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, mulia dari analisis masalah, *storyline*, *gameplay*, analisis algoritma, analisis kebutuhan *non*-fungsional dan analisi kebutuhan fungsional.

3.2. Analisis Masalah

Berdasarkan tinjauan analisis sistem yang telah dilakukan dengan melakukan evaluasi terhadap *game* yang akan dibangun. Beberapa permasalahan yang merupakan hasil dari analisis yang dilakukan dijadikan sebagai latar belakang dalam pembangunan *game* Selamatkan Orangutan.

3.2.1. Analisis *Game* Sejenis

Analisis *game* sejenis dilakukan untuk memberikan gambaran sistem agar lebih mudah dalam pembangunan dan sebagai pembanding untuk kesempurnaan *game* yang akan dicapai.

3.2.1.1. *Animal Saver* 3D

Animal Saver 3D adalah *game* ber-*genre Third-person shooter* (TPS), *game* ini yang mempunyai *gameplay* tentang penyelamatan binatang dari tangan pemburu, *game* ini dibuat dengan menggunakan *Unity* yang mempunyai desain 3D (3 dimensi). Beberapa cuplikan permainan dari *game Animal Saver* 3D dapat dilihat pada gambar 3.1.



Gambar 3.1 Cuplikan Main *Menu Game Animal Saver* 3D

Gambar 3.1 adalah tampilan menu utama pada *game Animal Saver* 3D, dimana terdapat pilihan *menu* yaitu Mulai Main, Cara Main dan Keluar.



Gambar 3.2 Cuplikan *Game Animal Saver* 3D

Gambar 3.2 adalah tampilan *Level* 1 pada *game Animal Saver* 3D, dimana pemain diserang oleh musuh.

Game memiliki grafis dan alur cerita yang cukup baik, namun di dalam *game* masih terdapat kekurangannya yaitu tidak ada *crosshair* bidikan senjata, kontrol gerakan dari karakter polisi hutan tidak sevariatif dari *game third person shooter* sekarang ini seperti tidak ada gerakan berlari dan merunduk, tidak adanya

instruksi yang diberikan sehingga dapat membingungkan dalam mencari jalan keluar, tidak ada map atau peta sehingga akan membingungkan pemain untuk mencari posisi musuh. *Game Animal Saver* 3D memilki 5 *level*, pada setiap *level*nya misi pemain hanya mencari titik untuk menuju ke *level* selanjutnya dan membebaskan hewan yang ditangkap. Tidak adanya *item* seperti darah dan peluru. Peluru pemain tidak habis-habis sehingga tidak ada tatangan pada g*ame.*

3.2.1.1.1. *Storyline*

Game animal saver 3D menceritakan tentang polisi hutan yang sedang menyelamatkan hewan dari perburuan. Cerita awal ketika polisi hutan yang ditempatkan dihutan harus mencari sebuah jalan untuk menemukan kawanan pemburu liar yang menangkap hewan. Hewan yang ditangkap yaitu harimau, kudanil, gajah dan beruang. Berikut *level-level* yang harus dilalui oleh pemain :

1. *Level* pertama, pemain akan ditempatkan disebuah hutan, misi dari level pertama pemain harus mencari titik dimana titik tersebut adalah tempat untuk melanjutkan ke *level* selanjutnya.
2. *Level* kedua, pemain akan ditempatkan disebuah hutan yang lebat, misi pada *level* ini pemain harus mencari binatang yang telah ditangkap oleh pemburu, pemain akan diserang oleh pemburu yang sedang berjaga, pemain harus melumpuhkan musuh agar bintang diselamatkan dan harus mencari titik dimana titik tersebut adalah tempat untuk melanjutkan ke *level* selanjutnya.
3. *Level* ketiga, pemain akan ditempatkan disebuah padang pasir, misi pada *level* ini sama dengan misi pada *level* kedua. Musuh pada *level* ketiga akan bertambah.
4. *Level* keempat, pemain akan ditempatkan disebuah hutan pada malam hari. Misi pada *level* ini sama dengan misi pada *level* ketiga. Musuh pada *level* ketiga akan bertambah.
5. *Level* kelima, pemain akan ditempatkan disebuah rawa. Pemain harus mencari muhuh yang telah menangkap binatang, pencarian akan cukup sulit karena pemain sangat jauh dengan posisi musuh. jika pemain telah mengalahkan musuh maka muncul sebuah titik, dimana titik

tersebut adalah temapt untuk menyelesaikan misi. Jika pemain tepat berada dititik itu maka permainan berakhir.

3.2.1.1.2. *Gameplay*

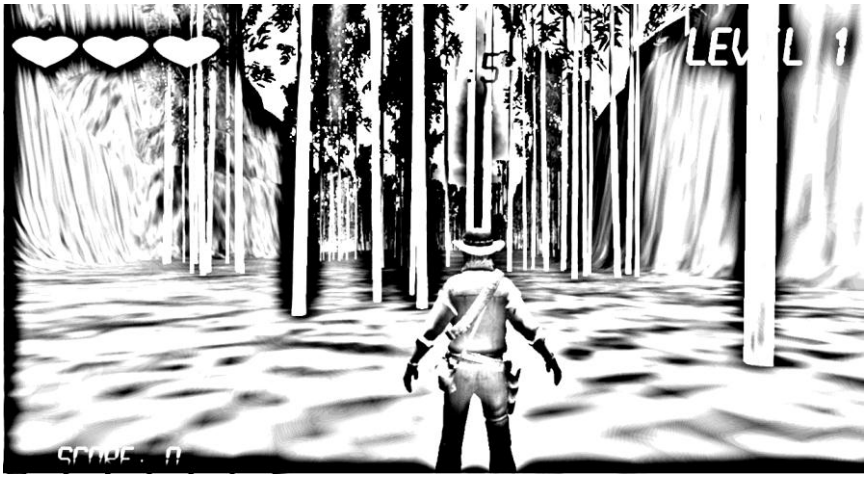
Misi utama pada *game* ini adalah membebaskan binatang dari pemburu. Pada *menu* utama memiliki beberapa *menu* yaitu main, petunjuk dan keluar. Pada *game animal saver* 3D ini tidak mempunyai item seperti darah, nyawa dan peluru, sementara peluru yang dipunyai oleh pemain tidak habis-habis.

3.2.1.1.3. Skenario *Game*

Skenario dalam permainan *Animal Saver* 3D dapat dilihat alur permainannya sebagai berikut :

a. Pemain berada di sebuah hutan.
b. Pemain harus mencari sebuah titik yang sudah ditentukan untuk menuju *level* selanjutnya.
c. Pemain harus menghindari dan membasmi musuh ketika berada di dalam arena permainan.
d. Pemain harus melepaskan binatang yang ditangkap oleh musuh dan pemain akan mendapatkan *score*.
e. Jika pemain terkena serangan dari musuh, maka *healthpoint* akan berkurang dan apabila habis maka permainan diulangi ke awal permainan.

3.2.1.1.4. *Story Board*



Gambar 3.3 *Storyboard Level* 1 pada *game Animal Saver* 3D

Gambar 3.3 Menceritakan ketika pemain memulai permainan di awal *level* dengan berada disebuah hutan, disinilah awal dari permainan. Pemain harus mencari titik dimana titik itu adalah tempat untuk melanjutkan ke *level* selanjutnya.



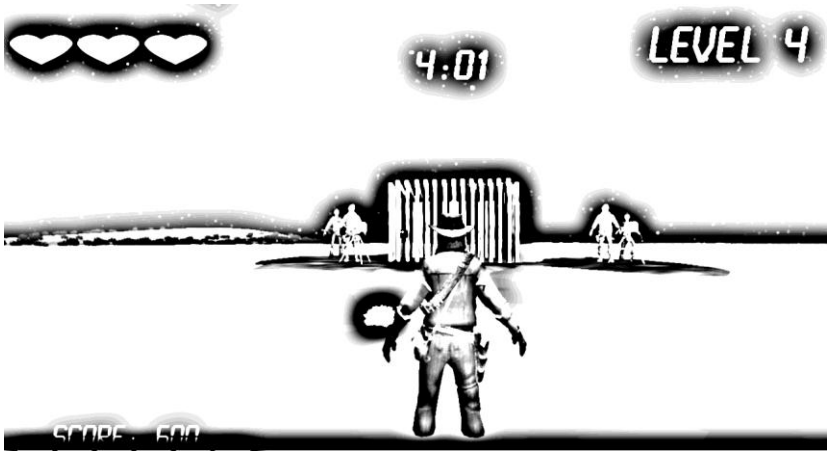
Gambar 3.4 *Storyboard Level* 2 pada *game Animal Saver* 3D

Gambar 3.4 Menceritakan ketika pemain yang diserang oleh musuh dan harus membebaskan hewan yang ditangkap. Pemain harus mencari titik dimana titik itu adalah tempat untuk melanjutkan ke *level* selanjutnya.



Gambar 3.5 *Storyboard Level* 3 pada *game Animal Saver* 3D

Gambar 3.5 Menceritakan ketika pemain yang berada disebuah padang pasir. Pemain akan diserang oleh musuh dan harus membebaskan hewan yang ditangkap. Pemain harus mencari titik dimana titik itu adalah tempat untuk melanjutkan ke *level* selanjutnya.



Gambar 3.6 *Storyboard Level* 4 pada *game Animal Saver* 3D

Gambar 3.6 Menceritakan ketika pemain akan diserang oleh musuh dan harus membebaskan hewan yang ditangkap. Pemain harus mencari titik dimana titik itu adalah tempat untuk melanjutkan ke *level* selanjutnya.



Gambar 3.7 *Storyboard Level* 4 pada *game Animal Saver* 3D

Gambar 3.7 Menceritakan ketika pemain ditempatkan disebuah rawa, pemain akan diserang oleh musuh dan harus membebaskan hewan yang ditangkap. Pemain harus mencari titik dimana titik itu adalah tempat untuk melanjutkan ke *level* selanjutnya.

3.2.1.1.5. Pengenalan Karakter pada *game Animal Saver* 3D

Karakter utama dalam g*ame* ini adalah seorang polisi hutan. *Game* ini merupakan *genre third person shooter* yang mempunyai sudut pandang orang ketiga, bentuk karakter utama dapat dilihat pada gambar 3.8



Gambar 3.8 Karakter utama pada *game Animal Saver* 3D

3.2.1.2. *Save* Orangutan

Game Save orangutan merupakan *game* buatan *dari developer game* Indonesia yaitu Garuda *Games*. *Game save* orangutan menceritakan tentang orangutan yang ingin diburu oleh pemburu. *game* ini dibuat dengan menggunakan UDK *(Unreal Engine)* yang mempunyai desain 3D (3 dimensi). Beberapa cuplikan permainan dari *game save* orangutan dapat dilihat pada gambar 3.9.



Gambar 3.9 cuplikan *Game Save* Orangutan



Gambar 3.10 *Tampilan Game Save* Orangutan

Gambar 3.10 adalah tampilan pada *game save* orangutan, dimana pemain harus melindungi orangutan ke sebuah penangkaran.

Game ini alur cerita yang cukup baik, menceritakan tentang kejadian masalah lingkungan yang terjadi di Indonesia dimana banyak hutan ditebang untuk membuka sebuah lahan perkebunan kelapa sawit, akibatnya populasi Orangutan menurun. Di dalam *game* masih terdapat kekurangannya yaitu pada setiap *level*nya mempunyai tampilan yang sama.

3.2.1.2.1. *Storyline*

Game save orangutan memilki jalan cerita tentang penyelamatan orangutan yang ingin diburu oleh pemburu. Semua *level* dari *game* ini memiliki tampilan yang sama tetapi pada tingkatan *level*nya memiliki rintangan yang berbeda. Semakin tinggi *level*nya maka akan semakin sulit untuk menyelesaikan *game* ini.

3.2.1.2.2. *Gameplay*

Misi utama pada *game* ini adalah menyelamatkan orangutan dan bekatan ke tempat penangkaran yang aman dengan meng-*drag and drop* pada karakter orangutan.

3.2.1.2.3. Skenario *Game*

Skenario dalam permainan *save* orangutan dapat dilihat alur permainannya sebagai berikut :

a. Orangutan dan bekatan berada disebuah hutan yang pohonnya telah ditebang.
b. Pemain harus meng-*drag and drop* karakter orangutan dan bekatan ke kiri atau kenan.
c. Pemain harus cepat mengambil orangutan, karana musuh akan mencari keberadaan orangutan dan mengambilnya
d. Jika pemain menyelamatkan orangutan ke tempat penangkaran maka akan mendapatkan *score* permaian.
e. Jika pemain gagal menyelamatkan orangutan maka permainan akan diulang.

3.2.1.2.4. *Story Board*



Gambar 3.11 *Storyboard* pada *game save* Orangutan

Gambar 3.11 Menceritakan ketika orangutan berada disebuah hutan yang pohonnya telah ditebang. Musuh sedang mencari orangutan dan menangkapnya.

3.2.1.2.5. Pengenalan Karakter pada *game Animal Saver* 3D

Karakter utama dalam g*ame* ini adalah orangutan dan bekatan. *Game* ini merupakan *genre Maze game*. bentuk karakter utama dapat dilihat pada gambar 3.12



Gambar 3.12 Karakter Orangutan dan Bekatan

3.2.1.3. *Game* Yang akan dibangun

Game Selamatkan Orangutan yang akan dibungun berupa *game* 3D yang ber*genre Third person shooter* atau TPS. *Game* ini mempunyai *genre*, karakter dan desain grafis yang sama dengan *game Animal Saver* 3D. dan *Game* selamatkan orangutan juga mempunyai *storyline* atau jalan cerita yang sama dengan *game save* orangutan yaitu menyelamatkan orangutan kesebuah penangkaran.

Game dibangun dengan konsep sudut padang ketiga atau TPS agar pemain dapat melihat karakter dari seorang polisi hutan Indonesia dimana terdapat atribut dari kepolisian yang menangani hutan di Indonesia dan pemain juga dapat melihat animasi pergerakan seperti berjalan, berlari, merunduk dan melompat. Berikut ini adalah beberapa *fitur* yang ditawarkan di dalam *game* ini.

1. Grafik *game* 3D
2. *Game* memiliki 3 *Level* dimana seteiap *level*-nya memilki misi dan tantangan yang berbeda.
3. Mengangkat tema tentang penyelamatan orangutan.
4. *Game* ber*genre Third person shooter* atau TPS.
5. Memperlihatkan karakter dari polisi hutan Indonesia.
6. Senjata yang digunakan adalah senjata yang dimiliki oleh kepolisian hutan pada umumnya.
7. Memiliki model rumah khas Indonesia yaitu rumah panggung.
8. *Game* ini mempunyai menu *pause* sehingga dapat dihentikan ketika permainan sedang berlangsung.
9. Karakter Musuh yang bervarian dan memiliki kecerdasan yaitu untuk memilih keputusan dengan kondisi tertentu.

3.2.1.3.1. Perbandingan *Game* yang akan dibangun dengan *Game* sejenis

Secara garis besar perbandingan *game* yang dibangun dengan *game* sejenis dijelaskan dengan tabel berikut ini.

Table 3.1 Perbandingan *Game* yang dibangun dengan *Game* sejenis

	***Game Animal Server* 3D**	***Game Save* Orangutan**	***Game* yang akan dibangun**
Genre Game	*Third person shooter*	*Maze Game*	*Third person shooter*
Desain Grafis	3D	3D	3D
Karakter Utama	Polisi hutan yang terlihat seperti *cowboy*	Orangutan dan bekatan	Polisi hutan dengan memakai atribut dan senjata yang dimiliki polisi hutan pada umumnya
Pergerakan Animasi	Pergerakan karakter berupa berjalan dan melompat	Pergerakan karakter berupa berjalan	Pergerakan karakter berupa berjalan, berlari, melompat, merunduk, *reload* senjata dan mengganti senjata
Enemy	Memilki 2 karakter musuh yang berbeda	Hanya 1 karakter musuh	Memiliki 3 karakter musuh yang berbeda
Storyline	Menyelamatan binatang yang telah ditangkap oleh pemburu dan pemain harus membebaskannya, setiap *level*-nya	Menyelamatkan orangutan ke tempat penangkaran.	Menyelamatkan binatang khas Indonesia yaitu Orangutan, pada setiap *level*-nya memilki misi dan tantangan yang

	memilki alur cerita yang sama		berbeda-beda.
Item	-	-	Nyawa, Darah dan Peluru
Basis	*Desktop*	*Mobile*	*Desktop*

Dari tabel diatas ada beberapa yang berkaitan antara *game* sejenis dengan *game* yang dibangun diantaranya Karakter dan *Storyline* yang sama namun pada *game* yang dibangun memiliki perbedaan yaitu :

1. Dari segi karakter, pada *game* selamatkan orangutan memiliki karakter utama seorang polisi hutan yang mempunyai atribut dan senjata polisi hutan Indonesia pada umumnya. Pergerakan pada karakter utama sangat variatif, berupa pergerakan berjalan, berlari, melompat, merunduk, reload peluru dan menganti senjata.
2. Dari segi *Storyline*, pada *game* selamatkan orangutan mempunyai jalan cerita yaitu harus menyelamatkan orangutan ke sebuah penangkaran, perbedaan dengan *game save* orangutan adalah pemain akan mencari dan membawa orangutan ke sebuah penangkaran, dalam penyelamatan orangutan pemain akan diserang oleh musuh sehingga *game* lebih menantang.

3.2.2. Analisis *game* yang dibangun

Game yang akan dibangun berjudul Selamatkan Orangutan. *Game* ini bergenre *third person shooter*, *Game* ini memiliki grafis 3D (3 Dimensi). Memiliki alur cerita tentang seorang polisi hutan yang berusaha menyelamatkan habitat orangutan dari perburuan liar yang diakibatkan oleh pembukaan lahan kebun kelapa sawit ilegal.

3.2.2.1. *Storyline*

Game Selamatkan Orangutan menceritakan tentang seorang polisi hutan yang diberikan misi untuk menyelamatkan habitat orangutan yang ada dihutan. Cerita awal ketika seorang polisi hutan yang bernama Sonny ditugaskan untuk pergi kesuatu hutan, disana Sonny akan bertemu dengan warga dari desa yang sedang mencari kayu. Misi pertama sonny mencari jalan ke desa yang bernama Wajok Hilir dan harus mencari lalu menghancurkan perangkap binatang yang diterdapat disepanjang jalan menuju ke desa.

Ketika Sonny berada di Desa Wajok Hilir, Sonny disambut oleh kepala desa yang memberitahukan bahwa ada penangkapan orangutan sehingga membuat habitat orangutan terganggu. Sonny pun berinisiatif menyelasaikan masalah-masalah yang ada untuk segera diselesaikan. Berikut *level-level* yang harus dilalui

1. *Level* pertama, pemain akan ditempatkan disebuah hutan dan akan disambut oleh warga desa yang sedang mencari kayu. Pemain akan berintraksi dengan warga desa dan menanyakannya jalan untuk sampai ke desa. Misi dari permain ialah harus mencari jalan untuk sampai ke desa Wajok Hilir, dimana dalam mencari desa itu pemain akan diberi rintangan seperti rintangan jurang dimana pemain harus melawati satu batang pohon yang akan menjadi jembatan penyebrangan, tanjakan yang curam dimana terdapat runtuhan batu dan harus melewati jurang dengan melewati jembatan yang rapuh. Pemainpun harus mencari perangkap bintang yang tersimpan didalam hutan. Jika pemain telah menemukan dan menghancurkan perangkap banatang lalu pemain akan sampai di Desa.

