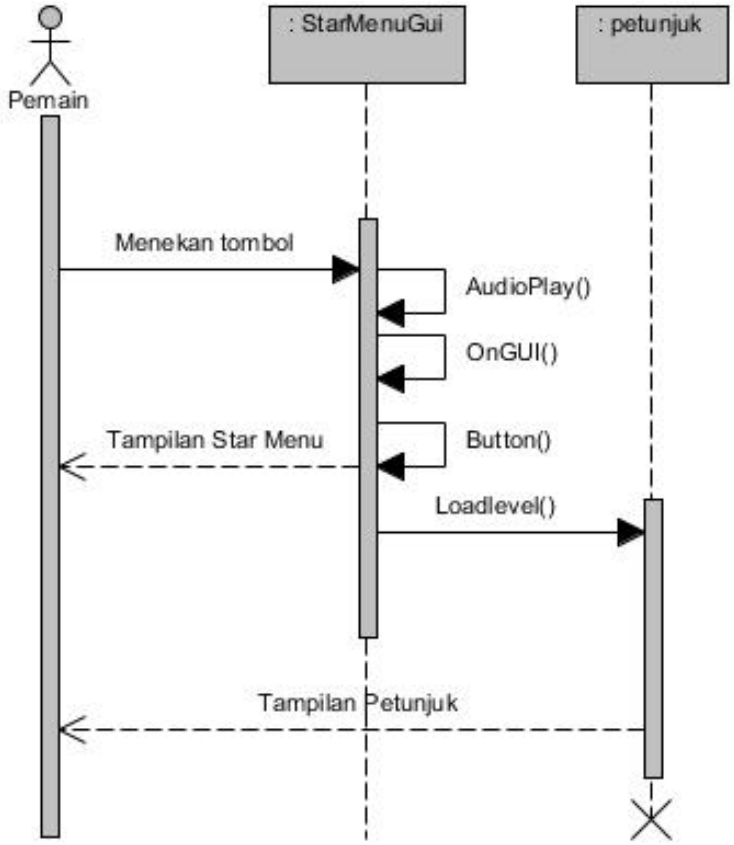
Sequence diagram untuk memainkan permainan bisa dilihat pada gambar 3.25.



Gambar 3. 25 *Sequence Diagram* Memainkan Permainan

3.4.4.2 Sequence Diagram Melihat Petunjuk Permainan

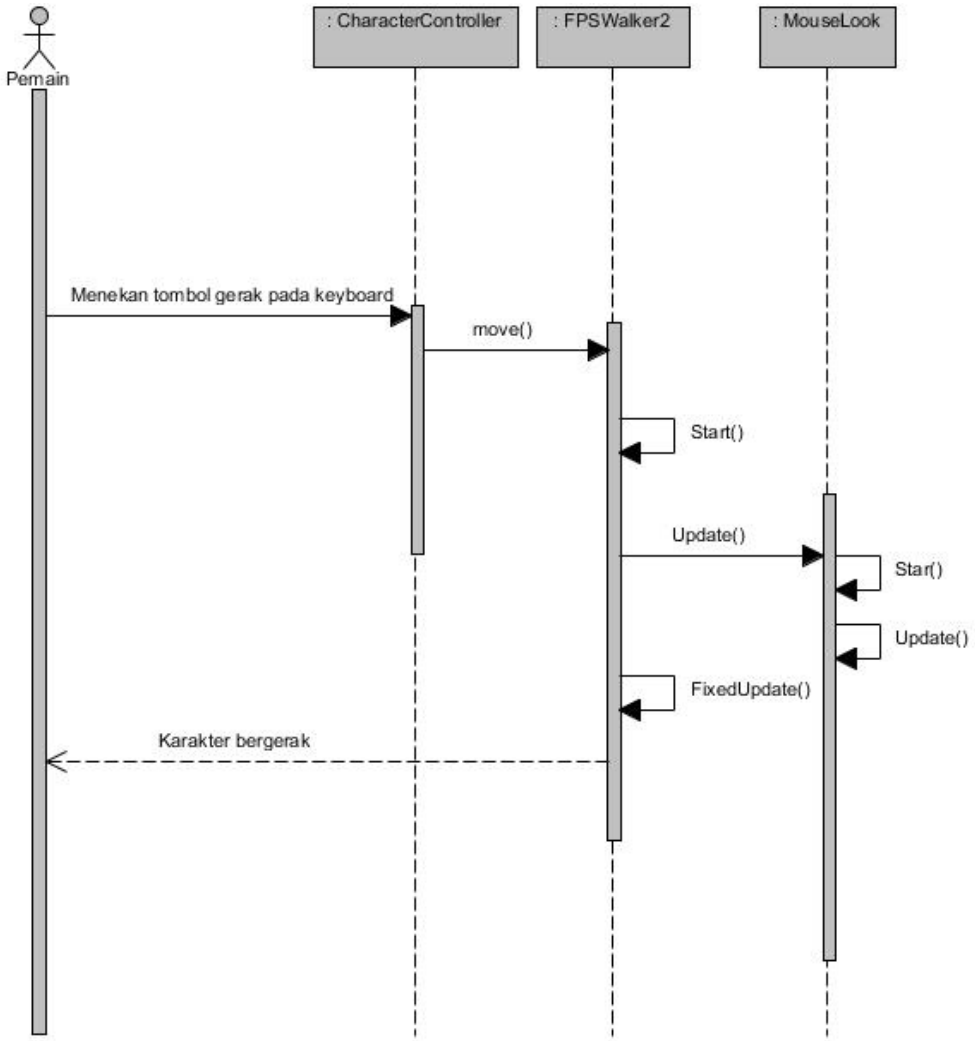
Sequence diagram untuk melakukan melihat petunjuk permainan bisa dilihat pada gambar 3.26.



Gambar 3. 26 *Sequence Diagram* Melihat Petunjuk Permainan

3.4.4.3 Sequence Diagram Menggerakan Karakter

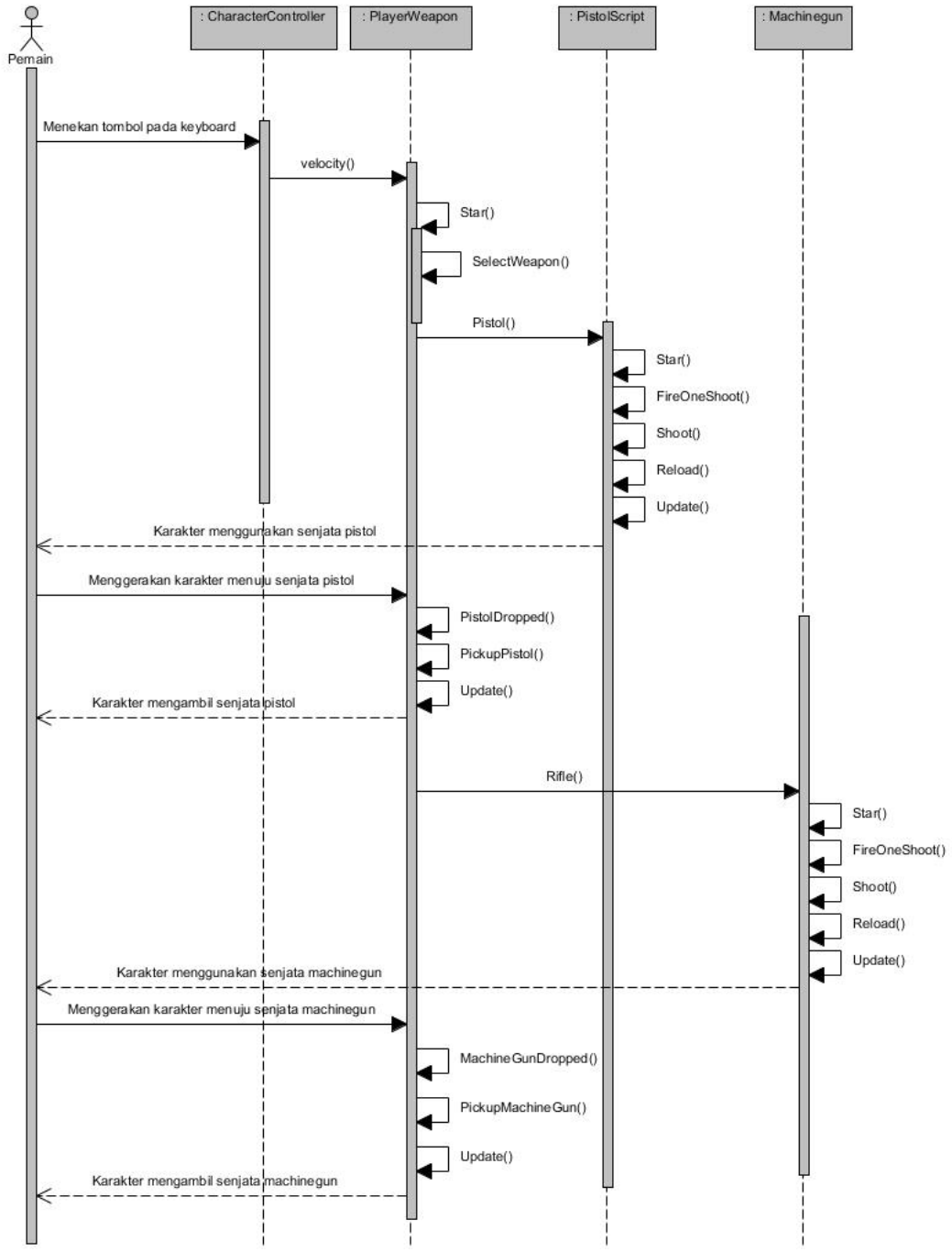
Sequence diagram untuk melakukan pergerakan karakter bisa dilihat pada gambar 3.27.