2. *Level* kedua, pemain akan berada didesa desa Wajok Hilir, dimana Pemain akan disambut oleh kepala desa yang memberitahukannya tentang masalah-masalah yang merenggut habitat orangutan. pemain harus mencari orangutan yang ditangkap oleh pemburu, jika pemain menemukan kurungan yang dijaga oleh pemburu maka, pemain akan memberi intruksi kepada pemburu bahwa pemburuan orangutan itu tidak baik, jika pemain melakukan pergerakan, maka pemburu akan menyerang pemain, dan pemainpun harus melumpuhkan pemburu itu dan membuka kurangan lalu membawa orangutan ke penangkaran dimana pemain harus mencari letak penangkaran itu. Pada pencarian orangutan pemain akan diberi tantang seperti harus melewati runtuhan batu, melawati pasir penghisap.
3. *Level* ketiga, marupakan *level* terakhir pemain akan ditempatkan disebuah kawasan pembuat minyak kelapa sawit ilegal. Disana pemain akan diserang oleh *boss* yang memerintahkan anak buahnya untuk menyerang pemain. Pemain harus mencari dan menyelamatkan orangutan yang ditangkap kesebuah tempat yang aman dan pemain harus menghancurkan tempat pengolahan minyak kelapa sawit yang dimiliki oleh *boss* dari tempat itu. Jika orangutan telah diselamatkan dan tempat pengolahan minyak kelapa sawit ilegal telah dihancurkan maka permainan akan berakhir dan misi terselesaikan.

3.2.2.2. *Gameplay*

Misi utama dari *game* ini adalah menyelamatkan orangutan dari pemburuan liar yang disebabkan oleh orang-orang yang membangun kawasan pembuatan minyak kelapa sawit ilegal. Berikut ini *gameplay* yang ada didalam *game* Selamatkan orangutan :

1. Terdapat tiga *level* di mana setiap *level*-nya mempunyai misi yang berbeda-beda.
2. Pada setiap *level*, akan terdapat keterangan misi bagi pemain untuk mempermudahkan pemain menyelesaikan misi permainan.

3. Pada setiap *level* Pemain akan mendapatkan beberapa *item* seperti *item* nyawa, *item* darah dan *item* peluru agar dapat mempermudah dalam menyelesaikan misi disetiap *level*-nya.
4. Pada *game* Selamatkan orangutan terdapat waktu sebagai tantangan waktu untuk pemain. Pemain harus menyelesaikan misi dengan batas waktu yang ditentukan, tantangan waktu akan ada pada *level* 2 dan 3. Jika pada batas waktu yang ditentukan pemain belum menyelesaikan misi maka pemain gagal menyelesaikan misi dan harus mengulang lagi dari awal.
5. Senjata yang digunakan dalam *game* ini berupa *Forest Guard Gun* PM2-V2 dan *Forest Guard Gun* PM2-V3 yang dipakai oleh polisi hutan untuk melumpuhkan musuh-musuh yang ada disetiap *level*-nya. Kedua senjata itu digunakan pada saat permainan dimainkan.
6. *Game* selamatkan orangutan memiliki *minimap* agar mempermudah pemain untuk melihat posisi perangkap, posisi musuh, posisi orangutan dan posisi penangkaran.
7. Memiliki posisi petunjuk *icon* yaitu berupa keterangan *icon* musuh, perangkap, rintangan, posisi desa, penangkaran dan intraksi dengan warga desa agar dapat mempermudah dalam mengetahui petunjuk yang ada.
8. Memiliki gambaran berupa keterangan tentang orangutan setiap pemain memulai misi setiap *level*-nya.

Pada *gameplay* dari semua *level*-nya memiliki *gameplay* yang sama adapun *gameplay* yang berbeda. Berikut ini deskripsi *gameplay* dari semua *level* yang ada dalam permainan yang dijelaskan pada tabel berikut ini.

Tabel 3.2 Penjelasan *Gameplay*

Level	**Deskripsi *Gameplay***
Level 1	1. Pemain harus mencari jalan untuk sampai ke sebuah desa. 2. Pemain mendapatkan misi untuk mencari dan menghancurkan perangkap binatang yang terdapat disetiap jalannya. Jika pemain menemukan

	perangkap bintang maka harus dihancurkan. Posisi perangkap dapat dilihat pada *minimap.* 3. Tidak terdapat musuh pada *level* pertama. 4. Rintangan yang ada pada *level* pertama adalah pemain harus melewati jurang yang dimana jurang tersebut diletakan sebuah pohon yang akan menjadi jembatan, melewati sebuah tanjakan curam yang dimana terdapat runtuhan batu yang akan membuat darah pemain berkurang, melewati jurang dimana jembatan yang terdapat pada jurang itu rapuh jadi pemain harus berhati-hati melewati jembatan itu. Jika pemain jatuh ke jurang maka nyawa pemain akan berkurang. Jika item nyawa pemain habis maka pemain akan mengulangi misi dari awal.
Level 2	1. Pemain akan berada disebuah desa, dimana akan berintraksi dengan kepala desa. 2. Misi pada *level* kedua adalah mencari orangutan yang dikurung oleh pemburu dan membawanya ke sebuah penangkaran. Jika pemain telah membawa orangutan kepenangkaran maka pemain harus mencari sebuah gerbang untuk sampai ke *level* selanjutnya. 3. Orangutan yang harus diselamatkan mempunyai nyawa, dimana pemain harus menghidari tembakan dari NPC agar peluru tidak mengenai orangutan, jika mengenai orangutan maka darah orangutan berkurang dan jika orangutan mati maka misi gagal. 4. Terdapat musuh yaitu pemburu dan anak buah *boss* yang akan menyerang pemain. Pemain tidak harus melumpuhkan semua musuh melainkan pemain

	harus harus melumpuhkan musuh jika musuh tersebut sedang mengikuti dan menyerang pemain. 5. Rintangan pada level 2 adalah terdapat runtuhan batu dimana pemain harus menghindar batu itu dan pasir penghisap dimana terdapat blok-blok yang harus dilewati pemain, jika blok itu jatuh maka pemain akan jatuh ke dalah pasir penghisap dan nyawa pemain berkurang.
Level 3	1. Pada *level* terakhir pemain ditempatkan disebuah pabarik pengolahan minyak kelapa sawit ilegal. 2. Pemain akan diserang oleh boss yang memiliki kemampuan tidak seperti musuh yang lainnya, tetapi boss memiliki kemampuan seperti pemain untuk mengambil keputusan menyranag, menghindar atau mengambil darah. 3. Misi pada level terakhir ini pemain harus mencari dan membawa orangutan kesebuah tempat yang aman dan misi selanjutnya pemain harus menghancurkan tempat pengolahan minyak kelapa sawit.

3.2.3. Analisis *Level*

Pada *game* Selamatkan Orangutan memiliki 3 tingkat *level* yaitu *Level* mudah, sedang dan sulit. Berikut ini penjelasan tingkatan *level* dapat dilihat pada tabel 3.3

Tabel 3.3 Penjelasan Tingkat *Level*

Tingkat *Level*	Deskripsi
Mudah	Pada *game* Selamatkan Orangutan dengan *level* mudah terdapat pada *level* 1. Misi pada *level* pertama hanya mencari dan menghancurkan 5 perangkap binatang

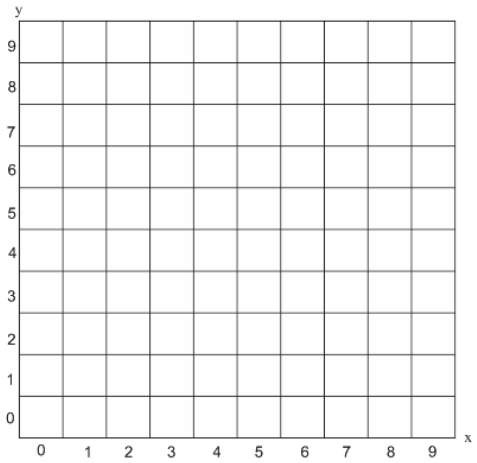
	yang dibuat oleh pemburu dan harus mencari sebuah desa dengan diberi tantang yaitu harus melewati jurang yang terdapat 1 buah pohon sebagai jembatannya, melewati tanjakan yang curam dan harus berhati-hati dengan longsoran batu, dan jemabatan yang berputar. Terdapat 3 *item* darah, 2 *item* nyawa dan 3 *item* peluru.
Sedang	Pada *game* Selamatkan Orangutan dengan *level* sedang terdapat pada *level* 3. Misi pada *level* ketiga mencari dan menyelamatkan 2 orangutan ke tempat yang aman serta harus menghancurkan tempat pengolahan minyak kelapa sawit ilegal yang dimiliki oleh *boss* dari kawasan itu. *Level* 3 terdapat 1 musuh *boss* dan 10 anak buah *boss* yang akan menyerang pemain. Akan terdapat 3 item yaitu 3 *item* nyawa, 10 *item* darah dan 10 *item* peluru. Tantangan waktu yang diberikan pada level 3 yaitu dengan waktu 8 menit.
Sulit	Pada *game* Selamatkan Orangutan dengan *level* sulit terdapat pada *level* 2. Pada *level* kedua pemain mempunyai misi mencari dan menyelamatkan 5 orangutan yang ditangkap lalu membawanya ke sebuah penangkaran yang dijaga oleh warga desa yang membantu pemain untuk menyelamatkan orangutan. Akan terdapat musuh yang akan menyerang pemain. jumlah musuh yaitu 10 pemburu dan 10 anak buah *boss* yang akan menyerang pemain. Terdapat 5 *item* nyawa, 10 *item* darah dan 10 *item* peluru pada *level* 2 ini. Tantangan waktu yang terdapat yaitu 10 menit untuk menyelesaikan misi pada *level* 2.

Tujuan pe*level*-an pada *game* Selamatkan Orangutan ini yaitu untuk mengembangkan permainan agar tidak bosan, pada dasarnya dalam sebuah *game* ada tingkat kesulitan yang dapat berubah-ubah dimana tingkat kesulitan permainan didasarkan pada aksi-aksi pemain dengan memberikan banyak pilihan aktifitas dengan berbagai tingkat kesulitan di dalam permainan.

3.2.4. Analisis Algoritma A* (A *Star*)

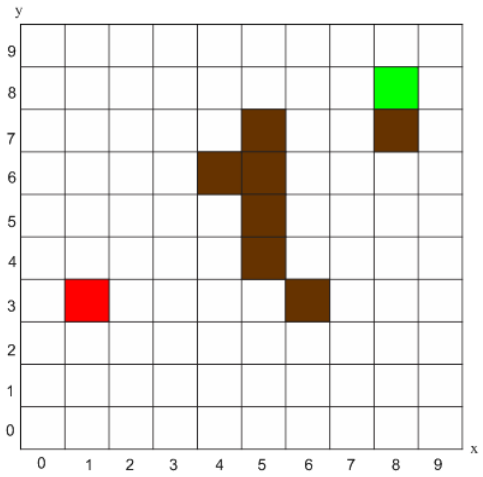
Algoritma A* (*Astar*) akan diterapkan pada NPC (*No Player Character*) sebagai penentu jalur terpendek atau *pathfinding* menuju target yaitu *player*.

Analisis perhitungan dengan algoritma A* pada *pathfinding* ini akan dijelaskan dengan ordo X x Y. setiap *Node* memiliki koordinat X dan Y berbentuk seperti *array* dua dimensi. Pada ilustrasi perhitungan A* dibawah akan dijelaskan dengan ordo 10x10 dikarenakan perhitungan dengan ordo berapapun hasil perhitungannya akan sama. Berikut ini gambaran dari ilustrasi A* dengan ordo 10x10.



Gambar 3.13 *Node* dengan koordinat X dan Y

Pada gambar 3.13 terdapat beberapa kotak yang memilki koordinat dengan ordo 10x10. kotak atau *node* yang masing-masing telah memiliki koordinat merupakan *node* yang digunakan untuk melakukan pencarian jalur terpendek atau *pathfinding*. Berikut ini penempatan karakter *player*, musuh dan penghalang pada ilustrasi A*.



Gambar 3.14 Penempatan *Player*, Musuh dan Penghalang pada *Node*

Pada gambar 3.14 terdapat *node* yang memilki warna berbeda, berikut penjelasannya dari masing-masing *node* tersebut :

1. *Node* berwarna merah adalah *node* awal dari posisi *Non Player Character* (NPC) yang berada pada koordinat (1.3).
2. *Node* berwarna coklat adalah *node* penghalang (*collider*) yang berada pada koordinat (4.6), (5.4), (5.5), (5.6), (5.7), (6.3) dan (8.7).
3. *Node* berwarna adalah *node* dari posisi pemain atau *player* yang berada pada *node* di koordinat (8.8).

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

1. Nilai F adalah nilai yang diperoleh dari penjumlahan nilai G dan H.
2. Nilai G adalah nilai atau biaya yang dikeluarkan dari *node* awal ke *node* yang ada disekitarnya.
3. Nilai H adalah nilai *heuristic* yang digunakan untuk mengitung perkiraan biaya yang dikeluarkan dari *node* awal menuju *node* akhir.

Perhitungan nilai F didapatkan dari persamaan F=G+H. Nilai G adalah Nilai dari *node* ke *node* yang akan dihitung secara *vertical* atau *horizontal*, nilai G akan digunakan dengan nilai 10. Sedangkan untuk nilai dari *node* ke *node* yang akan dihitung secara *diagonal* mengunakan metode *Euclidean* dengan perhitungan nilai *node* ke *node* yang akan dihitung dikali $\sqrt{2}$ yang berarti $10x\sqrt{2} = 14.14$ Untuk mempermudah perhitungan 14.14 dibulatkan menjadi 14. Dan untuk pehitungan

nilai H digunakan fungsi *heuristic*, metode yang digunakan adalah metode *Manhattan* dimana perhitungan jumlah *node* hanya yang bergerak secara *vertical* dan *horizontal* menuju tujuan serta mengabaikan penghalang atau *collider*, yang kemudian nilainya dikalikan Nilai *node*. Atau dirumuskan dengan :

$$H = 10*(\text{abs}(current\text{X} - target\text{X}) + \text{abs}(current\text{Y} - target\text{Y})) \quad (3.1)$$

Pada pencarian jalur terpendek oleh ada beberapa istilah yang digunakan seperti *open list, closed list, parent, current node* dan *pointer.*

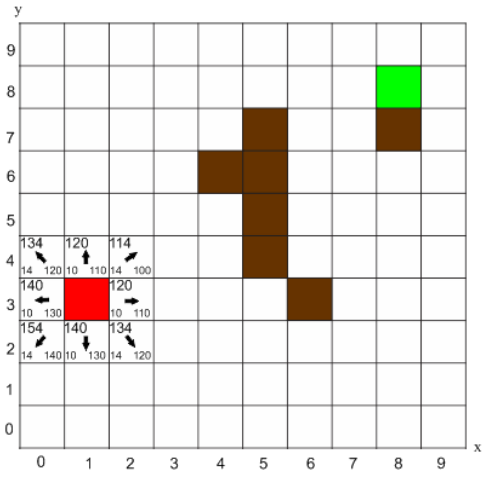
1. *Open list* adalah *list* untuk memeriksa *node-node* yang akan dihitung.
2. *Closed list* adalah *node-node* yang telah menjadi *node* terbaik atau *node* yang pernah mempunyai nilai terkecil.
3. *Parrent* adalah *node* yang telah mempunyai nilai terkecil untuk menelusuri jalan.
4. *Current node* adalah *node* dengan nilai terkecil yang sedang dijalankan dalam algoritma pencarian jalan terpendek.
5. *Pointer* adalah penunjuk jalan terhadap *node* awal ketika *node* akhir telah ditemukan.

Langkah berikutnya ketika telah mendapatkan *node* awal adalah menentukan rute terpendek dengan cara memeriksa setiap *node* yang berdekatan dengan *nod*e awal sampai *node* akhir. Dibawah ini merupakan penjelasan langkah-langkahnya.

1. Langkah pertama adalah memasukan *node* awal ke dalam *open list.*
2. Ambil *node* awal sebagai *current node*.
3. Periksa apakah *current node* sama dengan *node* akhir atau *current node* yang sudah didapatkan merupakan *node* akhir. Jika iya maka jalan sudah ditemukan, jika tidak maka hapus *node* awal dari *open list* dan tambahkan ke dalam *closed list.*
4. Periksa semua *neighbor node* atau *node* yang berdekatan dengan *node* awal apakah *walkable* atau tidak, abaikan *node* dengan penghalang atau *collider* lalu tambahkan semua *node* tersebut ke dalam *open list*. Dan untuk setiap *node* yang baru ditambahkan,

simpan *node* awal sebagai *parent* yang nantinya akan digunakan untuk menelusuri jalan.

Untuk lebih jelasnya proses perhitungan dijelaskan pada ilustrasi gambar 3.15.



Gambar 3.15 Penjelasan Algoritma A* pada langkah pertama

Pada langkah pertama, *node* (1.3) dimasukan ke dalam *open list* dan selanjutnya memeriksa *neighbor node* atau *node* yang berdekatan dengan *node* (1.3) yaitu node (0.2), (0.3), (0.4), (1.2), (1.4), (2.2), (2.3), (2.4). setelah diperiksa *node-node* tersebut dimasukan ke dalam *open list*. Hapus *node* (1.3) dari *open list* dan masukan node (1.3) kedalam *closed list.* Lalu dari *node-node* yang berada didalam *open list* pilih *node* dengan nilai F yang paling rendah, yaitu *node* (2.4) dengan nilai F=114. Perhitungan A* akan diilustrasikan dengan contoh dari *node* 2.3, berikut perhitungannya.

Menghitung node (2.3).

nilai n.x= 2 nilai n.y = 3 (3.2)

Node Tujuan (8.8) sehingga *targetX* = 8 dan *targetY* = 8

G(2.3) = 10 (nilai node ke node yang akan dihitung)

H(2.3) = 10*(abs(*current*X-*target*X) + abs(*current*Y-*target*Y))

= 10*(abs(2-8) + abs(3-8))

= 10*(abs(-6) + abs(-5))

= 10*(6+5)

= 10*11

= 110

F(2.3) = 10+110

= 120

Perhitungan A* pada langkah pertama semua langkah-langkahnya perhitungannya akan sama yang membedakan dari nilai G secara diagonal dengan menggunakan metode *Euclidean* atau nilai G dikali √2. untuk hasil perhitungan dari langkah-langkah perhitungan A* yang pertama akan dijelaskan dengan tabel berikut ini.

Tabel 3.4 Tabel Hasil Perhitungan A* pada langkah pertama

Node	Nilai G	Nilai H	Nilai F
0.2	14	140	154
0.3	10	130	140
0.4	14	120	134
1.2	10	130	140
1.4	10	110	120
2.2	14	120	114
2.3	10	110	120
2.4	14	100	134

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

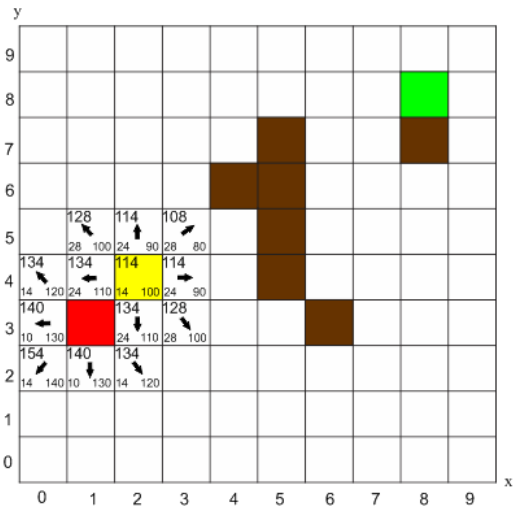
Tabel 3.5 Tabel Perhitungan langkah pertama

Open List	(0.2), (0.3), (0.4), (1.2), (1.4), (2.2), (2.3), (2.4).
Closed List	(1.3)
Parent	(1.3)
Currrent node	(2.4)

Proses dijelaskan diatas, 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 ke dalam *open list* dan masukan ke dalam *closed list.*
2. Periksa semua *node* yang berdekatan dengan *current node*, abaikan *node* yang *unwalkable* atau node dengan penghalang atau *collider*. Jika tidak ada pada *open list,* tambahkan ke dalam *open list.* Dan tambahkan *parent* untuk *node* tersebut.
3. Jika *node* sudah ditemukan pada *open list*, periksa apakah ini jalan yang lebih baik dari *node* awal, dengan kata lain periksa apakah nilai G lebih rendah apabila menggunakan *node* ini, jika lebih rendah maka hitung ulang niali F dan G.

Berikut ini perhitungan pada langkah kedua yang berada pada *node* dengan koordinat (2.4).



Gambar 3.16 Penjelasan Algoritma A* pada langkah kedua

Pada gambar 3.16 terlihat ada 7 *node* yang masuk ke dalam *open list* yaitu *node* dengan koordinat (1.4), (1.5), (2.3), (2.5), (3.3), (3.4) dan (3.5) sedangkan untuk *node* (1.3) dan (2.4) merupakan *node closed list.* Dari langkah kedua *node* yang masuk ke dalam *open list* terdapat niali F yang paling rendah

yaitu *node* (3.5) dengan nilai F = 108. Untuk hasil perhitungan dari langkah-langkah perhitungan A* yang kedua akan dijelaskan dengan tabel berikut ini.

Tabel 3.6 Tabel Hasil Perhitungan A* pada langkah kedua

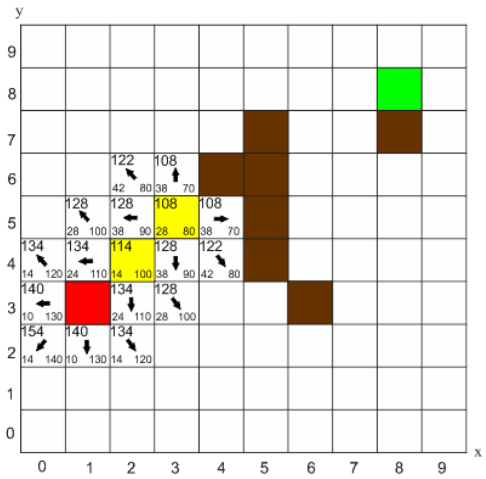
Node	Nilai G	Nilai H	Nilai F
1.4	24	110	134
1.5	28	111	128
2.3	24	110	134
2.5	24	90	114
3.3	28	100	128
3.4	24	90	114
3.5	28	80	108

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

Tabel 3.7 Tabel Perhitungan langkah kedua

Open List	(1.4), (1.5), (2.3), (2.5), (3.3), (3.4), (3.5)
Closed List	(1.3), (2.4)
Parent	(2.4)
Currrent node	(3.5)

Pada gambar 3.17 menunjukan perhitungan pada langkah ketiga yang berada pada *node* dengan koordinat (3.5).



Gambar 3.17 Penjelasan Algoritma A* pada langkah ketiga

Pada gambar 3.17 terlihat ada 6 *node* yang masuk ke dalam *open list* yaitu *node* dengan koordinat (2.5), (2.6), (3.4), (3.6), (4.4)dan (4.5) sedangkan untuk *node* (1.3), (2.4) dan (3.5) merupakan *node closed list*, sementara *node* (4.6) merupakan penghalang (*collider*). Dari langkah ketiga *node* yang masuk ke dalam *open list* terdapat niali F yang paling rendah yaitu *node* (4.5) dengan nilai F = 108 dan *node* (3.6) dengan nilai F = 108. jika kondisi seperti ini terjadi maka sistem akan memeriksa *node-node* yang masih berada di *open list* yaitu *node* (2.5), (2.6), (3.4) dan (4.4) untuk dicek apakah nilai dari *node* tersebut tidak lebih kecil dari *node* (3.6) dan (4.5). karena nilai dari *node* (2.5), (2.6), (3.4) dan (4.4) tidak lebih kecil, maka sistem akan memilih *node* (3.6) sebagai *node* dengan nilai F lebih kecil meskipun nilai F *node* tersebut sama dengan *node* (4.5). hal ini disebabkan sistem akan menentukan nilai F pada *node* terakhir yang ditambahkan ke dalam *open list*. Untuk hasil perhitungan dari langkah-langkah perhitungan A* yang ketiga akan dijelaskan dengan tabel berikut ini.

Tabel 3.8 Tabel Hasil Perhitungan A* pada langkah ketiga

Node	Nilai G	Nilai H	Nilai F
2.5	38	90	128
2.6	42	80	122
3.4	38	90	128

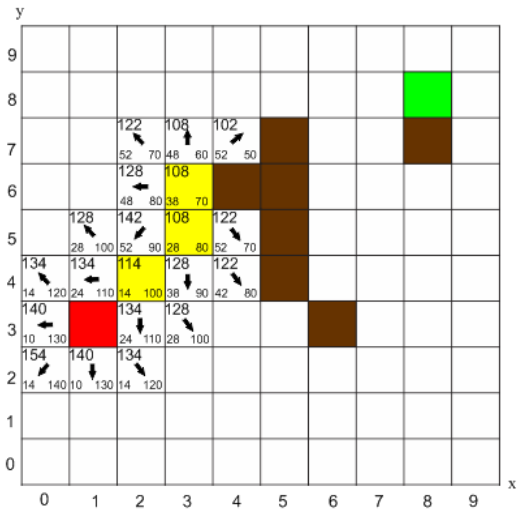
3.6	38	70	108
4.4	42	80	122
4.5	38	70	108

Berikut ini penjelasan melalui tabel untuk pencarian jalur terpendek pada langkah ketiga.

Tabel 3.9 Tabel Perhitungan langkah ketiga

Open List	(2.5), (2.6), (3.4), (3.6), (4.4), (4.5)
Closed List	(1.3), (2.4), (3.5)
Parent	(3.5)
Currrent node	(3.6)

Pada gambar 3.18 menunjukan perhitungan pada langkah keempat yang berada pada *node* dengan koordinat (3.6).



Gambar 3.18 Penjelasan Algoritma A* pada langkah keempat

Pada gambar 3.18 terlihat ada 6 *node* yang masuk ke dalam *open list* yaitu *node* dengan koordinat (2.5), (2.6), (2.7), (3.7), (4.5) dan (4.7) sedangkan untuk *node* (1.3), (2.4), (3.5) dan (3.6) merupakan *node closed list,* sementara *node* (4.6) merupakan penghalang (*collider*). Dari langkah kedua *node* yang masuk ke dalam *open list* terdapat niali F yang paling rendah yaitu *node* (4.7)

dengan nilai F = 102. Untuk hasil perhitungan dari langkah-langkah perhitungan A* yang keempat akan dijelaskan dengan tabel berikut ini.

Tabel 3.10 Tabel Hasil Perhitungan A* pada langkah keempat

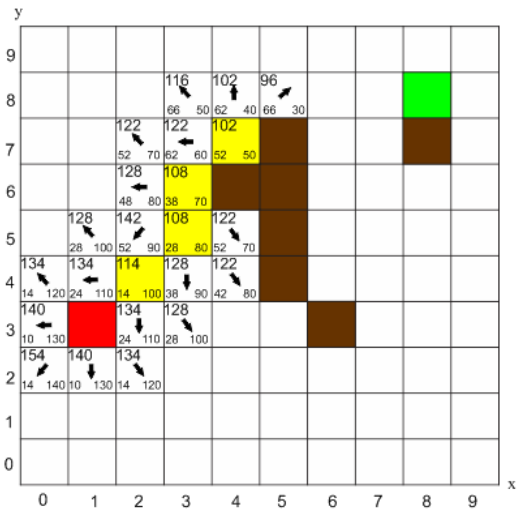
Node	Nilai G	Nilai H	Nilai F
2.5	52	90	142
2.6	48	80	128
2.7	52	70	122
3.7	48	60	108
4.5	52	70	122
4.6	52	50	102

Berikut ini penjelasan melalui tabel untuk pencarian jalur terpendek pada langkah keempat.

Tabel 3.11 Tabel Perhitungan langkah keempat

Open List	(2.5), (2.6), (2.7), (3.7), (4.5), (4.7)
Closed List	(1.3), (2.4), (3.5), (3.6)
Parent	(3.6)
Currrent node	(4.7)

Pada gambar 3.19 menunjukan perhitungan pada langkah kelima yang berada pada *node* dengan koordinat (4.7).



Gambar 3.19 Penjelasan Algoritma A* pada langkah kelima

Pada gambar 3.19 terlihat ada 4 *node* yang masuk ke dalam *open list* yaitu *node* dengan koordinat (3.7), (3.8), (4.8) dan (5.8) sedangkan untuk *node* (1.3), (2.4), (3.5), (3.6) dan (4.7) merupakan *node closed list*, sementara *node* (4.6), (5.6) dan (5,7) merupakan penghalang (*collider*). Dari langkah kedua *node* yang masuk ke dalam *open list* terdapat niali F yang paling rendah yaitu *node* (5.8) dengan nilai F = 96. Untuk hasil perhitungan dari langkah-langkah perhitungan A* yang kelima akan dijelaskan dengan tabel berikut ini.

Tabel 3.12 Tabel Hasil Perhitungan A* pada langkah kelima

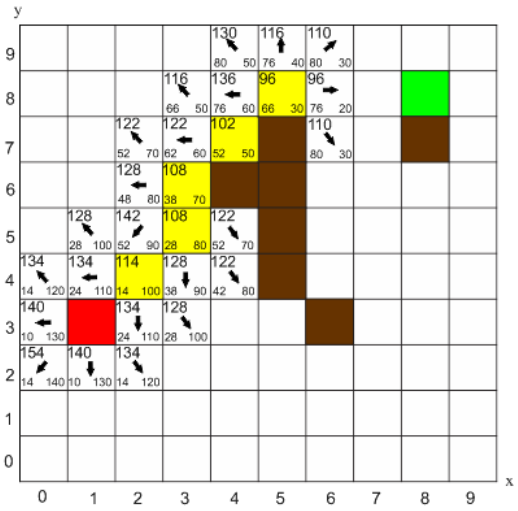
Node	Nilai G	Nilai H	Nilai F
3.7	62	60	122
3.8	66	50	116
4.8	62	40	102
5.8	66	30	96

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

Tabel 3.12 Tabel Perhitungan langkah kelima

Open List	(3.7), (3.8), (4.8), (5.8)
Closed List	(1.3), (2.4), (3.5), (3.6), (4.7)
Parent	(4.7)
Currrent node	(5.8)

Pada gambar 3.20 menunjukan perhitungan pada langkah keenam yang berada pada *node* dengan koordinat (5.8).



Gambar 3.20 Penjelasan Algoritma A* pada langkah keenam

Pada gambar 3.20 terlihat ada 6 *node* yang masuk ke dalam *open list* yaitu *node* dengan koordinat (4.8), (4.9), (5.9), (6.7), (6.8) dan (6.9) sedangkan untuk *node* (1.3), (2.4), (3.5), (3.6), (4.7) dan (5.8) merupakan *node closed list*, sementara *node* (5,7) merupakan penghalang (*collider*). Dari langkah kedua *node* yang masuk ke dalam *open list* terdapat niali F yang paling rendah yaitu *node* (6.8) dengan nilai F = 96. Untuk hasil perhitungan dari langkah-langkah perhitungan A* yang keenam akan dijelaskan dengan tabel berikut ini.

Tabel 3.14 Tabel Hasil Perhitungan A* pada langkah keenam

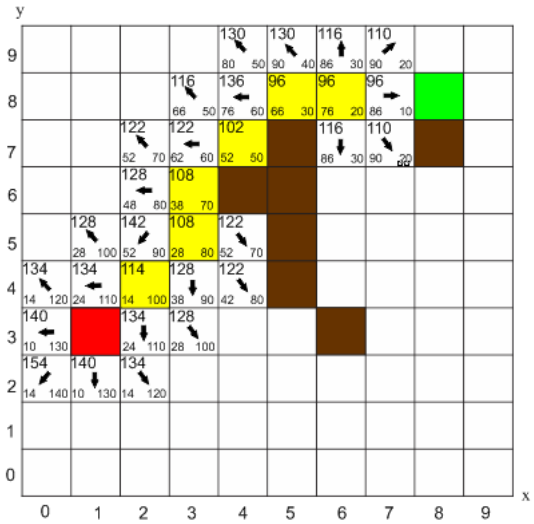
Node	Nilai G	Nilai H	Nilai F
4.8	76	60	136
4.9	80	50	130
5.9	76	40	116
6.7	80	30	110
6.8	76	20	96
6.9	80	30	110

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

Tabel 3.15 Tabel Perhitungan langkah keenam

Open List	(4.8), (4.9), (5.9), (6.7), (6.8), (6.9)
Closed List	(1.3), (2.4), (3.5), (3.6), (4.7), (5.8)
Parent	(5.8)
Currrent node	(6.8)

Pada gambar 3.21 menunjukan perhitungan pada langkah ketujuh yang berada pada *node* dengan koordinat (6.8).



Gambar 3.21 Penjelasan Algoritma A* pada langkah ketujuh

Pada gambar 3.21 terlihat ada 6 *node* yang masuk ke dalam *open list* yaitu *node* dengan koordinat (5.9), (6.7), (6.9), (7.7), (7.8) dan (7.9) sedangkan untuk *node* (1.3), (2.4), (3.5), (3.6), (4.7), (5.8) dan (6.8) merupakan *node closed list*, sementara *node* (5,7) merupakan penghalang (*collider*). Dari langkah kedua *node* yang masuk ke dalam *open list* terdapat niali F yang paling rendah yaitu *node* (7.8) dengan nilai F = 96. Untuk hasil perhitungan dari langkah-langkah perhitungan A* yang ketujuhakan dijelaskan dengan tabel berikut ini.

Tabel 3.16 Tabel Hasil Perhitungan A* pada langkah ketujuh

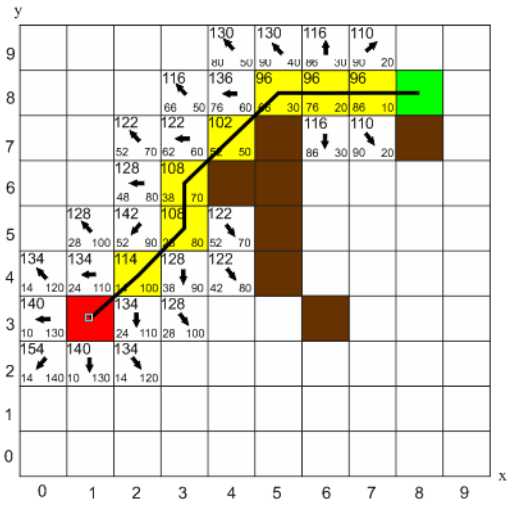
Node	Nilai G	Nilai H	Nilai F
5.9	90	40	130
6.7	86	30	116
6.9	86	30	116
7.7	90	20	110
7.8	86	10	96
7.9	90	20	110

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

Tabel 3.17 Tabel Perhitungan langkah keenam

Open List	(5.9), (6.7), (6.9), (7.7), (7.8), (7.9)
Closed List	(1.3), (2.4), (3.5), (3.6), (4.7), (5.8), (6.8)
Parent	(6.8)
Currrent node	(7.8)

Penentuan jalan terbaik dengan melakukan pemeriksaan dimulai dari target atau *node* akhir, dalam ilustrasi adalah *node* yang berwarna merah. Dengan mengikuti arah *pointer* yang berarti akan membawa kembali ke *node* awal, maka akan didapatkan jalan terbaik, maka jika diimplementasikan pada *game*, nantinya NPC akan melewati penghalang dan menemukan rute terpendek untuk sampai ke tujuan, untuk lebih jelasnya seperti gambar berikut.



Gambar 3.22 Penjelasan Algoritma A* pada langkah Terakhir

Berikut ini penjelasan melalui tabel untuk pencarian jalur terpendek pada langkah terakhir.

Tabel 3.18 Tabel Perhitungan langkah terakhir

Open List	(6.7), (6.9), (7.7), (7.9), (8.9)
Closed List	(1.3), (2.4), (3.5), (3.6), (4.7), (5.8), (6.8), (7.8)
Parent	(7.8)
Currrent node	(8.8)

3.2.4.1. *Pseudocode* Algoritma A*

Berikut adalah *pseudocode* Algoritma A* prosedur pencarian jalan dan prosedur pengecekan node terdekat

3.2.4.1.1. Prosedur pencarian jalan

Algoritma A* untuk pencarian jalan

{ Algoritma untuk pencarian jalan pada musuh}

Deklarasi

node : real

openlist : array

closedlist : array

node yang dicari : Boolean

Algoritma

```
while (node yang dicari = false)
      cek node sampai mendapatkan nilai F
        if node yang dicek tidak menghasilkan nilai F then
          fungsi getbackup dilakukan untuk mencari nilai F terkecil
        endif
          pindahkan node yang dicek ke dalam closedlist
          hapus node yang dicek dari openlist
             if node yang dicek merupakan node yang dicari then
               target yang dicari sudah ada
               fungsi tracebackpath dilakukan
            endif
            else
               for pengecekan node dilakukan lagi sebanyak jumlah node yang dekat
               sampai mendapatkan nilai dari node yang dicek
              endfor
endwhile
```

3.2.4.1.2. Prosedur pengecekan node terdekat

Algoritma A* untuk pengecekan node terdekat

{ Algoritma untuk pengecekan node terdekat}

Deklarasi

node : real

openlist : array

closedlist : array

node yang dicari : Boolean

nilai G : int

Algoritma

```
if node yang dicek dalam closedlist tidak ada then
  if node yang dicek dalam openlist ada then
    if posisi current node dengan node yang dicek berdekatan then
       nilai perpindahan secara serong = 14
    else
       nilai perpindahan secara lurus = 10
        hitung nilai G baru dari nilai G current node + nilai perpindahan
        ke node yang dicek
            if nilai G baru kurang dari nilai G dari node yang dicek  then
               ubah node yang dicek menjadi parent untuk dijadikan current node
               ubah nilai G dari nilai G baru dari node yang dicek
               hitung nilai F dari node yang dicek
             endif
     endif
  else
        ubah node yang dicek menjadi parent untuk dijadikan current node
        ubah nilai G dari nilai G current node + nilai perpindahan
        hitung nilai F dari node yang dicek
        masukan node yang dicek ke dalam openlist
   endif
endif
```

3.2.5. Analisis Algoritma *Fuzzy Logic*

Pada *game* ini *fuzzy logic* digunakan menentukan kondisi perilaku yang dilakukan oleh *Non Player Character* (NPC). Dengan adanya *fuzzy logic* masing-masing NPC dapat merubah perilaku sesuai variabel masukan. Metode *fuzzy* yang digunakan dalam *game* ini adalah metode sugeno. Analisis *fuzzy logic* tahap-tahapannya sebagai berikut ini.

3.2.5.1. *Variabel Fuzzy*

Didalam *game* ini digunakan 3 variabel dalam fungsi *fuzzy*, yaitu variabel Jarak dan darah sebagai variabel *input*, sementara variabel *output*nya yaitu variabel keputusan.

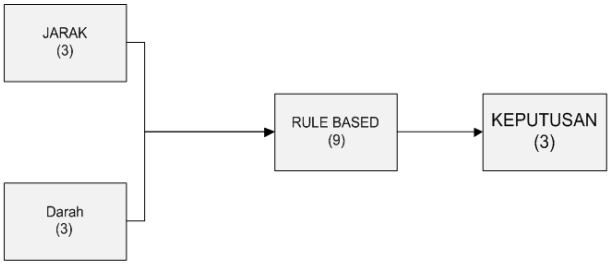
3.2.5.2. Nilai *Linguistik*

Dari tiga variabel yang digunakan, maka nilai *linguistik*ya sebagai berikut:

1. Variabel Jarak, dibagi menjadi 3 himpunan *fuzzy* yaitu: Dekat, Sedang dan Jauh.
2. Variabel Darah, dibagi menjadi 3 himpunan *fuzzy* yaitu: Sedikit, Sedang dan Banyak.
3. Variabel keputusan, dibagi menjadi 3 himpunan *fuzzy* yaitu: Menyerang, Menghindar dan Mengambil Darah.