Gambar 3. 27 *Sequence Diagram* Menggerakan Karakter

3.4.4.4 Sequence Diagram Mengelola Senjata

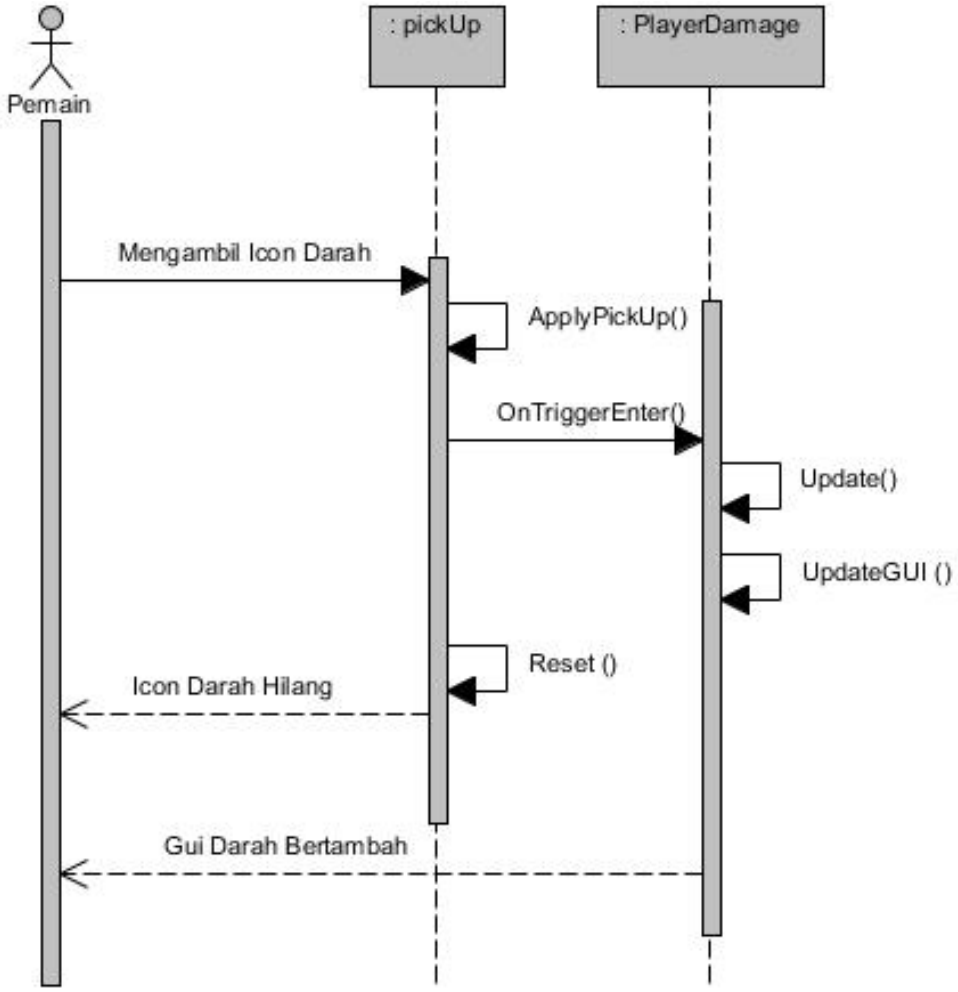
Sequence diagram untuk mengelola senjata bisa dilihat pada gambar 3.28.



Gambar 3. 28 *Sequence Diagram* Mengelola Senjata

3.4.4.5 Sequence Diagram Mengambil Darah

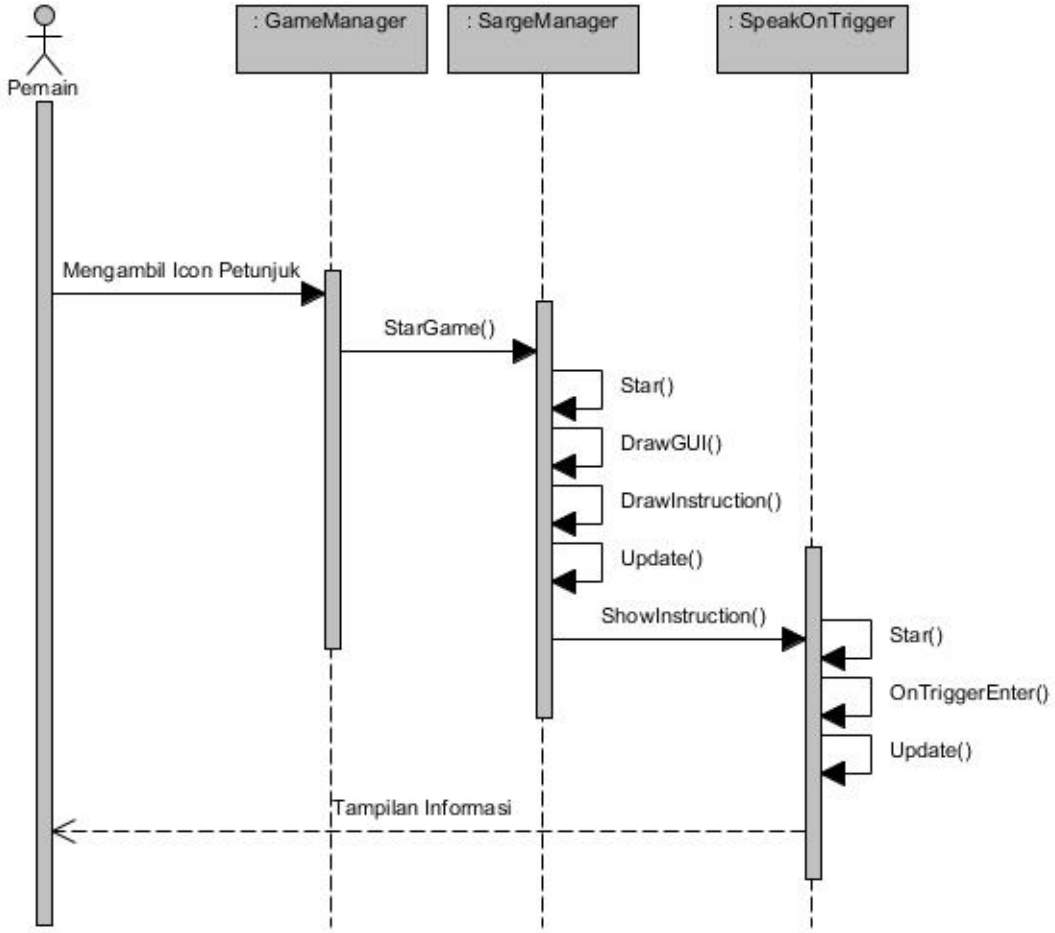
Sequence diagram untuk mengambil darah karakter bisa dilihat pada gambar 3.29.