3.2.5.3. *Fuzzyfikasi*

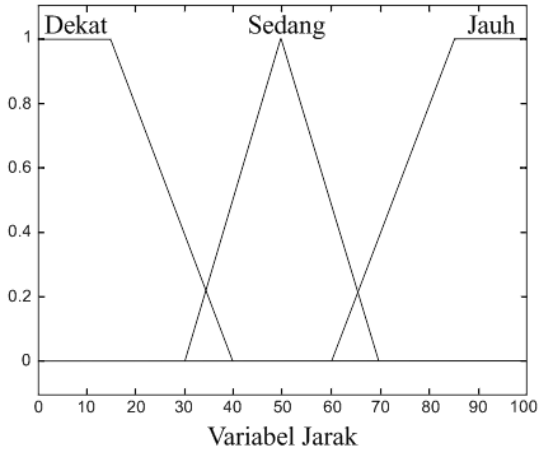
Fuzzyfikasi adalah proses memetakan nilai *crips* (numerik) ke dalam himpunan *fuzzy* dan menentukan derajat keanggotaanya. Secara garis besar pemetaan nilai *crips* ke dalam himpunan *fuzzy* dijelaskan dengan gambar berikut ini.



Gambar 3.23 *Fuzzyfikasi*

Berdasarkan *Fuzzy Interface system* diatas maka pemetaan himpunan *fuzzy* adalah sebagai berikut:

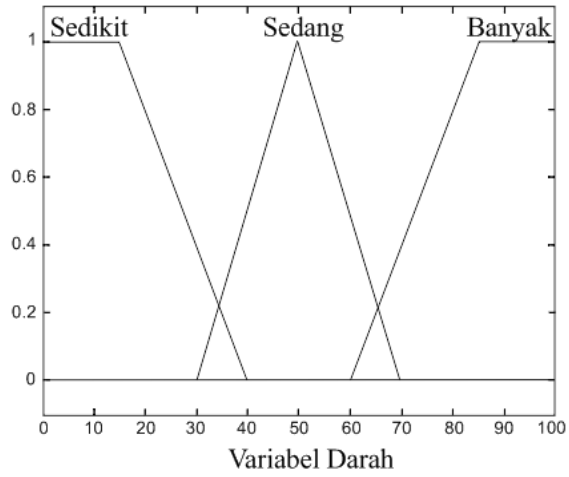
1. Variabel Jarak, terbagi menjadi 3 himpunan yaitu: Dekat, Sedang dan Jauh. *Range* nilai untuk variabel Jarak antara 0-100 akan dijelaskan sebagai berikut ini :
    1. Dekat = 0-40
    2. Sedang = 30-70
    3. Jauh = 60-100



Gambar 3.24 Grafik *Input* Variabel Jarak

Pada gambar 3.24 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 variabel Jarak ada 2 yaitu fungsi Trapesium dan Fungsi Segitiga.

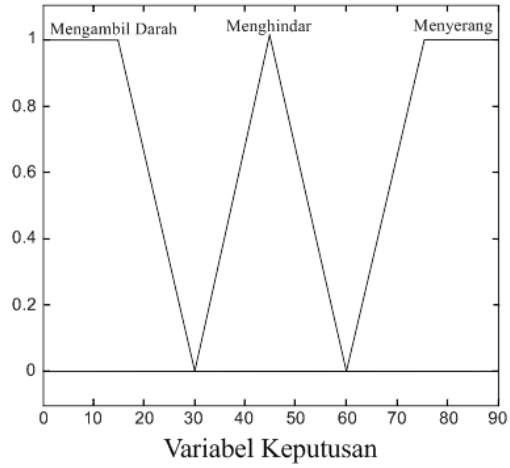
2. Variabel Darah, terbagi menjadi 3 himpunan yaitu: Sedikit, Sedang dan Banyak. *Range* nilai untuk variabel Darah antara 0-100 akan dijelaskan sebagai berikut ini :
    1. Sedikit = 0-40
    2. Sedang = 30-70
    3. Banyak = 60-100



Gambar 3.25 Grafik Input Variabel Darah

Pada gambar 3.25 menunjukan sebuah grafik darah yang mempunyai *range* nilai dari 0-100, setiap nilai *linguistik* dari variabel darah seperti sedikit, sedang dan banyak mempunyai nilai *fuzzyfikasi* yang berbeda-beda. Fungsi yang digunakan pada variabel darah ada 2 yaitu fungsi Trapesium dan fungsi Segitiga.

3. Variabel Keputusan, terbagi menjadi 3 himpunan yaitu: Menyerang, Menghindar dan Mengambil Darah. *Range* nilai untuk variabel Keputusan antara 0-90 akan dijelaskan sebagai berikut ini :
    1. Mengambil Darah = 0-30
    2. Menghindar = 30-60
    3. Menyerang = 60-90



Gambar 3.26 Grafik Variabel Keputusan

Pada gambar 3.26 Menunjukan sebuah grafik keputusan yang mempunyai *range* nilai dari 0 – 90, setiap nilai *lingustik* seperti mengambil darah, menghindar dan menyerang mempunyai nilai *fuzzyfikasi* yang berbeda-beda. Perhitungan nilai *fuzzyfikasi* didapatkan dari beberapa fungsi, fungsi yang digunakan pada variabel keputusan ada 2 yaitu fungsi Trapesium dan Fungsi Segitiga.

3.2.5.4. *Fuzzy Rules*

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

1. *If* (Jarak *is* Dekat) *and* (Darah is Sedikit) *then* (Keputusan NPC *is* Mengambil darah)
2. *If* (Jarak *is* Dekat) *and* (Darah is Sedang) *then* (Keputusan NPC *is* Menghindar)
3. *If* (Jarak *is* Dekat) *and* (Darah *is* Banyak) *then* (Keputusan NPC *is* Menyerang)
4. *If* (Jarak *is* Sedang) *and* (Darah *is* Sedikit) *then* (Keputusan NPC *is* Mengambil Darah)
5. *If* (Jarak *is* Sedang) *and* (Darah *is* Sedang) *then* (Keputusan NPC *is* Menghindar)
6. *If* (Jarak *is* Sedang) *and* (Darah *is* Banyak) *then* (Keputusan NPC *is* Menyerang)
7. *If* (Jarak *is* Jauh) *and* (Darah *is* Sedikit) *then* (Keputusan NPC *is* Menghindar)
8. *If* (Jarak *is* Jauh) *and* (Darah *is* Sedang) *then* (Keputusan NPC *is* Menyerang)
9. *If* (Jarak *is* Jauh) *and* (Darah *is* Banyak) *then* (Keputusan NPC *is* Menyerang)

3.2.5.5. 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*).

$$z^* = \frac{\sum \mu i.zi}{\sum \mu i} \tag{3.3}$$

z*= titik pusat nilai *crisp*

μi =derajat keanggotaan masing-masing variabel

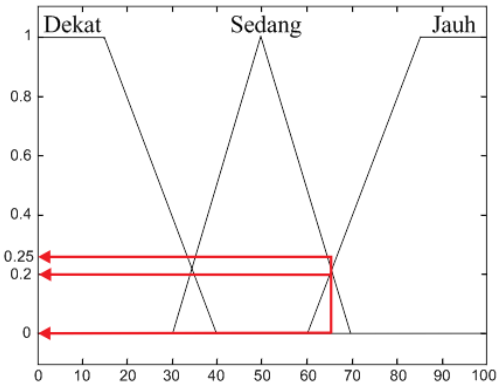
zi = nilai domain pada variabel *linguistik* z.

3.2.5.6. Contoh Kasus

Dalam contoh kasus ini akan dijelaskan langkah-langkah perhitungan *Fuzzy Logic* sesuai *input* nilai yang diberikan. Misalkan untuk variabel Jarak memiliki *input*an dengan nilai 65 sedangkan untuk variabel darah memilki *input*an dengan nilai 31. Untuk lebih jelasnya perhitungannya maka langkah-langkahnya sebagai berikut :

Ada 2 variabel yang akan dimodelkan, yaitu :

1. Variabel Jarak yang terdiri dari 3 himpunan yaitu Dekat, Sedang dan Jauh dengan *input*an yaitu 65 yang digambarkan sebagai berikut :



Gambar 3.27 Grafik Variabel Jarak dengan *Input*an 65

Pada gambar 3.27 terdapat nilai Keanggotaan yang berbeda disetiap himpunannya, berikut ini perhitungan untuk menentukan nilai keanggotaan dari masing-masing himpunannya :

a. Himpunan Dekat

$$\mu Dekat[x]\begin{cases}1; x \leq 15\\ \frac{40-x}{25}; 15 \leq x \leq 40\\ 0; x \geq 40\end{cases} \quad (3.4)$$

$$\mu Dekat[65] \quad = 65 \geq 40 = 0$$

Nilai keanggotaan dari Himpunan Dekat adalah 0 karena *range* nilai dari himpunan dekat adalah 0 sampai 40, maka 65 tidak termasuk keanggotaan dari himpunan Dekat.

b. Himpunan Sedang

$$\mu sedang[x]\begin{cases}0; x \leq 30 \; atau \; x \geq 70\\ \frac{x-30}{20}; 30 \leq x \leq 50\\ \frac{70-x}{20}; 50 \leq x \leq 70\end{cases} \quad (3.5)$$

$$\mu Sedang[65] = \frac{70-65}{20} = 0{,}25$$

Nilai keanggotaan dari Himpunan Sedang adalah 0.25 karena *range* nilai dari himpunan Sedang adalah 30 sampai 70, maka 65 termasuk keanggotaan dari himpunan Sedang yang mempunyai nilai yang tidak pasti.

c. Himpunan Jauh

$$\mu Jauh[x]\begin{cases}0; x \leq 60\\ \frac{x-60}{25}; 60 \leq x \leq 85\\ 1; x \geq 85\end{cases} \quad (3.6)$$

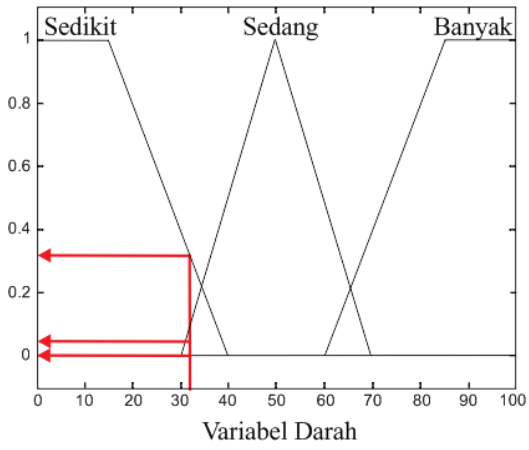
$$\mu Jauh[65] \quad = \frac{65-60}{25} = 0{,}2$$

Nilai keanggotaan dari Himpunan Jauh adalah 0,2 karena *range* nilai dari himpunan Sedang adalah 60 sampai 100, maka 65 termasuk keanggotaan dari himpunan Jauh yang mempunyai nilai yang tidak pasti.

Dari hasil perhitungan fungsi Keanggotaan yang memakai fungsi Trapesium dan fungsi Segitiga untuk mencari nilai keanggotaan dari variabel Jarak, maka didapatkan nilai derajat keanggotaan dari masing-masing himpunan yaitu :

1. Fungsi Keanggotaan Dekat dengan *input*an [65] = 0
2. Fungsi Keanggotaan Sedang dengan *input*an [65] = 0,25
3. Fungsi Keanggotaan Jauh dengan *input*an [65] = 0,2

2. Variabel Darah yang terdiri dari 3 himpunan yaitu Sedikit, Sedang dan Banyak dengan inputan yaitu 31 yang digambarkan sebagai berikut :



Gambar 3.28 Grafik Variabel Darah dengan *Input*an 31

Pada gambar 3.28 terdapat Nilai Keanggotaan yang berbeda disetiap himpunannya, berikut ini perhitungan untuk menentukan nilai keanggotaan dari masing-masing himpunannya:

a. Himpunan Sedikit

$$\mu Sedikit[x]\begin{cases} 1; x \leq 15 \\ \frac{40-x}{25}; 15 \leq x \leq 40 \\ 0; x \geq 40 \end{cases} \quad (3.7)$$

$$\mu Sedikit[31] = \frac{40-31}{25} = 0{,}36$$

Nilai keanggotaan dari Himpunan Sedikit adalah 0,36 karena *range* nilai dari himpunan dekat adalah 0 sampai 40, maka 31 termasuk keanggotaan dari himpunan Sedikit yang mempunyai nilai yang tidak pasti.

b. Himpunan Sedang

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

$$\mu Sedang[31] = \frac{31-30}{20} = 0{,}05$$

Nilai keanggotaan dari Himpunan Sedang adalah 0,05 karena *range* nilai dari himpunan Sedang adalah 30 sampai 70, maka 31 termasuk keanggotaan dari himpunan Sedikit yang mempunyai nilai yang tidak pasti.

c. Himpunan Banyak

$$\mu Banyak[x]\begin{cases} 0; x \leq 60 \\ \frac{x-60}{25}; 60 \leq x \leq 85 \\ 1; x \geq 85 \end{cases} \quad (3.8)$$

$$\mu Banyak[31] = 31 \leq 60 = 0$$

Nilai keanggotaan dari Himpunan Banyak adalah 0 karena *range* nilai dari himpunan Sedang adalah 60 sampai 100, maka 31 tidak termasuk keanggotaan dari himpunan banyak.

Dari hasil perhitungan fungsi Keanggotaan yang memakai fungsi Trapesium untuk mencari niali keanggotaan dari variabel Darah, maka didapatkan nilai derajat keanggotaan dari masing-masing himpunan yaitu :

a. Fungsi Keanggotaan Sedikit dengan *input*an [31] = 0,36

b. Fungsi Keanggotaan Sedang dengan *input*an [31] = 0,05

c. Fungsi Keanggotaan Banyak dengan *input*an [31] = 0

Setelah mendapatkan nilai derajat keanggotaan dari masing-masing variabel, maka nilai derajat keanggotaan tersebut dibandingkan sesuai dengan *Rule*. Dengan mencari nilai minimum untuk setiap *rule* yang dihitung dan mencari nilai z dari masing-masing *rule*. Berikut ini perhitungan mencari nilai minimum dari masing masing nilai derajat yang dibandingkan sesuai dengan *rule* dan mencari nilai z dari masing-masing *rule.*

1. *Rule* 1

   If (Jarak *is* Dekat) *and* (Darah is Sedikit) *then* (Keputusan NPC *is* Mengambil darah)

α-predikat1 = µjrkDekat And µdrhSedikit (3.9)

= min(µjrkDekat[65], µdrhSedang[31])

= min (0;0,36)

= 0

Nilai z1 = 30 – (0)*(30-0)

= 30

2. *Rule* 2

   If (Jarak *is* Dekat) *and* (Darah is Sedang) *then* (Keputusan NPC *is* Menghindar)

   α-predikat2 = µjrkDekat And µdrhSedang (3.10)

   = min(µjrkDekat[65], µdrhSedang[31])

   = min (0;0,05)

   = 0

   Nilai z2 = 30 – (0)*(30-0)

   = 30

3. *Rule* 3

   If (Jarak *is* Dekat) *and* (Darah *is* Banyak) *then* (keputusan NPC *is* Menyerang)

   α-predikat3 = µjrkDekat And µdrhBanyak (3.11)

   = min(µjrkDekat[65], µdrhBanyak[31])

   = min (0;0)

   = 0

   Nilai z3 = 60 + (0)*(90-60)

   = 60

4. *Rule* 4

   If (Jarak *is* Sedang) *and* (Darah *is* Sedikit) *then* (Keputusan NPC *is* Mengambil Darah)

   α-predikat4 = µjrkSedang And µdrhSedikit (3.12)

   = min(µjrkSedang[65], µdrhSedikit[31])

   = min (0,25;0,36)

   = 0,25

   Nilai z4 = 30 – (0,25)*(30-0)

   = 22,5

5. *Rule* 5

   If (Jarak *is* Sedang) *and* (Darah *is* Sedang) *then* (Keputusan NPC *is* Menghindar)

   α-predikat5 = µjrkSedang And µdrhSedang (3.13)

   = min(µjrkSedang[65], µdrhSedang[31])

   = min (0,25;0,05)

   = 0,05

   Nilai z5 = 30 – (0,05)*(60-30)

   = 28,5

6. *Rule* 6

   If (Jarak *is* Sedang) *and* (Darah *is* Banyak) *then* (keputusan NPC *is* Menyerang)

   α-predikat6 = µjrkSedang And µdrhBanyak (3.14)

   = min(µjrkSedang[65], µdrhBanyak[31])

   = min (0,25;0)

   = 0

   Nilai z6 = 60 + (0)*(90-60)

   = 60

7. *Rule* 7

   If (Jarak *is* Jauh) *and* (Darah *is* Sedikit) *then* (Keputusan NPC *is* Menghindar)

   α-predikat7 = µjrkJauh And µdrhSedikit (3.15)

   = min(µjrkJauh[65], µdrhSedikit[31])

   = min (0,2;0,36)

   = 0,2

   Nilai z7 = 30 - (0,2)*(60-30)

   = 24

8. *Rule* 8

   If (Jarak *is* Jauh) *and* (Darah *is* Sedang) *then* (Keputusan NPC *is* Menyerang)

   α-predikat8 = µjrkJauh And µdrhSedang (3.16)

   = min(µjrkJauh[65], µdrhSedang[31])

   = min (0,2;0,05)

   = 0,05

   Nilai z8 = 30 + (0,05)*(90-60)

   = 61,5

9. *Rule* 9

   If (Jarak *is* Jauh) *and* (Darah *is* Banyak) *then* (Perilaku NPC *is* Menyerang)

   α-predikat9 = µjrkJauh And µdrhBanyak (3.17)

   = min(µjrkSedang[65], µdrhBanyak[31])

   = min (0,2;0)

   = 0

   Nilai z9 = 60 + (0,02)*(90-60)

   = 60

Setelah mendapatkan nilai dari setiap *rule* maka langkah selanjutnya adalah menentukan *Defuzzyfikasi* yaitu proses pemetaan dari hinpunan *fuzzy set* yang menghasilakan nilai *crips* dengan menggunakan metode sugeno yang akan menghasilkan suatu nilai *crips* untuk menentukan variabel keputusan.

$$z^* = \frac{\sum \mu i.zi}{\sum \mu i} \quad (3.18)$$

$$z = \frac{0 \times 30 + 0 \times 30 + 0 \times 60 + 0{,}25 \times 22{,}5 + 0{,}05 \times 28{,}5 + 0 \times 60 + 0{,}2 \times 24 + 0{,}05 \times 61{,}5 + 0 \times 60}{0 + 0 + 0 + 0{,}25 + 0{,}05 + 0 + 0{,}2 + 0{,}05 + 0}$$

$$= \frac{14{,}9}{0{,}55}$$

$$= \mathbf{27{,}09}$$

Jumlah yang dihasilkan oleh *Defuzzyfikasi* adalah 27,09 yang termasuk keanggotaan nilai himpunan Mengambil Darah dari variabel Keputusan karena mempunyai *range* dari 0-30. Sesuai dengan contoh kasus yang dihasilkan dari nilai inputan variabel Jarak adalah 65 dan inputan variabel Darah 31 maka menghasilkan nilai z* yaitu **27,09** jadi keputusan yang diambil oleh NPC adalah **Mengambil Darah.**

3.2.5.7. *Pseudocode Fuzzy Logic*

Berikut adalah pseudocode Algoritma *Fuzzy Logic* untuk menentukan nilai variabel jarak, variabel darah dan hasil *defuzzyfikasi*.