Gambar 3. 29 *Sequence Diagram* Mengambil Darah

3.4.4.6 Sequence Diagram Mengambil Petunjuk Permainan

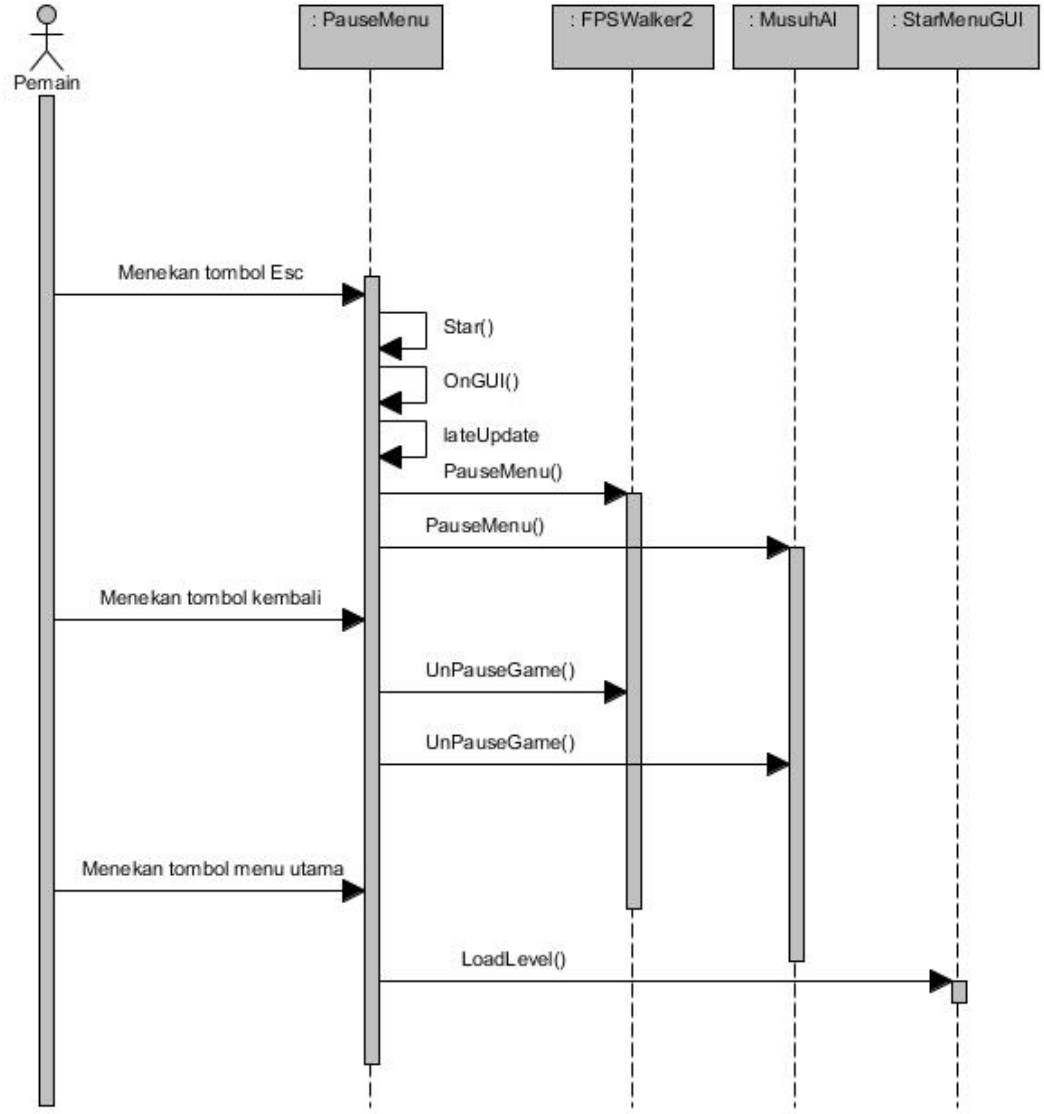
Sequence diagram untuk mengambil petunjuk informasi permainan bisa dilihat pada gambar 3.30.



Gambar 3. 30 *Sequence Diagram* Mengambil Petunjuk Permainan

3.4.4.7 Sequence Diagram Memberhentikan Permainan

Sequence diagram untuk melakukan memberhentikan permainan bisa dilihat pada gambar 3.311.



Gambar 3. 31 *Sequence Diagram* Memberhentikan Permainan

3.5 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..

3.5.1 Pengenalan Karakter

3.5.1.1. Karakter Utama

Karakter utama dalam *game* ini bernama Martadi seorang tentara atau pejuang yang akan melawan tentara Jepang. Martadi dilengkapi dua senjata, Pertama senjata pistol dan kedua senjata MachineGun. Karena *game* ini merupakan *game* yang mempunyai *genre first person shooter* yang mempunyai sudut pandang orang pertama, maka karakter Sudanco Mirza Midharta seperti tubuh, kepala dan kaki tidak diperlihatkan tetapi hanya memperlihatkan bagian tangan dan senjata dijelaskan pada tabel 3.21.

Tabel 3.21 Properties Karakter Utama

Nama	Sudanco Mirza Midharta
Darah	200
Kemampuan	Berlari, Berjalan, melompat, menembak
Senjata	Pistol dan MachineGun

3.5.1.2. Senjata

Senjata yang digunakan oleh karakter utama yang bernama Sudanco Mirza Midharta ada dua macam yaitu senjata pistol dan machinegun yang masing-masing memiliki peluru terbatas. Dapat dilihat pada gambar 3.32 dan 3.33.



Gambar 3.32 Senjata Pistol



Gambar 3.33 Senjata MachineGun

Perbandingan *properties* dari dua senjata yang di pakai oleh pemain dijelaskan pada tabel 3.22.

Tabel 3.22 Perbandingan Properties Senjata

Properti	Pistol	MachineGun
Damage	10	30
Jarak	50	100
Waktu Reload	0.5 second	1 second
Peluru	20 x 10	20 x 5

3.5.1.3. Musuh

Karakter musuh yang terdapat pada *game* ini adalah tentara jepang. Terdapat dua jenis tentara jepang yang ada di dalam *game*, yang pertama adalah tentara jepang yang menggunakan baju berwarna coklat dengan AI atau Kecerdacan buatan yaitu berlari dan menyerang. Senjata yang di pakai oleh tentara jepang yang menggunakan baju berwarna coklat adalah pistol dan machinegun pada dilahat pada gambar 3.34 dan gambar 3.35 serta propertiesnya dapat dilihat pada tabel 3.23 dan tabel 3.24. Sedangkan, yang kedua tentara jepang

yang menggunakan baju berwarna abu-abu dengan AI atau kecerdasan buatan menggunakan *fuzzy logic* dengan kemampuasn bisa menyerang, bertahan dan melarikan diri. Senjata yang di pakai oleh tentara jepang yang menggunakan baju coklat yaitu machinegun dapat dilihat pada gambar 3.36 serta propertiesnya dapat dilihat pada tabel 3.25 .

Berikut ini adalah penjelasan perbedaan dari kedua musuh :



Gambar 3.34 Karakter Musuh Tentara Jepang Senjata Pistol

Tabel 3.23 Properties Musuh Tentara Jepang Senjata Pistol

Tentara Jepang	
Darah	100
Kemampuan	Berlari dan Menyerang
Senjata	Pistol
Damage	10



Gambar 3.35 Karakter Musuh Tentara Jepang Senjata MachineGun

Tabel 3.24 Properties Musuh Tentara Jepang Senjata MachineGun

Tentara Jepang	
Darah	100
Kemampuan	Berlari dan Menyerang
Senjata	MachineGun
Damage	15



Gambar 3.36 Karakter Musuh Tentara Jepang Senjata Pistol

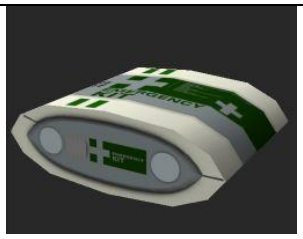
Tabel 3.25 Properties Musuh Tentara Jepang Senjata Pistol

Tentara Jepang	
Darah	100
Kemampuan	Menyerang, bertahan dan melarikan diri
Senjata	Pistol
Damage	20

3.5.1.4. Item

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

Tabel 3. 26 Tabel Penjelasan *Items*

Gambar *Items*	**Nama *Items***	**Keterangan**
	Petunjuk Informasi	*Item* yang di ambil oleh karakter utama untuk melihat petunjuk permainan. Item ini terdapat di setiap levelnya.
	Darah	*Item* yang bisa diambil oleh karakter utama untuk menambah darah atau *hitpoints*. Satu kali ambil bertambah darah 30 *hitpoints*. *Item* ini terdapat di setiap levelnya, di *level* 1 terdapat 6 buah *item* darah dan di *level* 2 terdapat 8 buah *item* darah.