3.2.5.7.1. *Pseudocode* untuk menentukan variabel jarak

```
Deklarasi
nilai jarak : float
jarak dekat : boolean
jarak sedang : boolean
jarak jauh : boolean
Algoritma
if nilai jarak < 40 then
  jarak dekat
endif
 if nilai jarak > 30 dan < 70 then
   jarak sedang
 endif
   if nilai jarak > 60 dan < 100 then
    jarak jauh
   endif
```

3.2.5.7.2. *Pseudocode* untuk menentukan variabel darah

```
Deklarasi
nilai darah: float
darah sedikit : boolean
darah sedang : boolean
```

```
darah banyak : Boolean
Algoritma
if nilai darah < 40 then
  darah sedikit
endif
 if nilai darah > 30 dan < 70 then
   darah sedang
 endif
   if nilai darah > 60 dan < 100 then
     darah banyak
   endif
```

3.2.5.7.3. *Pseudocode* menentukan hasil *defuzzyfikasi*

```
Deklarasi
hitpoint : float
jarak : float
damage : int
posisi player : real
posisi tempat darah : real
posisi musuh : real
animasi tembak : string
animasi jalan : string
animasi lari : string
animasi diam : string
shootrange : float
Algoritma
if target
 jarak didapat dari posisi player – posisi musuh
 rule baru didapat dari jarak dan hitpoint
 periksa semua rule
  set rule 1 dari rule 1
  set rule 2 dari rule 2
```

```
 set rule 3 dari rule 3
 set rule 4 dari rule 4
 set rule 5 dari rule 5
 set rule 6 dari rule 6
 set rule 7 dari rule 7
 set rule 8 dari rule 8
 set rule 9 dari rule 9
menghasilkan hasil defuzzyfikasi
 if posisi tempat darah – posisi musuh then
      mainkan animasi diam pada musuh dan tambahkan darah musuh
endif
   if hasil defuzzyfikasi < 30 then
        print out “mangambil darah”
          posisi musuh – posisi tempat darah
           mainkan animasi lari untuk musuh sampai pada posisi tempat darah
             if animasi lari mulai then
               mainkan suara berjalan
             endif
    endif
    else
        if hasil defuzzyfikasi > 30 dan < 60 then
            print out “ menghindar”
              posisi player – posisi musuh
             if jarak < range tembak then
                mainkan animasi tembak pada musuh dengan jarak yang tidak dekat
                        if waktu > waktu tembak then
                            tembakan musuh akan delay
                        endif
              endif
         endif
         else
```

```
if hasil defuzzyfikasi > 60 dan < 90 then
    print out " menyerang"
      posisi player – posisi musuh
       if jarak < range menyarang then
         set animasi jalan pada musuh
           if animasi jalan mulai then
             memainkan suara berjalan
           endif
       endif
       if jarak < range tembak then
         set animasi tembak pada musuh
              if waktu > waktu tembak then
                tembakan musuh akan delay
              endif
        endif
    endif
endif
```

3.2.6. Analisis Kebutuhan *Non*-Fungsional

Analisi *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. Analisi 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 analisi kebutuhan sistem *non*-fungsional ini dijelaskan analisis kebutuhan perangkat lunak, analis kebutuhan perangkat keras, dan analisi pengguna

3.2.6.1. Analisis Kebutuhan Perangkat Lunak

Game Selamatkan Orangutan memiliki kebutuhan perangkat lunak sebagai berikut :

1. Sistem Oprasi *Windows* 7.
2. *Unity* 4.0.0.
3. *Blender* 2.64.
4. *Mono Develop* 2.8.2.
5. *Autodesk* 3ds Max 2011.
6. *Adobe Photoshop* CS6.

Perangkat lunak yang dibutuhkan pengguna untuk menjalankan *game* Selamatkan orangutan ini adalah sebagai berikut :

1. Sistem Oprasi *Windows* 7.
2. *DirectX* 9.

3.2.6.2. 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* Selamatkan orangutan 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* 512 MB.
5. Monitor dengan *resolusi* 1024 x 768 *pixel.*
6. *Speaker.*
7. *Keyboard* dan *mouse.*

Spesifikasi minimum perangkat keras bagi pengguna untuk memainkan *game* Selamatkan Orangutan 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 *pixel.*
6. *Speaker.*
7. *Keyboard* dan *mouse*.

3.2.6.3. Analisis Pengguna

Pada analisis pengguna ini akan mencakup analisis beberapa parameter terhadap calon pengguna *game*.

Game yang akan dibangun ini dapat dimainkan oleh berbagai kalangan akan tetapi karena *game* ini mengandung unsur kekerasan, pertimbangan tersebut membuat *game* ini ditunjukan untuk usia 13 tahun ke atas karena menurut ESRB (*Entering software Ratting Board)* pada usia 13 tahun ke atas mungkin cocok untuk *game* yang mengandung unsur kekerasan, sedikit mengeluarkan darah, dan tidak jarang menggunakan bahasa yang kuat [5]. Dari urain diatas sangat memungkinkan bahwa *game* selamatkan orangutan ini ditunjukan untuk usia 13 tahun ke atas karena *game* selamatkan orangutan mengandung unsur kekerasan, sedikit mengeluarkan darah, dan menggunakan bahasa yang kuat.

1. *User knowledge and Experience* dari target pengguna yang akan menggunakan *game.*

   Game Selamatkan orangutan ini bisa digunakan oleh kalangan apapun, tetapi pengetahuan dan pengalaman akan memudahkan pengguna dalam penggunaanya. Terutama pengetahuan dan pengalaman dalam

memainkan *game*. Berikut ini kelasifikasi *Knowledge and experience* dari pengguna *game* :

Tabel 3.19 Analisi Klasifikasi *Knowledge ana experience*

Education Level	***Reading Level***	***Typing Skills***
Bisa digunakan oleh berbagai kalangan, seperti pelajar, mahasiswa hingga masyarakat awam	Bisa digunakan oleh berbabgai *level* pendidikan dengan reading *level* yang sedang	Tidak memerlukan *skills* atau kemampuan yang tinggi
Computer Literacy	***Task Experience***	***System Experience***
Game ini bisa digunakan oleh pengguna yang mememilki kemampuan komputer yang *moderate* (menengah)	*Game* ini bisa digunakan oleh dengan pengelaman penggunaan komputer dan *game* yang sedang	Bisa digunakan oleh dengan pengalaman pengguna komputer dana *game* sedang
Appliacation Experience	***Native Language***	***Use Of Other System***
Bisa digunakan dalam semua sistem operasi *Windows* xp/vista/7	Menggunakan satu bahasa, yaitu Bahasa Indonesia	Bisa dijalankan tanpa menginstal aplikasi lain

2. *User physical Characteristic*

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

Tabel 3.20 Analisis *User Physical Characteristic*

Age	13 tahun ke atas
Gender	Pria dan Wanita
Handednes	Kanan dan Kiri
Color Blind	Pengguna yang tidak bisa membedakan warna yang satu dengan yang lainnya (buta warna) masih mampu menggunakan aplikasi ini, karena tidak ada indikator 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

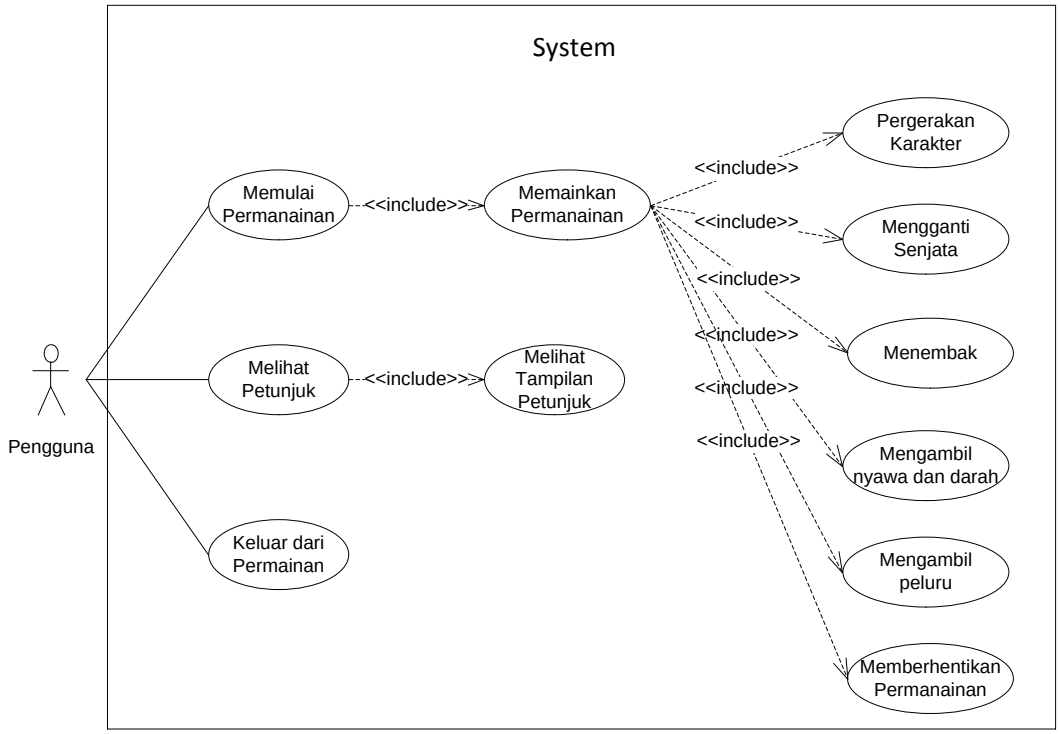
3.2.7. Analisis Kebutuhan Fungsional

Analsis 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.

Analisi kebutuhan fungsional dimodelkan dengan menggunakan UML (*Unified Modeling language*), dimana tahap-tahap analisis yang dilakukan dalam membangun *game* Selamatkan Orangutan ini antara lain *Use Case Diagram, Activity Diagram, Class Diagram, Sequence Diagram.*

3.2.7.1. *Use Case Diagram*

Use Case Diagram merupakan konstruksi untuk mendeskripsikan hubungan-hubungan yang terjadi antara *actor* dengan aktivitas yang terdapat pada sistem. Sasaran pemodelan *use case* diantaranya adalah mendefinisikan kebutuhan fungsional dan oprasional sistem dengan mendefinisikan skenario penggunaan sistem yang akan dibangun. Dari hasil analisis aplikasi yang ada maka *use case* diagram untuk *game* Selamatkan Orangutan dapat dilihat pada gambar 3.29.



Gambar 3.29 *Use Case Diagram*

3.2.7.2. Definisi *Actor*

Definis *Actor* berfungsi untuk menjelaskan *actor* yang terdapat pada *use case diagram*. Definisi *actor* diterangkan pada tabel 3.21 Berikut ini.

Tabel 3.21 Definisi *Actor*

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

3.2.7.3. Definisi *Use Case*

Definisi *use case* berfungsi untuk menjelaskan fungsi *use case* yang terdapat pada *use case diagram*. Definisi *use case* diterapkan pada tabel 3.22 Berikut.

Tabel 3.22 Definisi *Use Case*

No.	*Use case*	Deskripsi
1.	Memulai Permainan	**Kode *Use Case* :** UC-01 **Deskripsi Singkat :** Proses untuk memulai permainan **Aktor :** Pengguna ***Trigger* :** Pengguna memilih *menu* Main **Prekondisi :** Halaman *menu* utama telah ditampilkan, Pengguna memilih *menu* Main **Pascakondisi :** sistem menampilkan *scene* misi *level* 1
2.	Melihat petunjuk	**Kode *Use Case* :** UC-02 **Deskripsi Singkat :** Proses untuk menampilkan petunjuk atau panduan bermain

		Aktor : Pengguna ***Trigger* :** Pengguna memilih *menu* Petunjuk **Prekondisi :** Halaman *menu* utama telah ditampilkan, Pengguna memilih *menu* Petunjuk **Pascakondisi :** sistem menampilkan halaman Petunjuk atau panduan bermain
3.	Keluar dari permainan	**Kode *Use Case* :** UC-03 **Deskripsi Singkat :** Proses keluar dari permainan **Aktor :** Pengguna ***Trigger* :** Pengguna memilih *menu* Keluar **Prekondisi :** Halaman *menu* utama telah ditampilkan, Pengguna memilih *menu* keluar **Pascakondisi :** sistem akan keluar dari permainan
4.	Memainkan Permainan	**Kode *Use Case* :** UC-04 **Deskripsi Singkat :** Proses permainan sedang berjalan. **Aktor :** Pengguna ***Trigger* :** Pengguna memilih *menu* Main pada menu utama, lalu menuju ke *scene* misi, pengguna memilih *menu* Mulai pada menu yang terdapat pada *scene* misi . **Prekondisi :** Halaman *menu* utama telah

		ditampilkan, Pengguna memilih *menu* main **Pascakondisi :** sistem akan menjalankan permainan
5.	Melihat Tampilan petunjuk	**Kode *Use Case* :** UC-05 **Deskripsi Singkat :** Proses melihat tampilan petunjuk atau panduan bermain **Aktor :** Pengguna ***Trigger* :** Pengguna memilih *menu* Petunjuk **Prekondisi :** Halaman *menu* utama telah ditampilkan, Pengguna memilih *menu* Petunjuk **Pascakondisi :** sistem menampilkan halaman Petunjuk atau panduan bermain
6.	Pergerakan Karakter	**Kode *Use Case* :** UC-06 **Deskripsi Singkat :** Proses untuk menggerakan karakter dengan animasi **Aktor :** Pengguna ***Trigger* :** Pengguna menggerakan karakter dengan interaksi menggunakan *keyboard* dengan menekan tombol W, S, A, D, Shift dan X **Prekondisi :** Pengguna memainkan permainan **Pascakondisi :** sistem akan menjalankan pergerakan karakter sesuai dengan *input* yang diberikan

7.	Mengganti Senjata	**Kode *Use Case* :** UC-07 **Deskripsi Singkat :** Proses untuk mengganti senjata karakter **Aktor :** Pengguna ***Trigger* :** Pengguna menggerakan karakter dengan interaksi menggunakan *keyboard* dengan menekan tombol 1 dan 2 **Prekondisi :** Pengguna memainkan permainan **Pascakondisi :** sistem akan menjalankan pergantian senjata pada karakter sesuai dengan input yang diberikan
8.	Menembak	**Kode *Use Case* :** UC-08 **Deskripsi Singkat :** Proses untuk menembak senjata karakter **Aktor :** Pengguna ***Trigger* :** Pengguna menggerakan karakter dengan interaksi menggunakan tombol kiri pada *mouse* atau pada *keyboard* dengan menekan tombol Ctrl **Prekondisi :** Pengguna memainkan permainan **Pascakondisi :** sistem akan menjalankan pergerakan karakter dengan animasi menembak sesuai dengan *input* yang diberikan
9.	Mengambil Nyawa dan Darah	**Kode *Use Case* :** UC-09 **Deskripsi Singkat :** Proses untuk mengambil nyawa dan darah

		Aktor : Pengguna ***Trigger* :** Pengguna menggerakan karakter munuju *item* box nyawa atau box darah. **Prekondisi :** Pengguna memainkan permainan **Pascakondisi :** sistem akan menambahkan nyawa dan darah pada karakter
10.	Mengambil Peluru	**Kode *Use Case* :** UC-10 **Deskripsi Singkat :** Proses untuk mengambil Peluru **Aktor :** Pengguna ***Trigger* :** Pengguna menggerakan karakter munuju *item* box peluru. **Prekondisi :** Pengguna memainkan permainan **Pascakondisi :** sistem akan menambahkan peluru pada karakter
11.	Memberhentikan Permainan	**Kode *Use Case* :** UC-11 **Deskripsi Singkat :** Proses untuk memberhentikan permainan **Aktor :** Pengguna ***Trigger* :** Pengguna menekan tombol yang ada pada *keyboard* dengan menekan tombol Esc **Prekondisi :** Pengguna memainkan permainan **Pascakondisi :** sistem akan memberhentikan permainan

3.2.7.4. *Use Case* Skenario dan *Activity Diagram*

Skenario setiap bagian pada *Use Case* menunjukan 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*.

Activity Diagram gambaran berbagai aliran aktifitas dalam sistem yang sedang dirancang, bagaimana masing-masing alir berawal, *decision* yang mungkin terjadi dan bagaimana berakhir. *Activity diagram* juga dapat menggambarkan proses paralel yang mungkin terjadi pada beberapa eksekusi. *Activity diagram* merupakan *state diagram* khusus, di mana sebagian besar *state* adalah *action* dan sebagian besar transisi di-*trigger* oleh selesainya *state* sebelumnya (*internal processing*). Oleh karena itu *activity diagram* tidak menggambarkan *behaviour* internal sebuah sistem dan interaksi antar subsistem secara eksak, tetapi lebih menggambarkan proses-proses dan jalur-jalur aktifitas dari *level* atas secara umum. Sebuah aktivitas dapat direalisasikan oleh satu *Use Case* atau lebih. Aktifitas menggambarkan proses yang berjalan, sementara *Use Case* menggambarkan bagaimana aktor menggunakan sistem untuk melakukan aktifitas.

3.2.7.4.1. *Use Case* Skenario Memulai Permainan

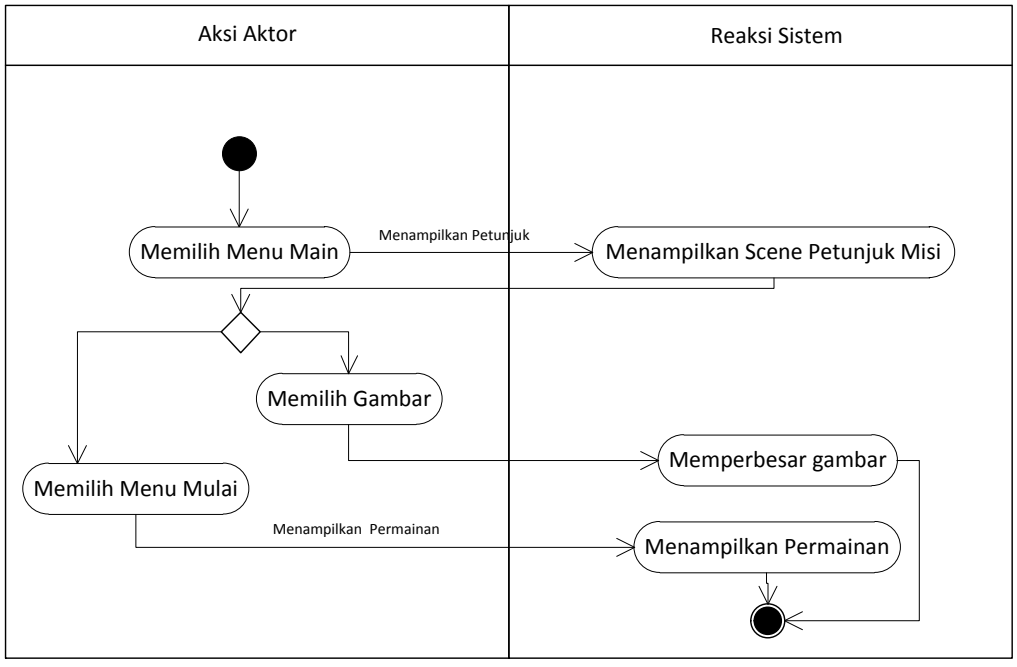
Nama *Use Case* : Memulai Permainan (UC-01)

Tabel 3.23 Skenario *Use Case* Memulai Permainan

Aksi Aktor	Reaksi Sitem
Skenario Normal : Memulai Permainan SC-01-01	
1.Memilih *Menu* Main	
	2.Menampilkan *Scene* Petunjuk Misi *Level*
3.Memilih Gambar	

	4.Memperbesar Gambar
Skenario Alternatif 1 : Memulai Permainan SC-01-02	
1.Memilih *Menu* Main	
	2.Menampilkan *Scene* Petunjuk Misi *Level*
5.Memilih *Menu* Mulai	
	6.Menampilkan Permainan

3.2.7.4.1.1. *Aktivity Diagram Use Case* Skenario Memulai Permainan



Gambar 3.30 *Activity Diagram* Memulai Permainan

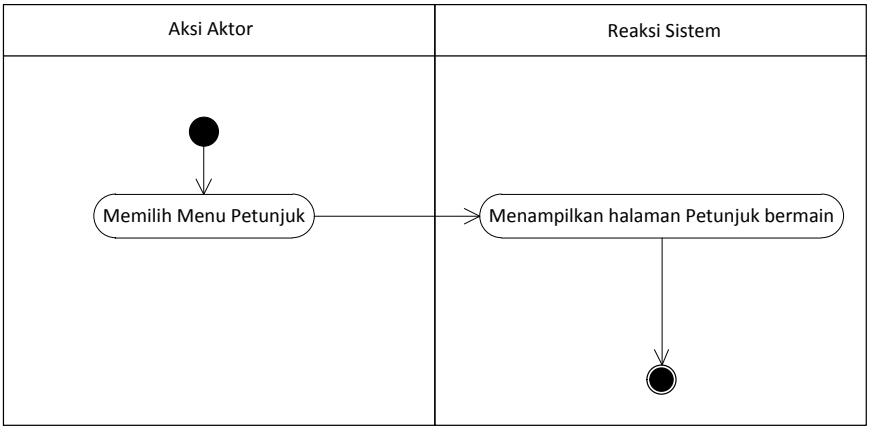
3.2.7.4.2. *Use Case* Skenario Melihat Petunjuk

Nama *Use Case* : Melihat Petunjuk (UC-02)

Tabel 3.24 Skenario *Use Case* Melihat Petunjuk

Aksi Aktor	Reaksi Sitem
Skenario Normal : Melihat Petunjuk SC-02-01	
1.Memilih *Menu* Petunjuk	
	2.Menampilkan Halaman Petunjuk atau panduan bermain

3.2.7.4.2.1. *Aktivity Diagram Use Case* Skenario Melihat Petunjuk



Gambar 3.31 *Activity Diagram* Melihat Petunjuk

3.2.7.4.3. *Use Case* Skenario Keluar dari Permainan

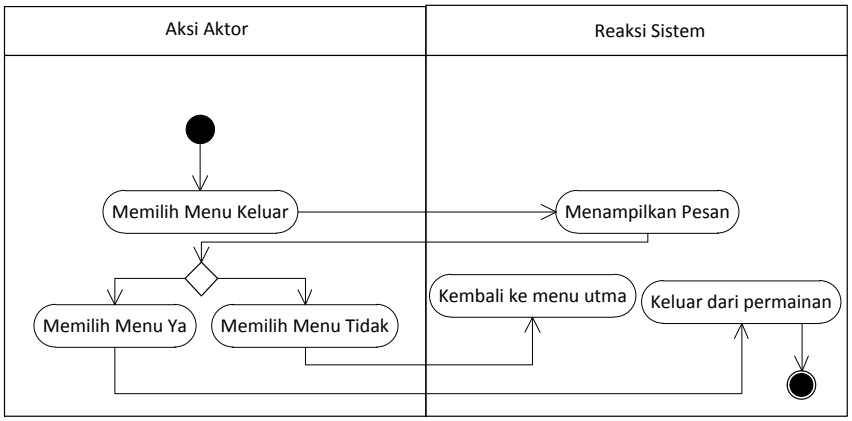
Nama *Use Case* : Keluar dari Permainan (UC-03)

Tabel 3.25 Skenario *Use Case* keluar dari Permainan

Aksi Aktor	Reaksi Sitem
Skenario Normal : Keluar dari Permainan SC-03-01	

1.Memilih *Menu* Keluar	
	2.Menampilkan Pesan keluar
3.Memilih *Menu* Ya	
	4.Keluar dari permianan
Skenario Normal 1: Keluar dari Permainan SC-03-02	
1.Memilih *Menu* Keluar	
	2.Menampilkan Pesan keluar
3.Memilih *Menu* Tidak	
	4.Menampilkan Menu utama

3.2.7.4.3.1. *Aktivity Diagram Use Case* Skenario Keluar dari Permainan



Gambar 3.32 *Activity Diagram* Keluar dari Permainan

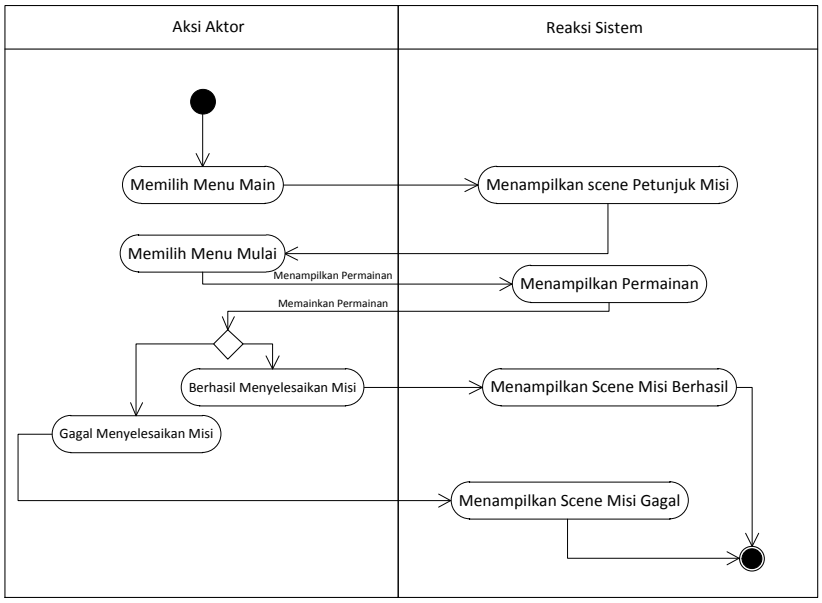
3.2.7.4.4. *Use Case* Skenario Memainkan Permainan

Nama *Use Case* : Memainkan Permainan (UC-04)

Tabel 3.26 Skenario *Use Case* Memainkan Permainan

Aksi Aktor	Reaksi Sitem
Skenario Normal : Memainkan Permainan Berhasil SC-04-01	
1.Memilih *Menu* Main	
	2.Menampilkan *Scene* Petunjuk Misi
3.Memilih *Menu* Mulai	
	4.Menampilkan Permainan
5.Berhasil Menyelesaikan Misi	
	6.Menampilkan *Scene* Misi Berhasil
Skenario Normal 1 : Memainkan Permainan Gagal SC-04-02	
1.Memilih *Menu* Main	
	2.Menampilkan *Scene* Petunjuk Misi
3.Memilih *Menu* Mulai	
	4.Menampilkan Permainan
5.Gagal Menyelesaikan Misi	
	6.Menampilkan *Scene* Misi Gagal

3.2.7.4.4.1. *Aktivity Diagram Use Case* Skenario memainkan Permainan



Gambar 3.33 *Activity Diagram* Memainkan Permainan

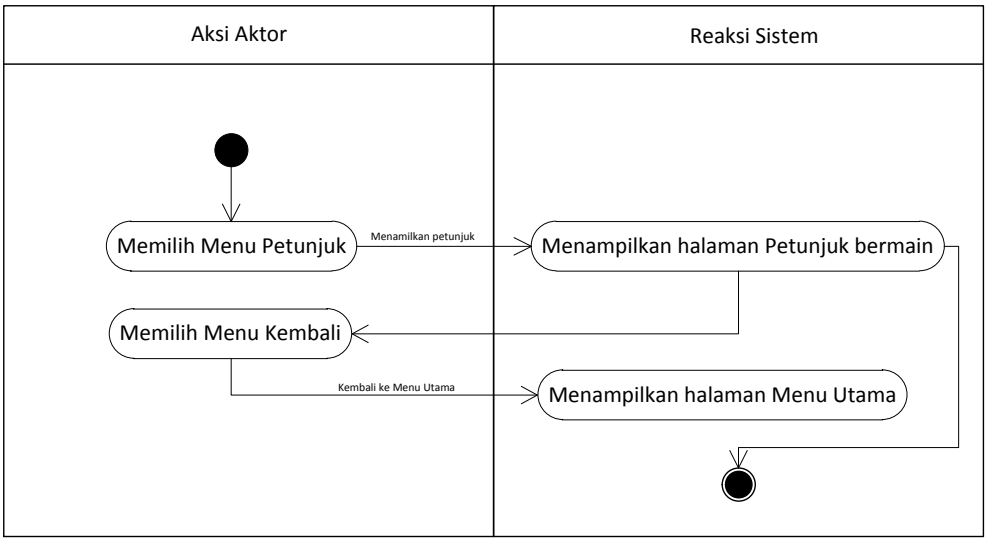
3.2.7.4.5. *Use Case* Skenario Melihat Tampilan Petunjuk

Nama *Use Case* : Melihat Tampilan Petunjuk (UC-05)

Tabel 3.27 Skenario *Use Case* Melihat Tampilan Petunjuk

Aksi Aktor	Reaksi Sitem
Skenario Normal : Melihat Tampilan Petunjuk SC-05-01	
1.Memilih *Menu* Petunjuk	
	2.Menampilkan Halaman Petunjuk bermain
3.Memilih *Menu* Kembali	
	4.Menampilkan Halaman Menu Utama

3.2.7.4.5.1. *Aktivity Diagram Use Case* Skenario Melihat Tampilan Petunjuk



Gambar 3.34 *Activity Diagram* Melihat Tampilan Petunjuk

3.2.7.4.6. *Use Case* Skenario Pergerakan Karakter

Nama *Use Case* : Pergerakan Karakter (UC-06)

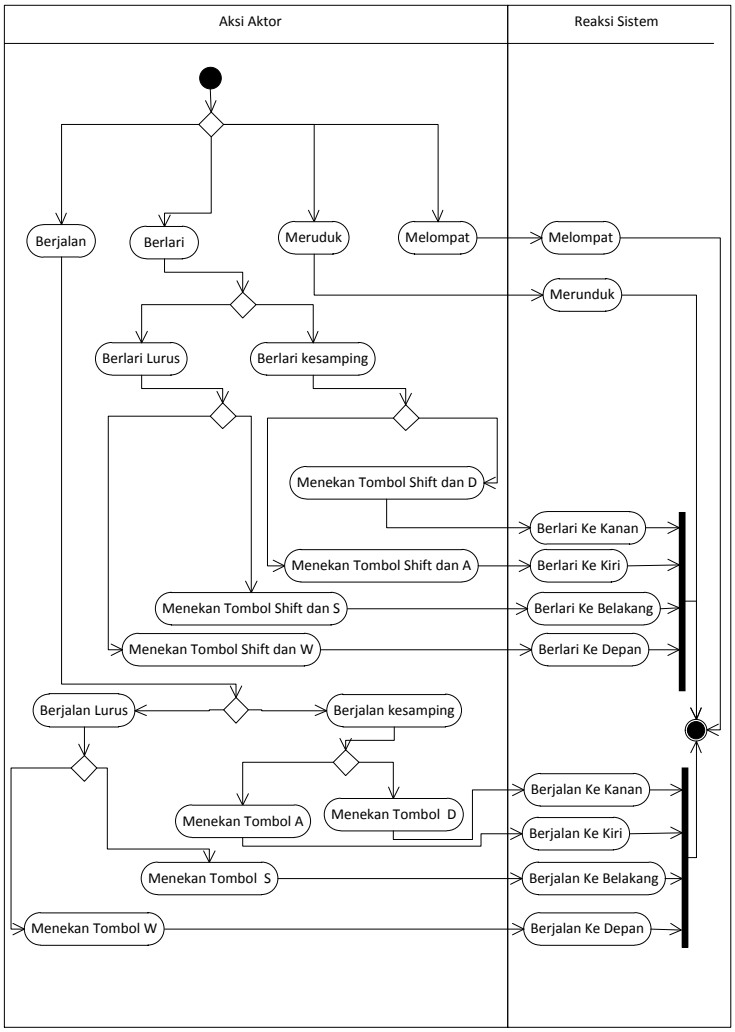
Tabel 3.28 Skenario *Use Case* Pergerakan Karakter

Aksi Aktor	Reaksi Sitem
Skenario Normal : Pergerakan Karakter berjalan SC-06-01	
Skenario Alternatif 1 : Pergerakan Karakter berjalan Lurus SC-06-02	
Skenario Alternatif 2: Pergerakan Karakter berjalan Lurus ke depan SC-06-03	
1. Menekan tombol W	
	2.Menggerakan Karakter berjalan ke depan
Skenario Alternatif 3: Pergerakan Karakter berjalan Lurus ke belakang SC-06-04	

1. Menekan tombol S	
	2.Menggerakan Karakter berjalan ke belakang
Skenario Alternatif 4: Pergerakan Karakter berjalan ke samping SC-05-05	
Skenario Alternatif 5: Pergerakan Karakter berjalan ke samping kiri SC-05-06	
1. Menekan tombol A	
	2.Menggerakan Karakter berjalan ke kiri
Skenario Alternatif 6: Pergerakan Karakter berjalan ke samping kanan SC-06-07	
3. Menekan tombol D	
	4.Menggerakan Karakter berjalan ke kanan
Skenario Normal 1: Pergerakan Karakter berlari SC-06-08	
Skenario Alternatif 7: Pergerakan Karakter berlari Lurus SC-06-09	
Skenario Alternatif 8: Pergerakan Karakter berlari Lurus ke depan SC-06-10	
1. Menekan tombol Shift dan W	
	2.Menggerakan Karakter berlari ke depan
Skenario Alternatif 9 : Pergerakan Karakter berlari Lurus ke belakang SC-06-11	
3. Menekan tombol Shift dan S	
	4.Menggerakan Karakter berlari ke

	belakang
Skenario Normal 2: Pergerakan Karakter berlari Ke samping SC-06-12	
Skenario Alternatif 10: Pergerakan Karakter berlari Ke samping kiri SC-06-13	
5. Menekan tombol Shift dan A	
	6.Menggerakan Karakter berlari ke kiri
Skenario Alternatif 11: Pergerakan Karakter berlari Ke samping kanan SC-06-14	
7. Menekan tombol Shift dan D	
	8.Menggerakan Karakter berlari ke kanan
Skenario Normal 3 : Pergerakan Karakter Meruduk SC-05-15	
9. Menekan tombol X	
	10.Menggerakan Karakter merunduk
Skenario Normal 4 : Pergerakan Karakter Melompat SC-05-16	
11. Menekan tombol *Space*	
	12.Menggerakan Karakter merunduk

3.2.7.4.6.1. *Aktivity Diagram Use Case* Skenario Pergerakan Karakter



Gambar 3.35 *Activity Diagram* pergerakan Karakter

3.2.7.4.7. *Use Case* Skenario Mengganti Senjata

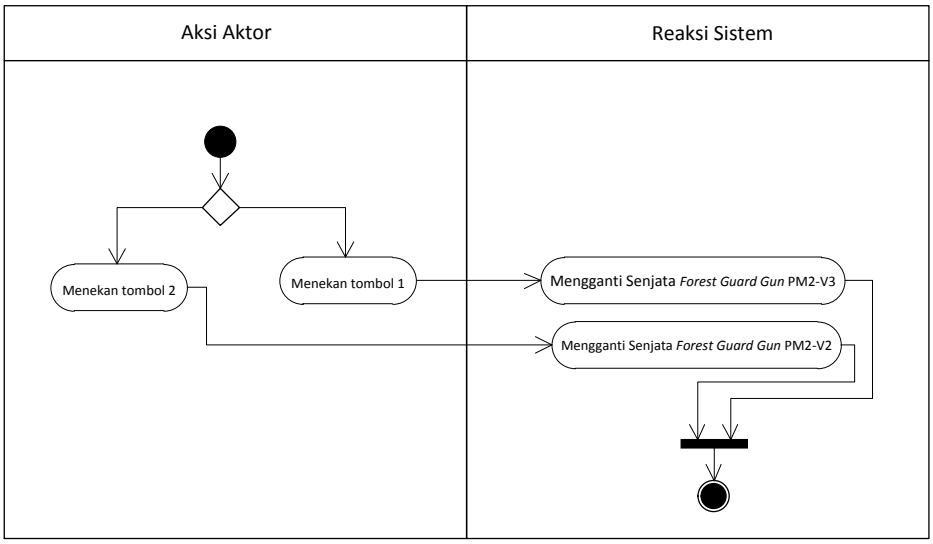
Nama *Use Case* : Mengganti Senjata (UC-07)

Tabel 3.29 Skenario *Use Case* Mengganti Senjata

Aksi Aktor	Reaksi Sitem
Skenario Normal : Mengganti Senjata SC-07-01	
Skenario Alternatif 1 : Mengganti Senjata *Forest Guard Gun* PM2-V3 SC-07-02	

1.Menekan tombol 1	
	2.Mengganti Senjata *Forest Guard Gun* PM2-V3
Skenario Alternatif 1 : Mengganti Senjata *Forest Guard Gun* PM2-V2 SC-07-03	
1.Menekan tombol 2	
	2.Mengganti Senjata *Forest Guard Gun* PM2-V2

3.2.7.4.7.1. *Aktivity Diagram Use Case* Skenario Mengganti Senjata



Gambar 3.36 *Activity Diagram* Mengganti Senjata

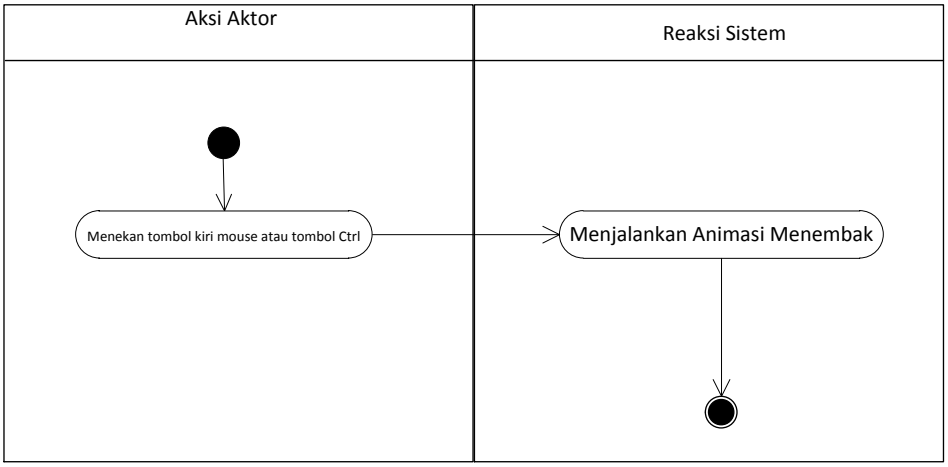
3.2.7.4.8. *Use Case* Skenario Menembak

Nama *Use Case* : Menembak (UC-08)

Tabel 3.30 Skenario *Use Case* Menembak

Aksi Aktor	Reaksi Sitem
Skenario Normal : Mengganti Senjata SC-08-01	
1.Menekan tombol kiri *Mouse* atau Menekan Tombol Ctrl	
	2.Menjalankan Animasi Menembak