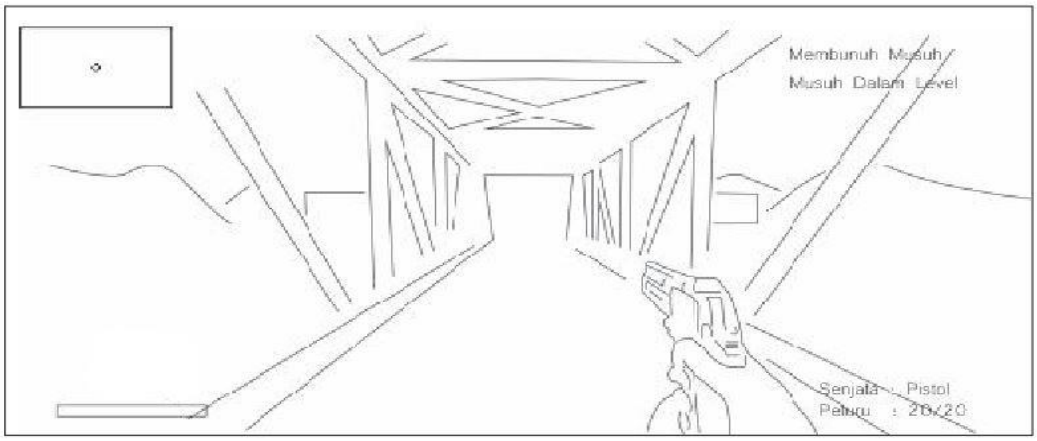
	Senjata pistol dan machinegun	*Item* yang bisa diambil oleh karakter utama apabila peluru sudah habis, terdapat dua senjata yaitu senjata pistol dengan jumlah peluru 20 dan 20 kali reload sedangkan senjata machinegun dengan jumlah peluru 20 dan 10 kali reload.

3.5.2 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.

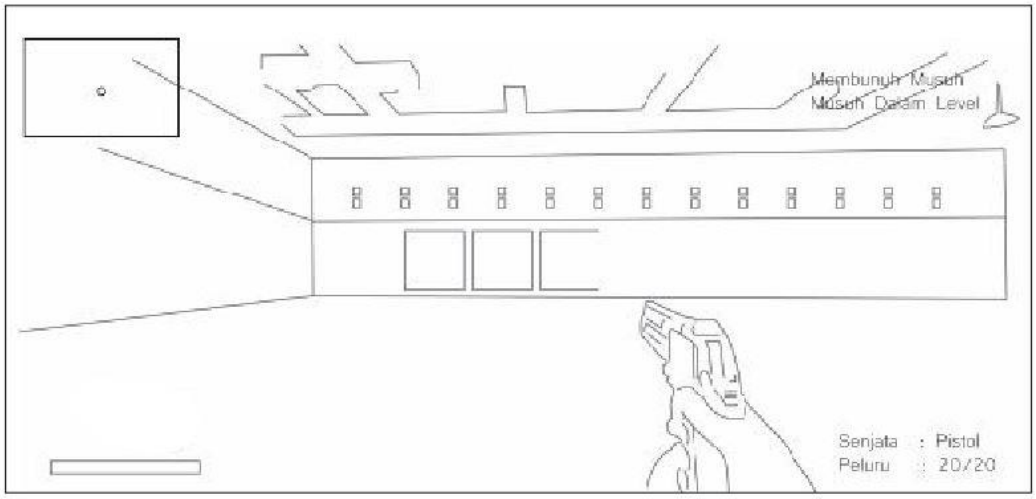
Berikut beberapa *storyboard* dari *game first person shooter* 3D sejarah pertempuran lima hari di Semarang.

1. Tampilan *Storyboard* di *Level* 1



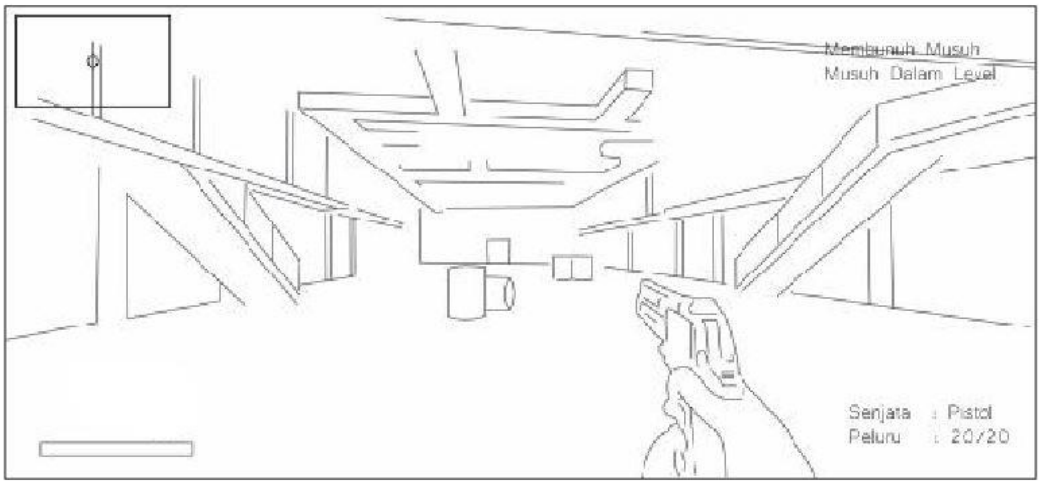
Gambar 3.37 *Storyboard level* 1-1

Pada gambar 3.37 di atas menunjukkan *level* 1 dimana cerita dimulai di sebuah jalan menuju gudang yang di kuasai oleh tentara jepang.



Gambar 3.38 *Storyboard level* 1-2

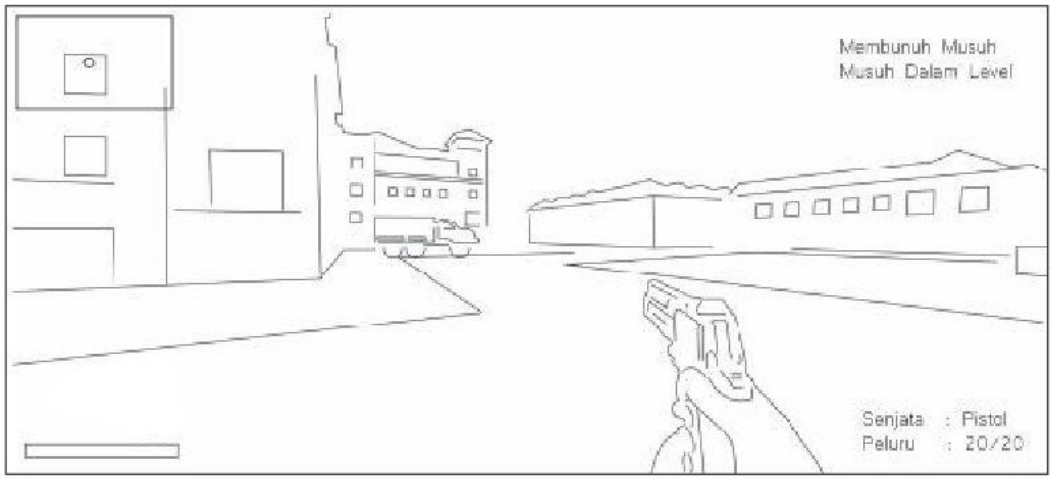
Pada gambar 3.38 Pemain berada di dalam gudang yang di penuhi oleh tentara jepang di mana pemain harus membunuh tentara jepang yang berada di dalam gudang tersebut.



Gambar 3.39 *Storyboard level* 1-3

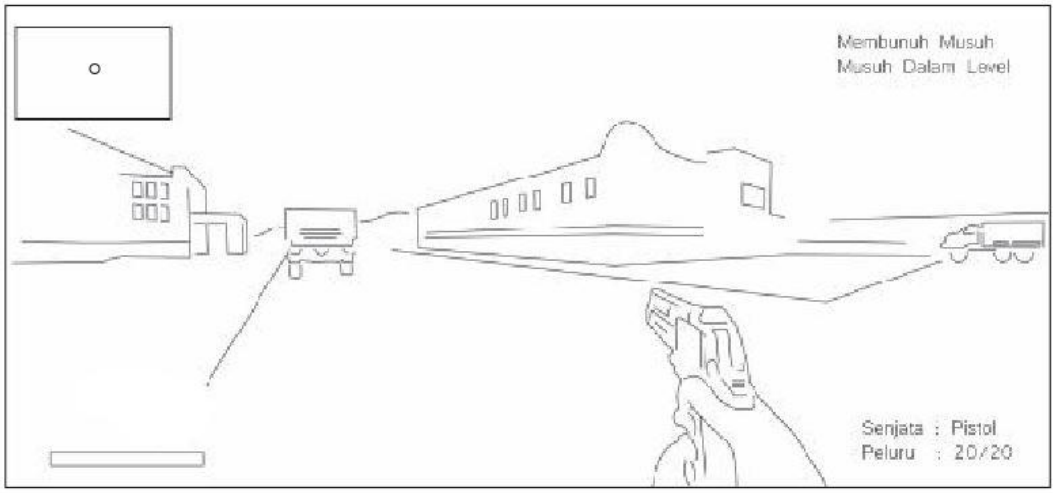
Pada gambar 3.39 Pemain berada di dalam salah satu ruangan di dalam gudang yang di penuhi oleh tentara jepang.

2. Tampilan *Storyboard* di *Level* 2



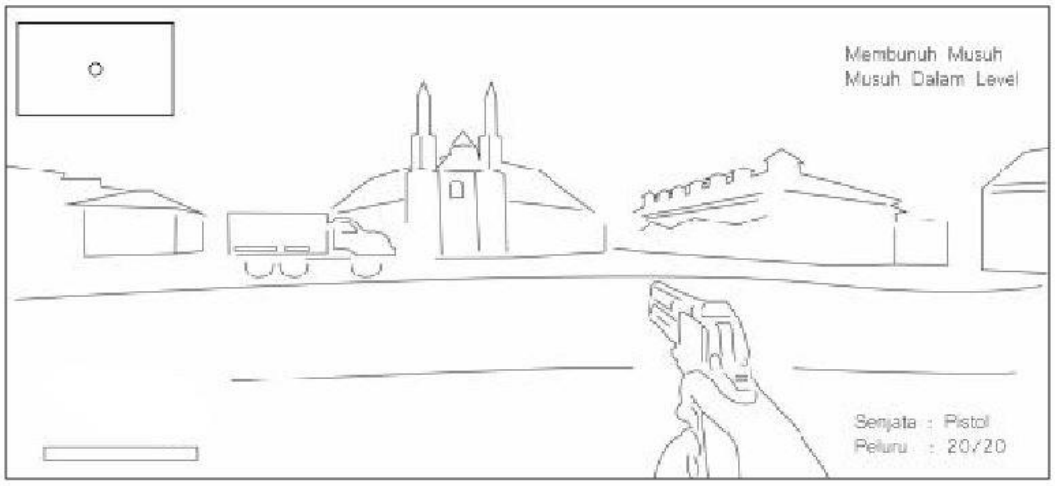
Gambar 3.40 *Storyboard level* 2-1

Pada gambar 3.40 Pemain berada di simpang lima semarang di mana pemain harus menemukan lawang sewu.



Gambar 3.41 *Storyboard level* 2-2

Pada gambar 3.41 Pemain berada di salah satu jalan menuju lawang sewu yang di penuhi oleh tentara jepang.



Gambar 3.42 *Storyboard level* 2-3

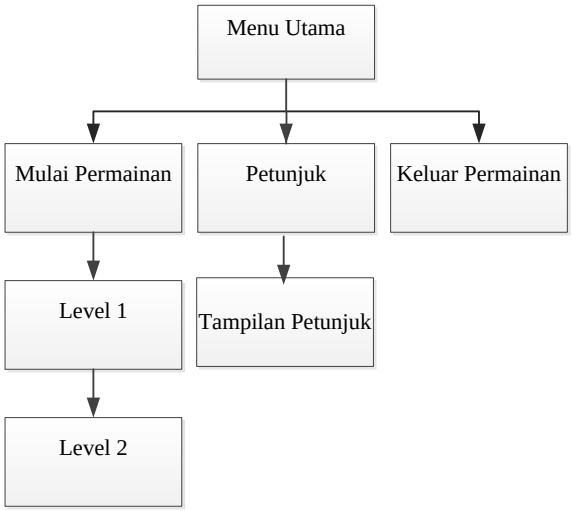
Pada gambar 3.42 Pemain berada di lawang sewu di mana pemain harus menemukan bendera merah putih untuk mengakhiri permainan.

3.6 Perancangan Arsitektur

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

3.6.1 Perancangan Struktur Menu

Perancangan Struktur Menu dibuat untuk mempermudah pembuat dalam membuat rancangan menu agar bisa dengan mudah digunakan oleh *user* `dalam memilih menu aplikasi yang sedang berjalan struktur menu dapat dilihat pada gambar 3.43.



Gambar 3.43 Struktur *Menu* Utama

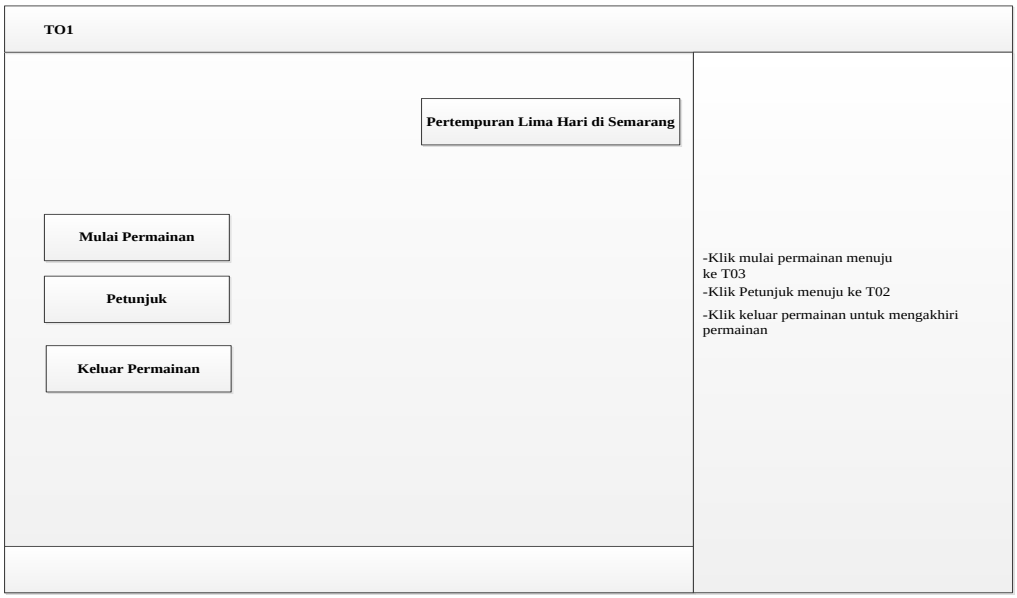
3.6.2 Perancangan Antarmuka

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

3.6.2.1 Perancangan Antarmuka Menu Utama

Berikut dapat dilihat perancangan antarmuka menu utama dari *game* pertempuran lima hari di semarang.