3.2.7.4.8.1. *Aktivity Diagram Use Case* Skenario Menembak



Gambar 3.37 *Activity Diagram* Menembak

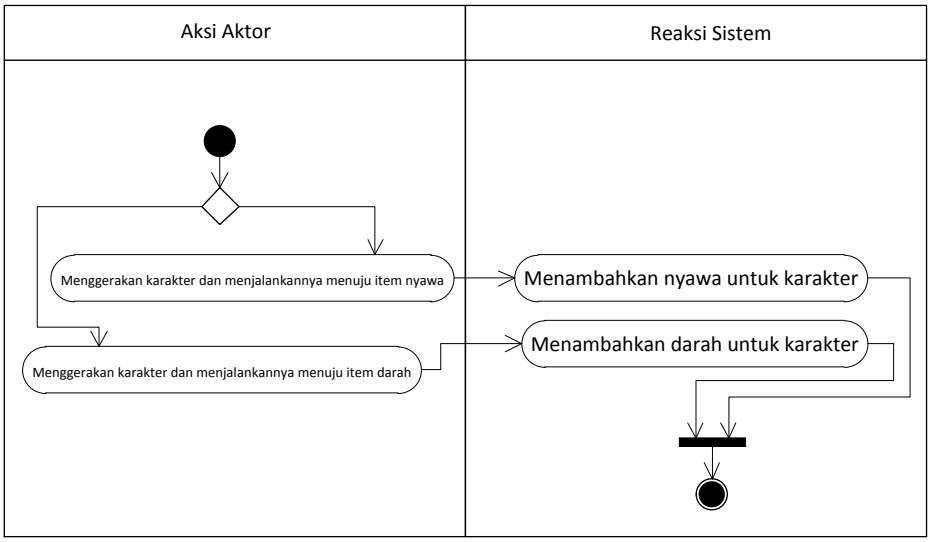
3.2.7.4.9. *Use Case* Skenario Mangambil Nyawa dan Darah

Nama *Use Case* : Mengambil Darah (UC-09)

Tabel 3.31 Skenario *Use Case* Mengambil nyawa dan darah

Aksi Aktor	Reaksi Sitem
Skenario Normal : Mengambil Nyawa dan Darah SC-09-01	
1.Menggerakan karakter dan menjalankannya menuju *item* Nyawa	
	2.Menambahkan Nyawa untuk karakter
Skenario Normal : Mengambil Nyawa dan Darah SC-09-02	
1.Menggerakan karakter dan menjalankannya menuju *item* Darah	
	2.Menambahkan darah untuk karakter

3.2.7.4.9.1. *Aktivity Diagram Use Case* Skenario Mengambil Nyawa dan Darah



Gambar 3.38 *Activity Diagram* Mengambil Darah

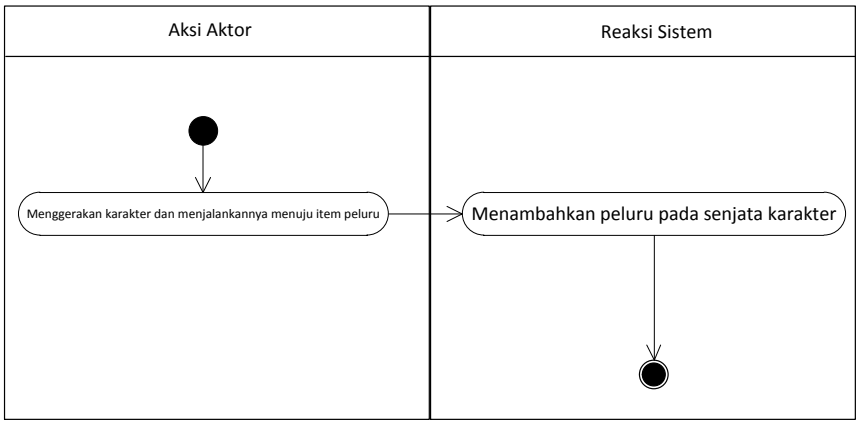
3.2.7.4.10. *Use Case* Skenario Mengambil Peluru

Nama *Use Case* : Mengambil Peluru (UC-10)

Tabel 3.32 Skenario *Use Case* Mengambil Peluru

Aksi Aktor	Reaksi Sitem
Skenario Normal : Mengambil Peluru SC-10-01	
1.Menggerakan karakter dan menjalankannya menuju *item* Peluru	
	2.Menambahkan Peluru pada senjata karakter

3.2.7.4.10.1. *Aktivity Diagram Use Case* Skenario Mengambil Peluru



Gambar 3.39 *Activity Diagram* Mengambil Peluru

3.2.7.4.11. *Use Case* Skenario Memberhentikan Permainan

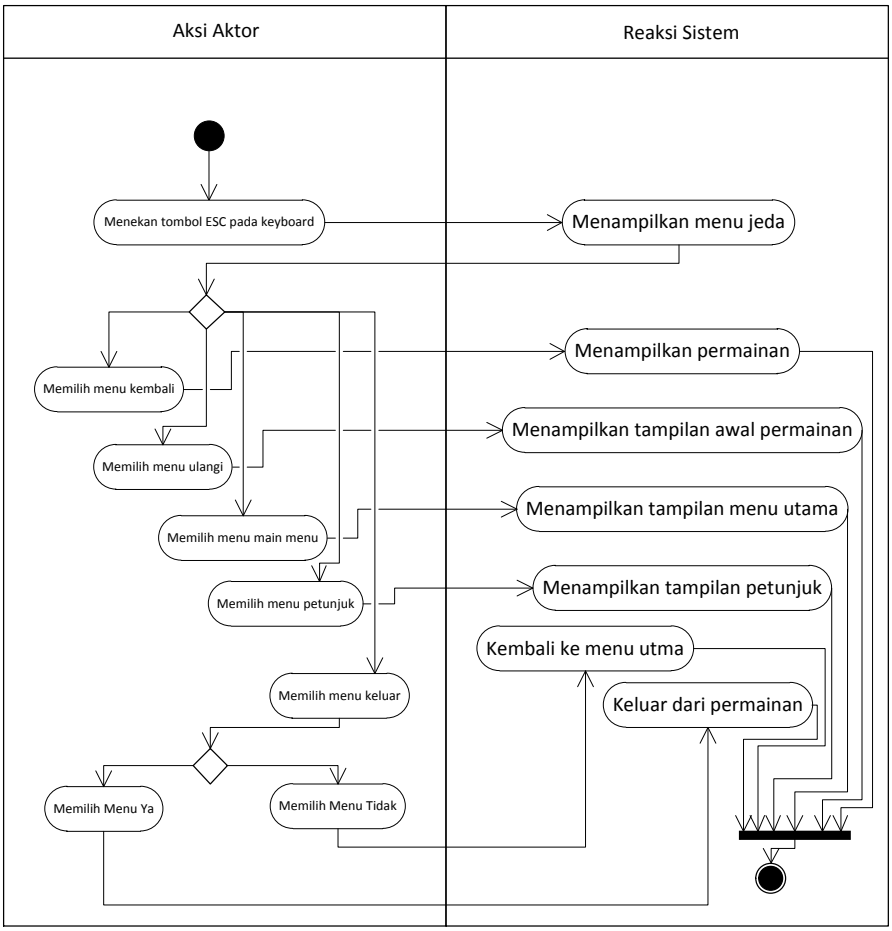
Nama *Use Case* : Memberhentikan permainan (UC-11)

Tabel 3.33 Skenario *Use Case* Memberhentikan Permainan

Aksi Aktor	Reaksi Sitem
Skenario Normal : Memberhentikan Permainan SC-11-01	
1.Menekan tombol Esc	
	2.Menampilkan menu jeda
Skenario Alternatif : Memilih *Menu* Kembali SC-11-02	
1.Memilih *Menu* Kembali	
	2.Menampilkan Permainan
Skenario Alternatif 1 : Memilih *Menu* Ulangi SC-11-03	
1.Memilih *Menu* Ulangi	
	2.Menampilkan Awal Permainan
Skenario Alternatif 2 : Memilih *Menu Main Menu* SC-11-04	
1.Memilih *Menu Main Menu*	
	2.Menampilkan tampilan Menu Utama
Skenario Alternatif 3 : Memilih *Menu* Petunjuk SC-11-05	
1.Memilih *Menu* Petunjuk	
	2.Menampilkan tampilan Petunjuk
Skenario Alternatif 4 : Memilih *Menu* Keluar Ya SC-11-06	
1.Memilih *Menu* Keluar Ya	

	2.Keluar dari Permainan
Skenario Alternatif 4 : Memilih *Menu* Keluar Tidak SC-11-07	
1.Memilih *Menu* Keluar Tidak	
	2.Kembali ke *pause menu*

3.2.7.4.11.1. *Aktivity Diagram Use Case* Skenario Memberhentikan Permainan

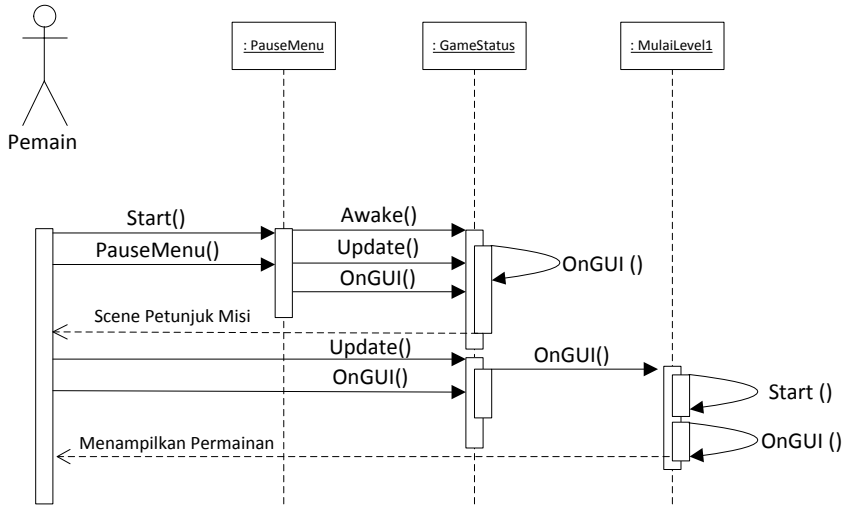


Gambar 3.40 *Activity Diagram* Memilih memberhentikan Permainan

3.2.7.5. *Sequence Diagram*

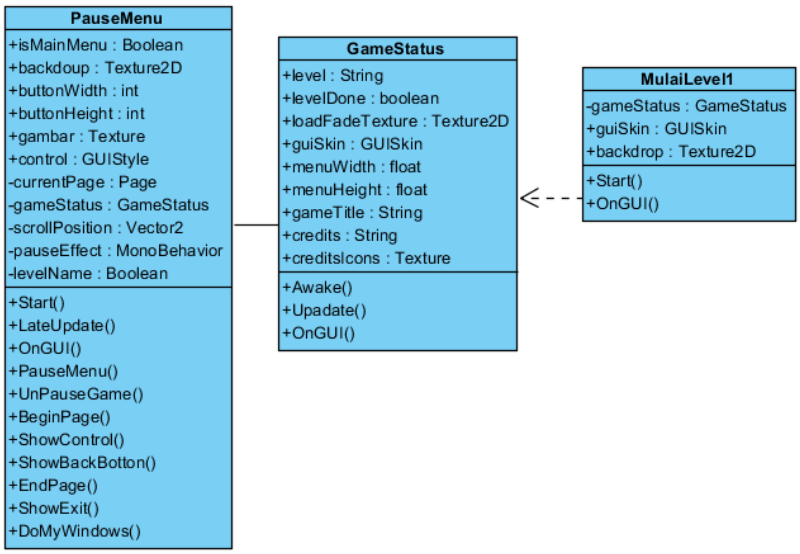
Sequence diagram menggambarkan interaksi antara object di dalam dan di sekitar sistem berupa message yang digambarkan terhadap waktu. Berikut ini beberapa *sequence diagram* yang terdapat pada *game* Selamatkan Orangutan.

3.2.7.5.1. *Sequence Diagram* Memulai permainan



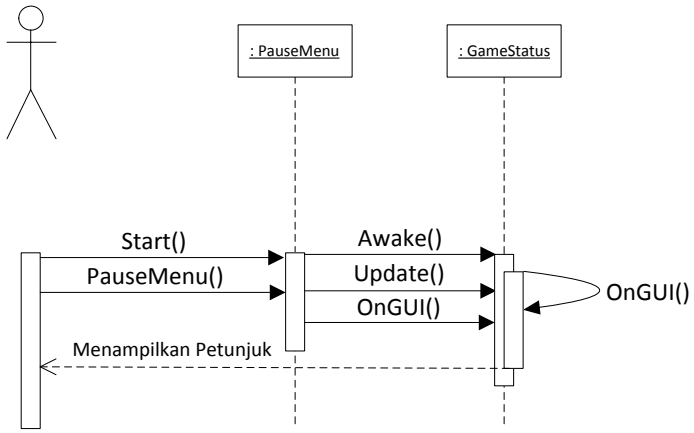
Gambar 3.41 *Sequence Diagram* Memulai permainan

3.2.7.5.1.1 *Class Diagram* Memulai Permainan



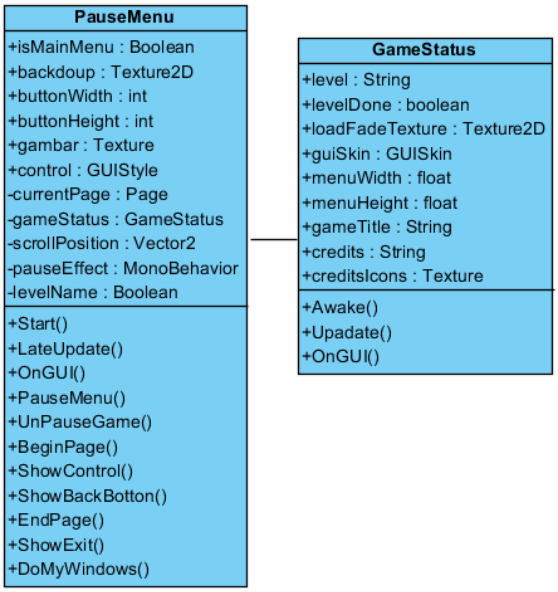
Gambar 3.42 *Class Diagram* Memulai permainan

3.2.7.5.2. *Sequence Diagram* Melihat Petunjuk



Gambar 3.43 *Sequence Diagram* Melihat Petunjuk

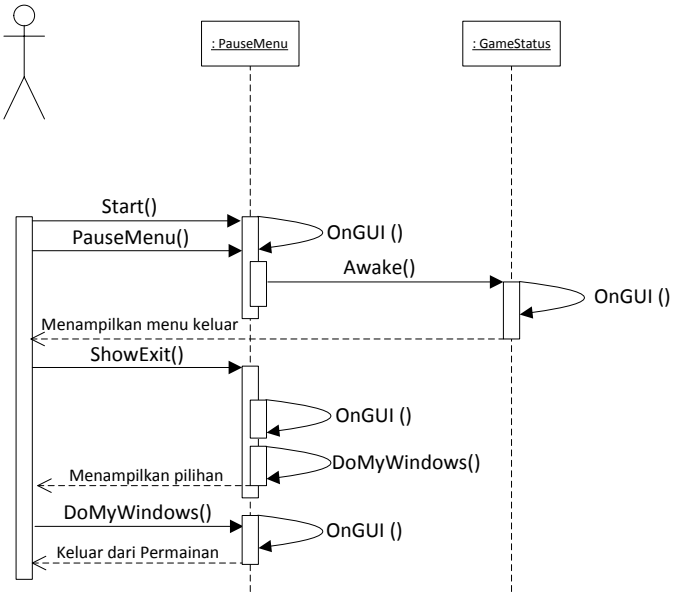
3.2.7.5.2.1. *Class Diagram* Melihat Petunjuk



Gambar 3.44 *Class Diagram* Melihat Petunjuk

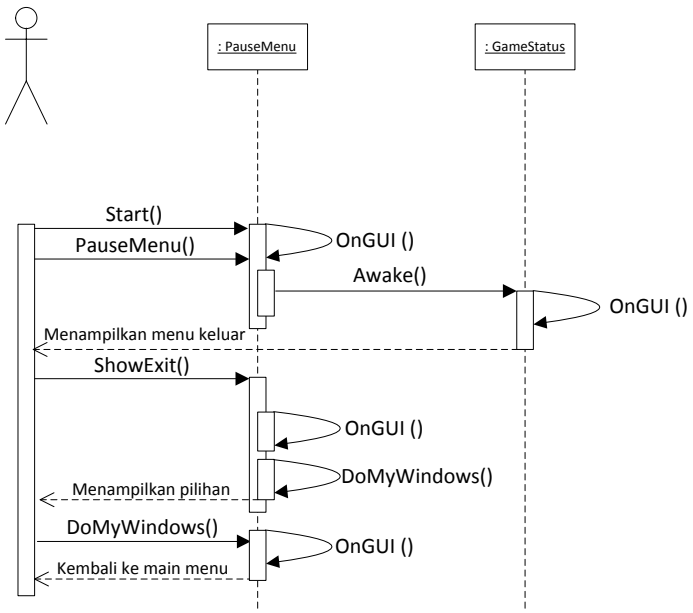
3.2.7.5.3. *Sequence Diagram* Keluar dari Permainan

3.2.7.5.3.1. Sequence Diagram Keluar dari Permainan memilih Ya



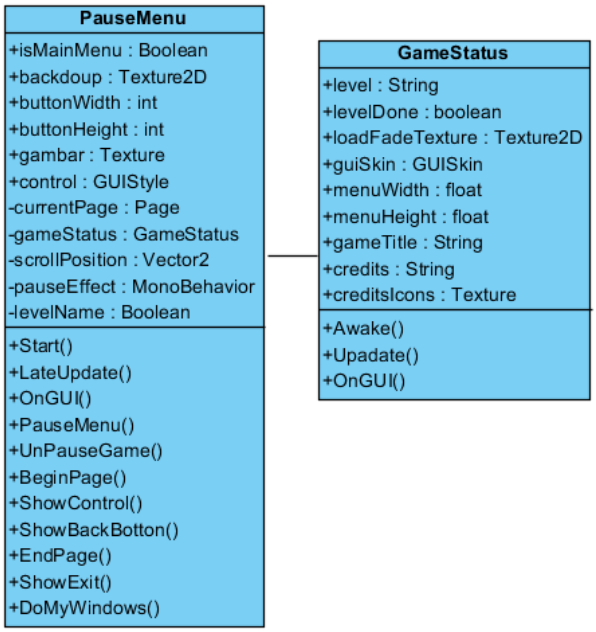
Gambar 3.45 *Sequence Diagram* Keluar dari Permainan Memilih Ya

3.2.7.5.3.2. Sequence Diagram Keluar dari Permainan memilih Tidak



Gambar 3.46 *Sequence Diagram* Keluar dari Permainan memilihTidak

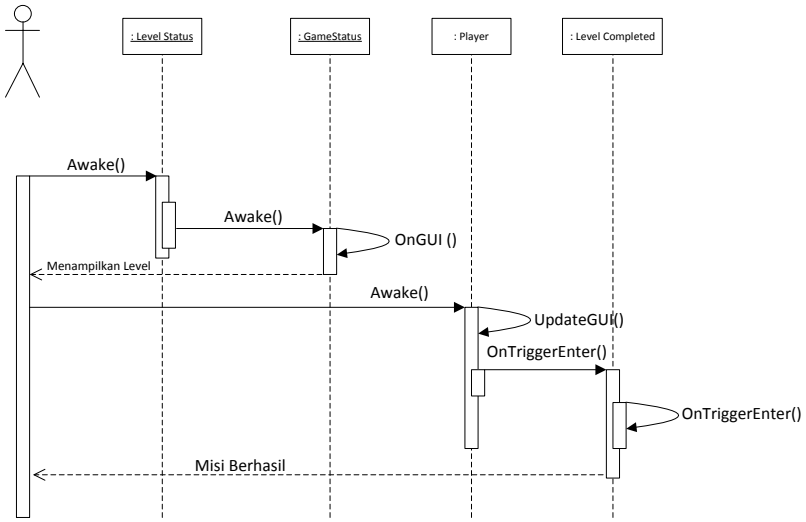
3.2.7.5.3.3. *Class Diagram* keluar dari Permainan