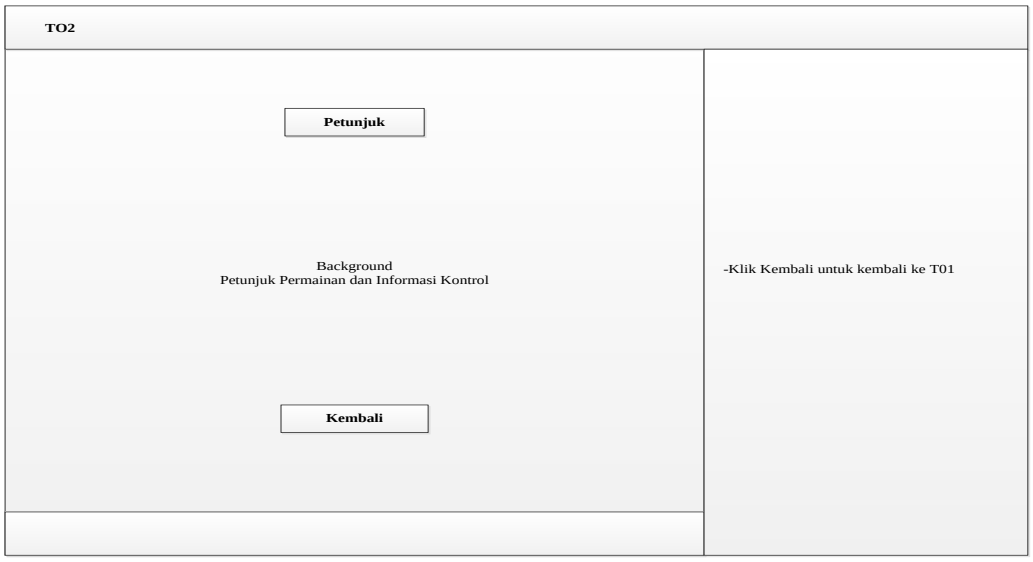
Masuk ke Menu Utama menampilkan tiga tombol pada layar yang bisa dipilih oleh *user* sebelum bermain *game*. Antarmuka Menu Utama dapat dilihat pada gambar 3.44.



Gambar 3.44 Tampilan *Menu* Utama

3.6.2.2 Perancangan Antarmuka Menu Petunjuk

Perancangan antarmuka menu petunjuk digunakan untuk melihat informasi cara bermain dapat dilihat pada gambar 3.45.



Gambar 3.45 Tampilan Antarmuka *Menu* Petunjuk

3.6.2.3 Perancangan Antarmuka Scene Video

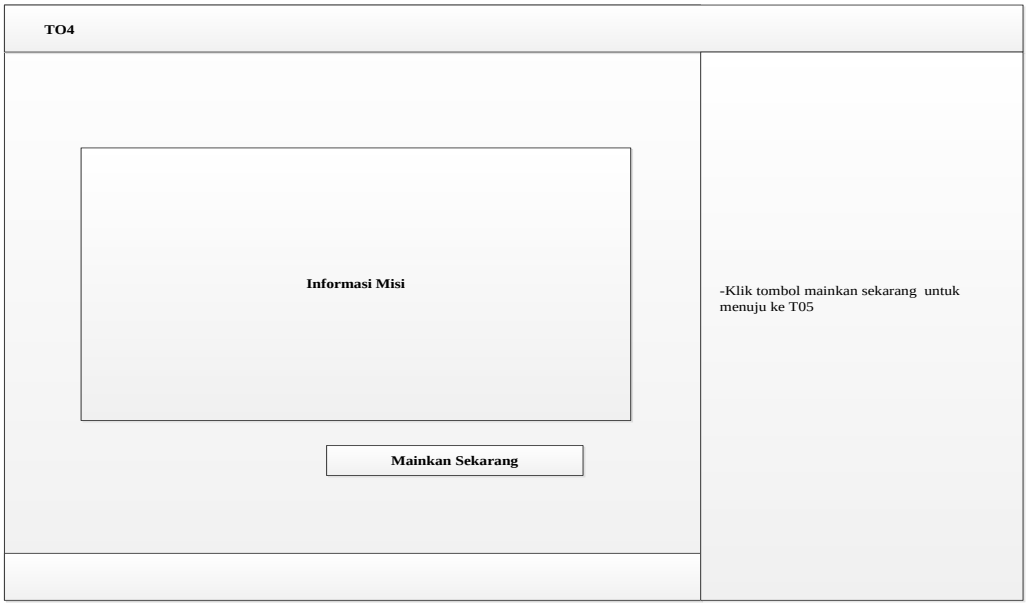
Perancangan antarmuka Scane video digunakan untuk melihat cerita atau peristiwa sejarah pertempuran lima hari di semarang dapat dilihat pada gambar 3.46.



Gambar 3.46 Tampilan Antarmuka Scene Video

3.6.2.4 Perancangan Antarmuka Informasi Misi

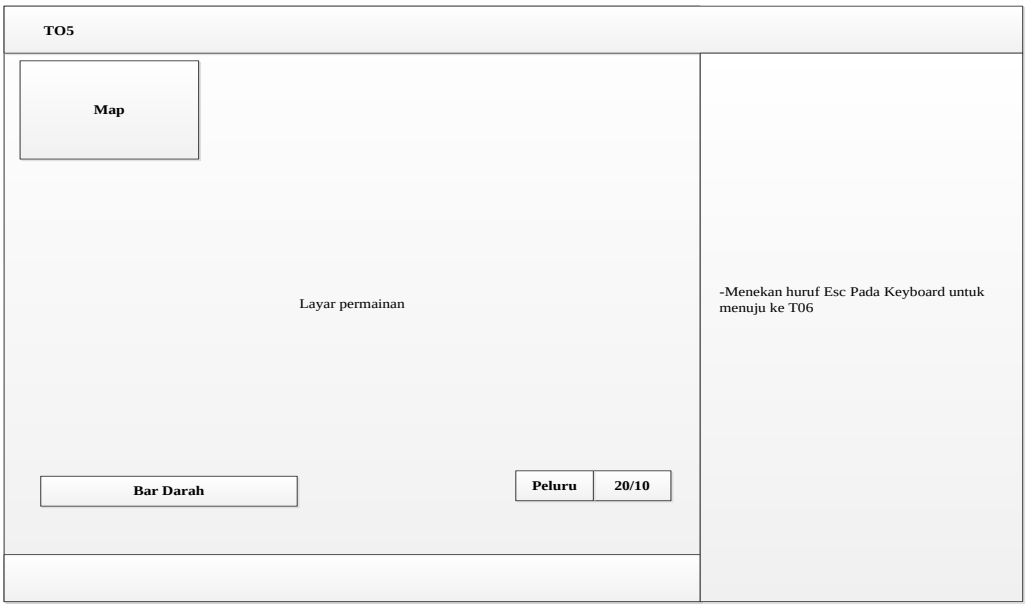
Perancangan antarmuka informasi misi digunakan untuk misi yang harus diselesaikan dalam permainan dapat dilihat pada gambar 3.47



Gambar 3.47 Tampilan Antarmuka Informasi Misi

3.6.2.5 Perancangan Antarmuka Permainan

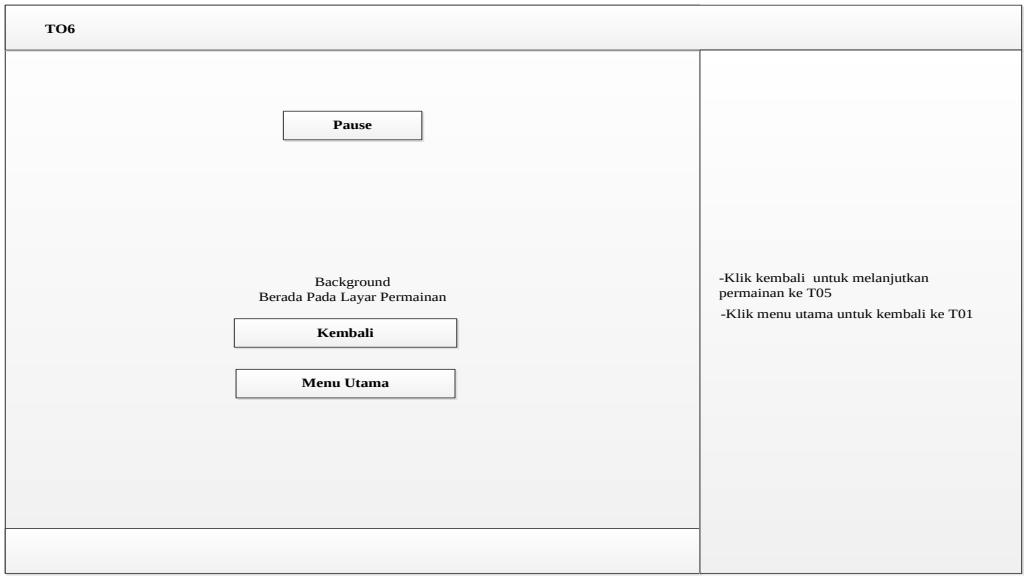
Perancangan antarmuka permainan pertempuran lima hari di semarang dapat dilihat pada gambar 3.48.



Gambar 3.48 Tampilan Antarmuka Permainan

3.6.2.6 Perancangan Antarmuka *Pause Menu*

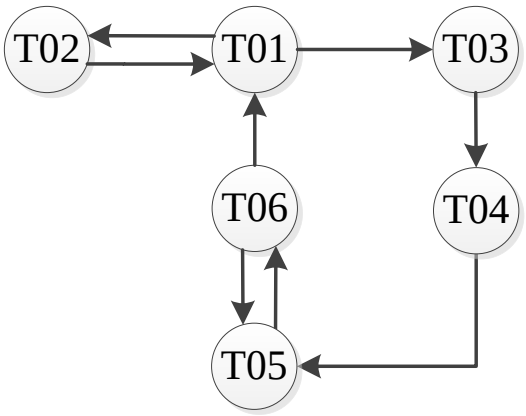
Perancangan antarmuka *pause menu* digunakan untuk melanjutkan permainan atau keluar permainan untuk kembali ke menu utama dapat dilihat pada gambar 3.49



Gambar 3.49 Tampilan Antarmuka *Pause Menu*

3.6.3 Jaringan Semantik

Gambaran pengetahuan grafis yang menunjukkan hubungan antar berbagai objek, terdiri dari lingkaran-lingkaran yang dihubungkan dengan anak panah yang menunjukkan objek dan informasi tentang objek-objek tersebut disebut jaringan semantik. Jaringan semantik dari *game* pertempuran lima hari di semarang ini bisa dilihat pada gambar 3.50.



Gambar 3.50 Jaringan Semantik

Keterangan:

T01: Antarmuka menu utama

T02: Antarmuka menu petunjuk

T03: Antarmuka scene video

T04: Antarmuka informasi misi

T05: Antarmuka permainan

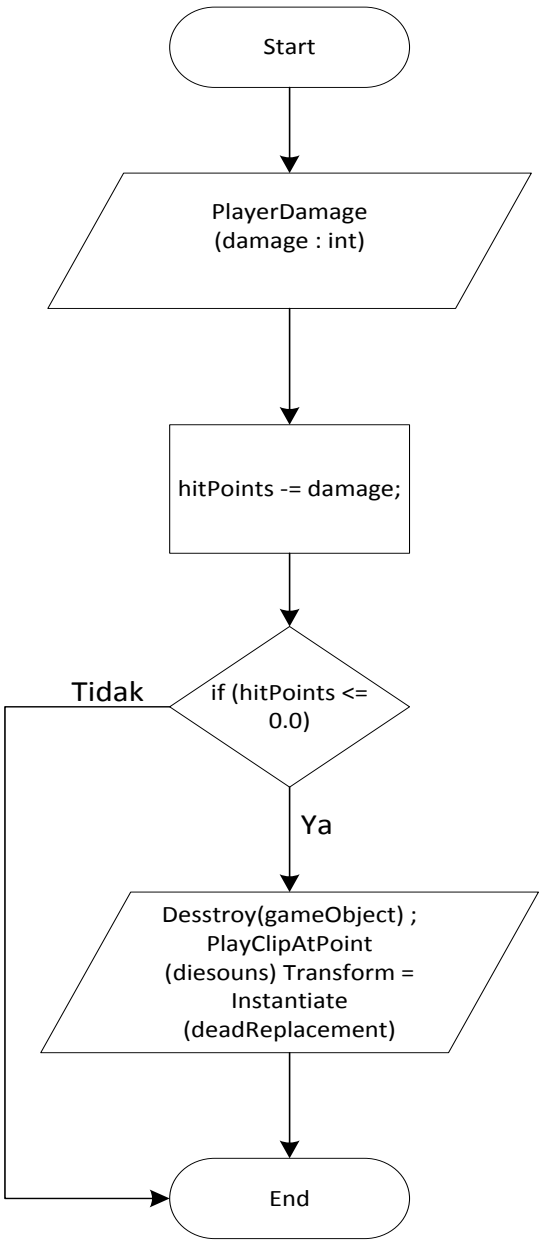
T06: Antarmuka pause menu

3.6.4 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* 3D Pertempuran Lima Hari di Semarang yang akan dibangun dijelaskan melalui *flowchart* sebagai berikut.

3.6.4.1 Method PlayerDamage

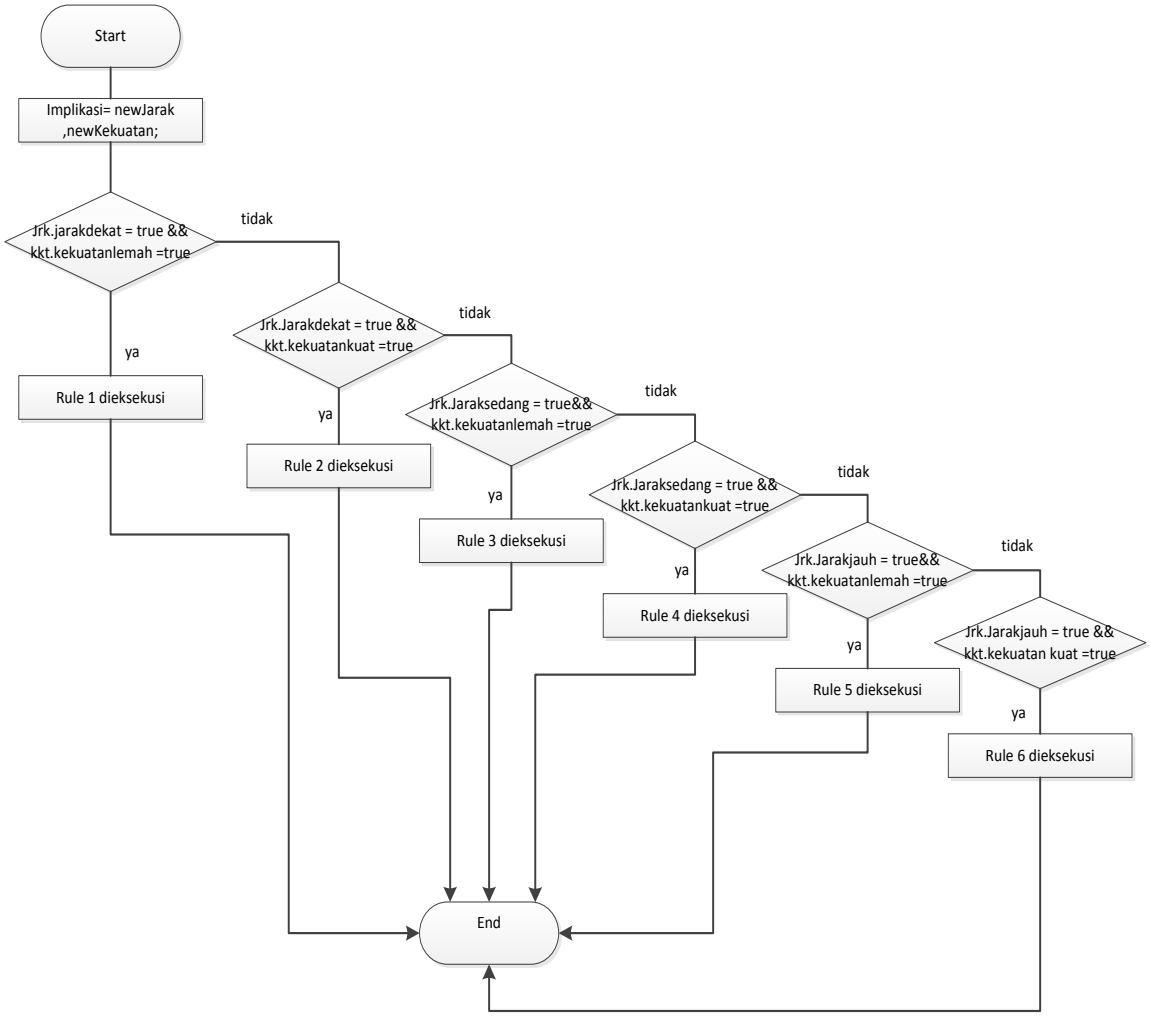
Method playerdamage digunakan untuk pengecekan terhadap setiap serangan musuh yang mengenai pemain bisa dilihat pada gambar 3.51.



Gambar 3.51 ***Flowchart Method CharacterDamage***

3.6.4.2 Method Periksarule

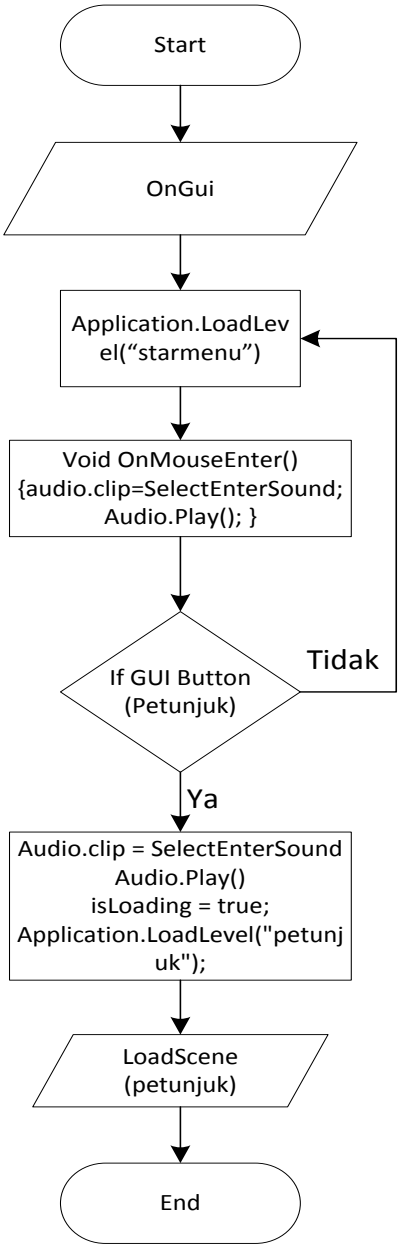
Method ini digunakan untuk melakukan pengecekan rule yang ada di area permainan, bisa dilihat pada gambar 3.52.



Gambar 3.52 *Flowchart Method* Periksarule

3.6.4.3 Method OnGui (Petunjuk)

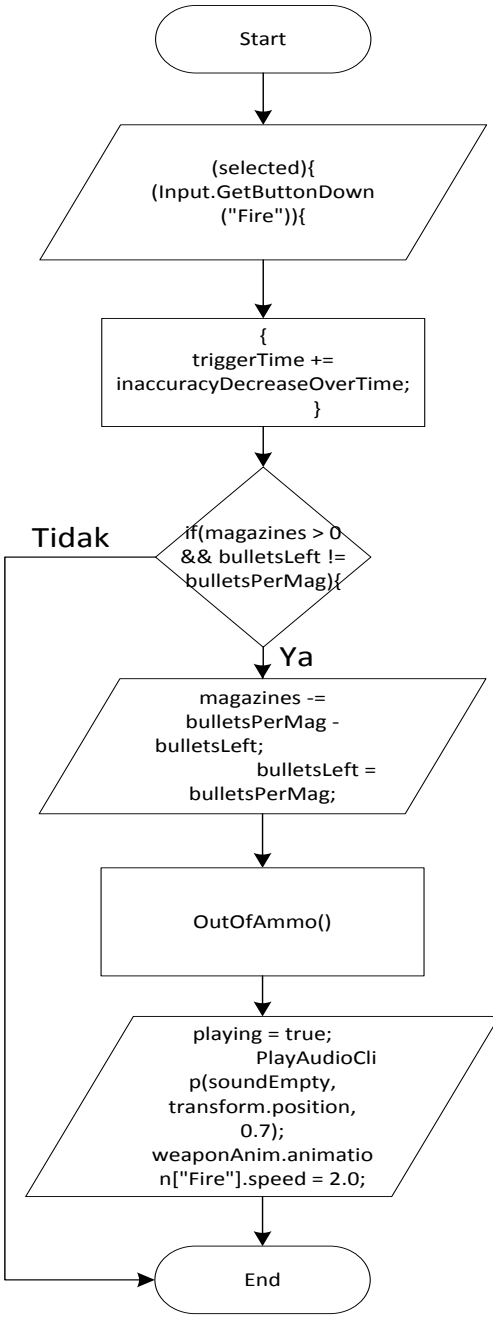
Method ini digunakan untuk melihat alur proses perancangan menu petunjuk bisa dilihat pada gambar 3.53



Gambar 3.53 *Method OnGui* (Petunjuk)

3.6.4.4 Method OnGui (*Weapon*)

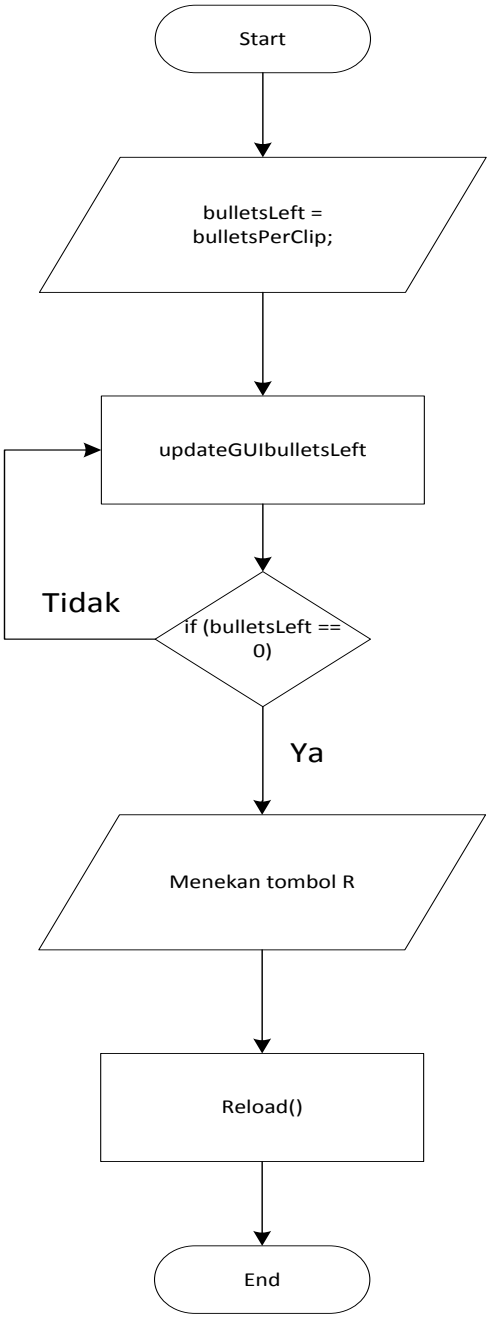
Method ini melakukan perubahan terhadap setiap peluru/*ammo* yang digunakan untuk menyerang musuh menggunakan senjata machinegun, bisa dilihat pada gambar 3.54.



Gambar 3.54 *Flowchart Method OnGui* (*Weapon*)

3.6.4.5 *Method Pistolreload*

Method ini berada pada *pistol*, digunakan untuk melakukan pengisian otomatis terhadap amunisi senjata jika telah habis *method* bisa dilihat pada gambar 3.55.



Gambar 3.55 ***Flowchart Method Pistolreload***