Gambar 3.47 *Class Diagram* keluar dari Permainan

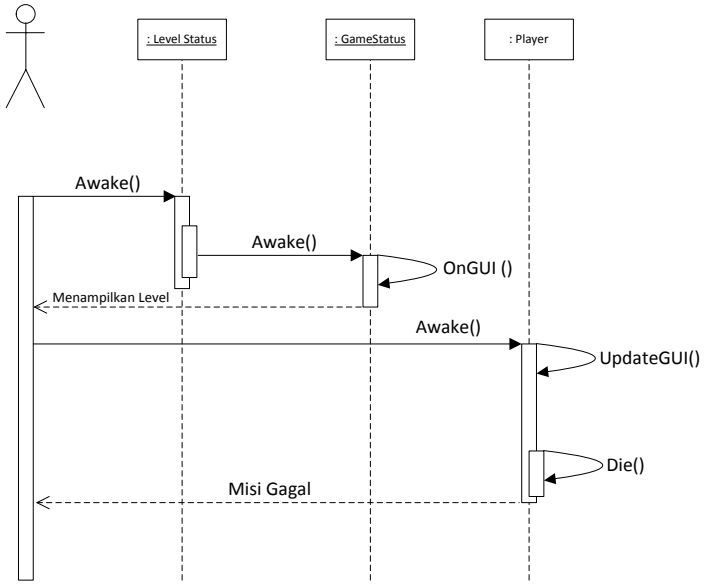
3.2.7.5.4. *Sequence Diagram* Memainkan Permainan

3.2.7.5.4.1. *Sequence Diagram* Memainkan Permainan Berhasil



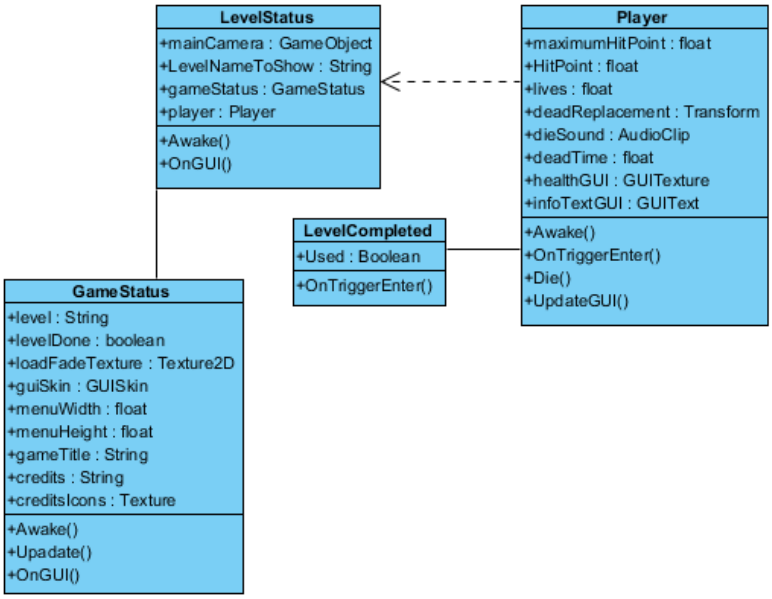
Gambar 3.48 *Sequence Diagram* Memainkan Permainan Berhasil

3.2.7.5.4.2. *Sequence Diagram* Memainkan Permainan Gagal



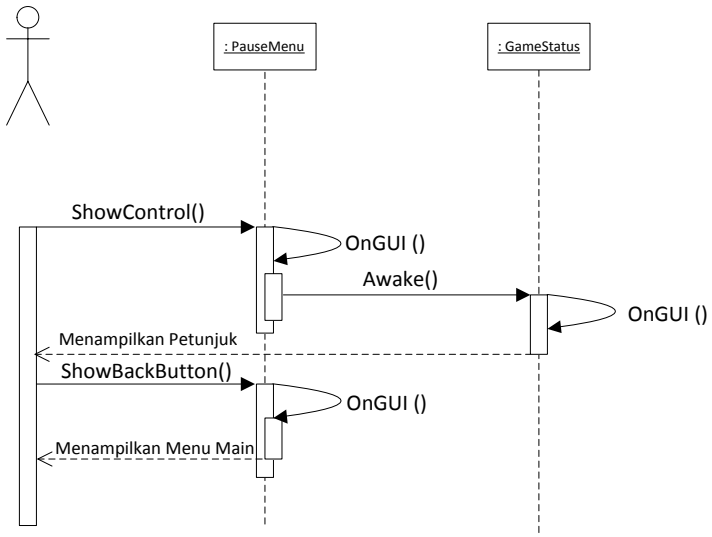
Gambar 3.49 *Sequence Diagram* Memainkan Permainan Gagal

3.2.7.5.4.3. *Class Diagram* Memainkan Permainan



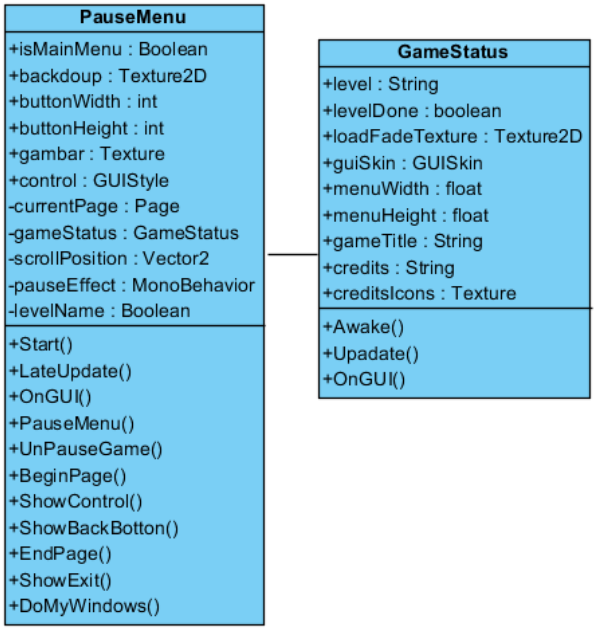
Gambar 3.50 *Class Diagram* Memainkan Permainan

3.2.7.5.5. *Sequence Diagram* Melihat Tampilan Petunjuk



Gambar 3.51 *Sequence Diagram* Melihat Tampilan Petunjuk

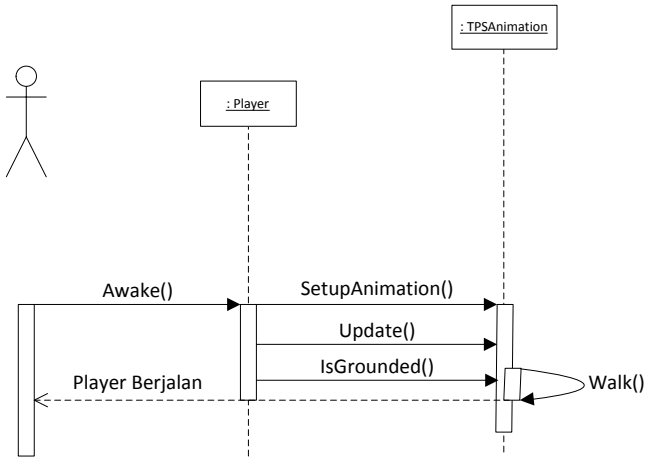
3.2.7.5.5.1. *Class Diagram* Melihat Tampilan Petunjuk



Gambar 3.52 *Class Diagram* Melihat Tampilan Petunjuk

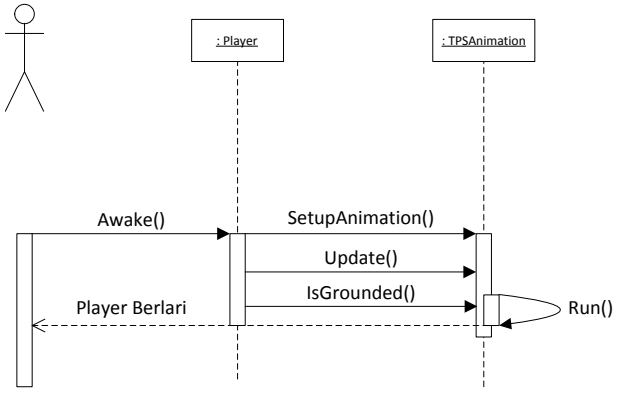
3.2.7.5.6. *Sequence Diagram* Pergerakan Karakter

3.2.7.5.6.1. *Sequence Diagram* Pergerakan Karakter Berjalan



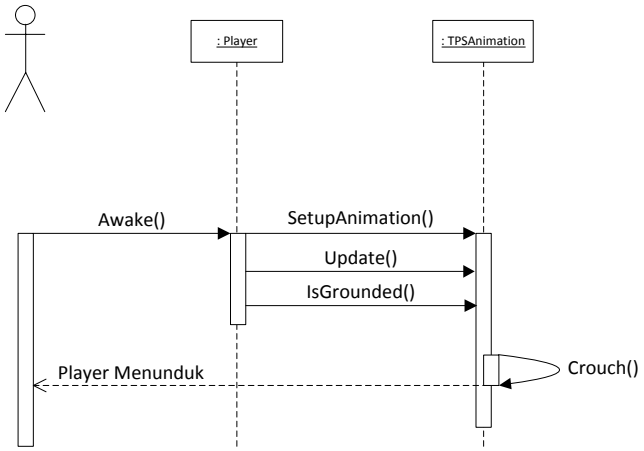
Gambar 3.53 *Sequence Diagram* Pergerakan Karakter Berjalan

3.2.7.5.6.2. *Sequence Diagram* Pergerakan Karakter Berlari



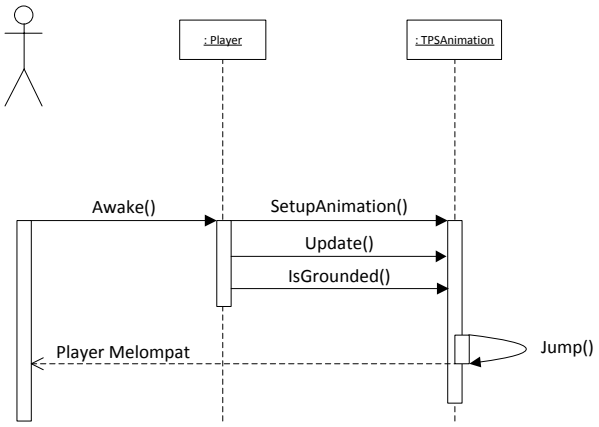
Gambar 3.54 *Sequence Diagram* Pergerakan Karakter Berlari

3.2.7.5.6.3. *Sequence Diagram* Pergerakan Karakter Merunduk



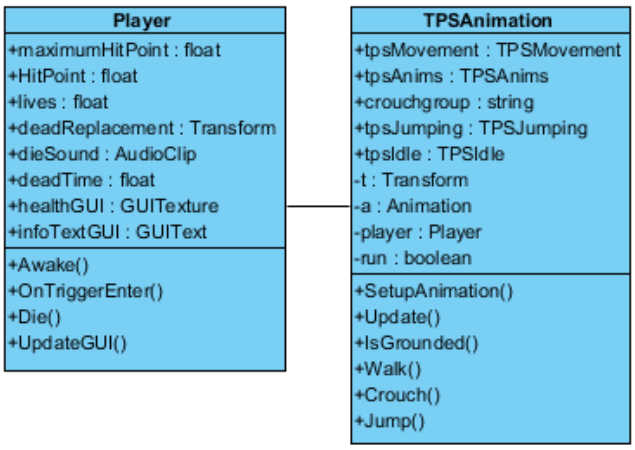
Gambar 3.55 *Sequence Diagram* Pergerakan Karakter Merunduk

3.2.7.5.6.4. *Sequence Diagram* Pergerakan Karakter Melompat



Gambar 3.56 *Sequence Diagram* Pergerakan Karakter Melompat

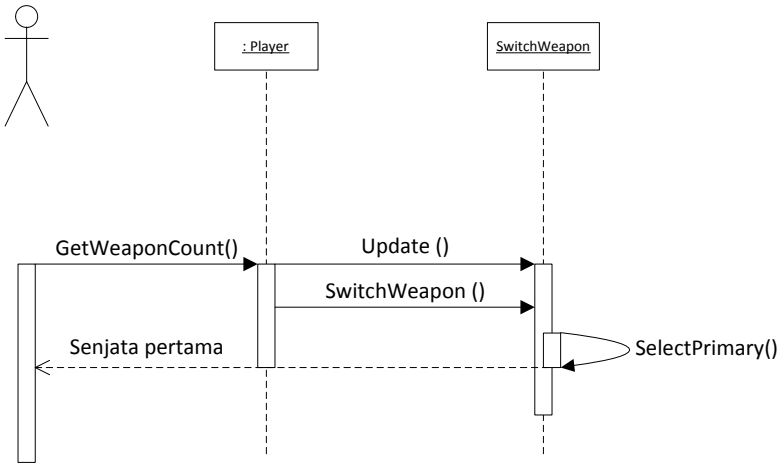
3.2.7.5.6.5. *Class Diagram* Pergerakan Karakter



Gambar 3.57 *Class Diagram* Pergerakan Karakter

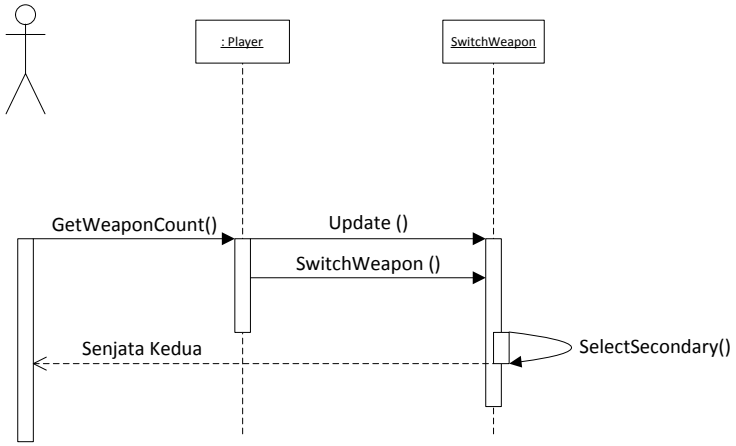
3.2.7.5.7. *Sequence Diagram* Mengganti Senjata

3.2.7.5.7.1. *Sequence Diagram* Mengganti Senjata *Forest Guard Gun* PM2-V3



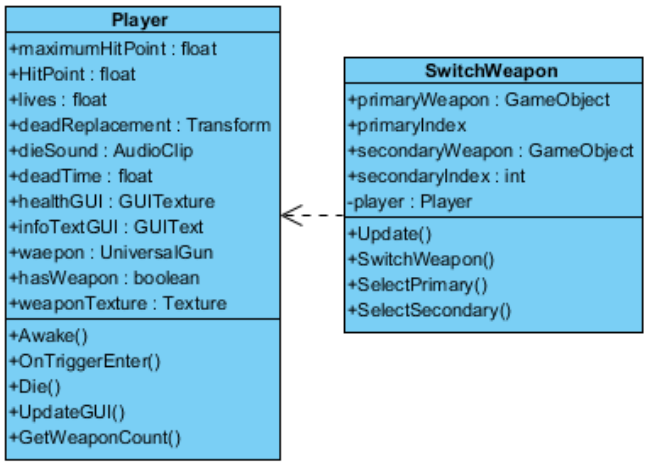
Gambar 3.58 *Sequence Diagram* Mengganti Senjata *Forest Guard Gun* PM2-V3

3.2.7.5.7.2. *Sequence Diagram* Mengganti Senjata *Forest Guard Gun* PM2-V2



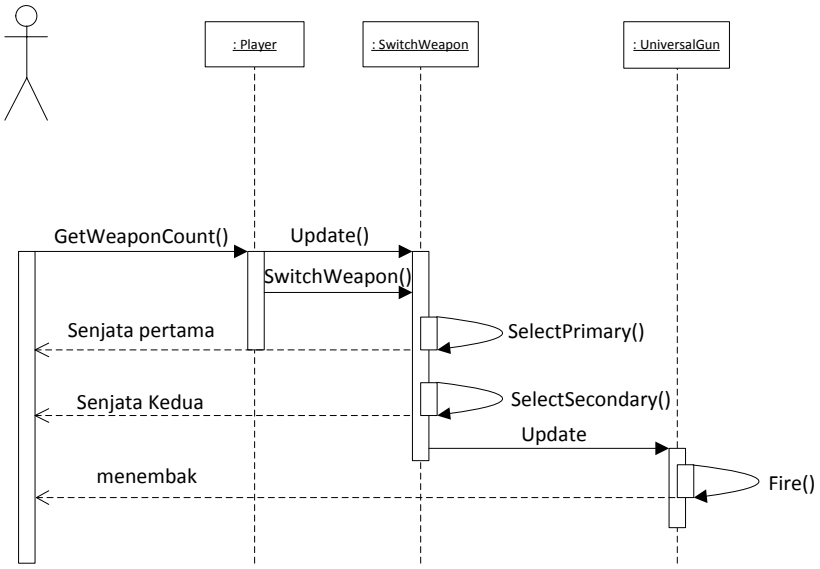
Gambar 3.59 *Sequence Diagram* Mengganti Senjata *Forest Guard Gun* PM2-V2

3.2.7.5.7.3. *Class Diagram* Mengganti Senjata



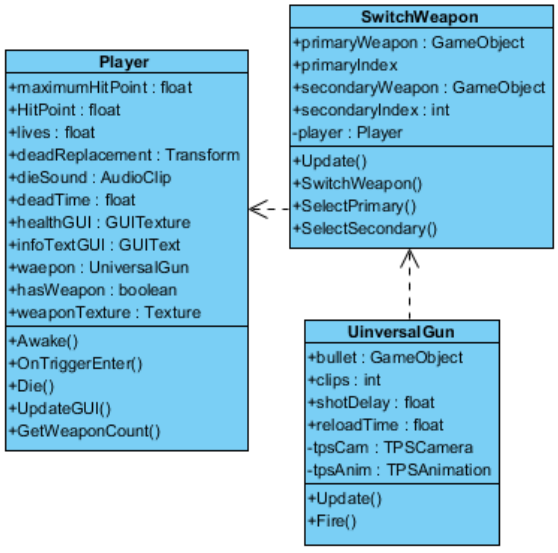
Gambar 3.60 *Sequence Diagram* Mengganti Senjata

3.2.7.5.8. *Sequence Diagram* Menembak



Gambar 3.61 *Sequence Diagram* Menembak

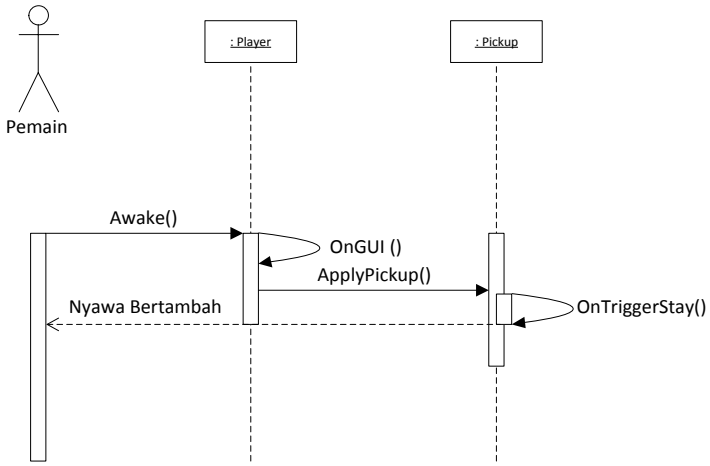
3.2.7.5.8.1. *Class Diagram* Menembak



Gambar 3.62 *Class Diagram* Menembak

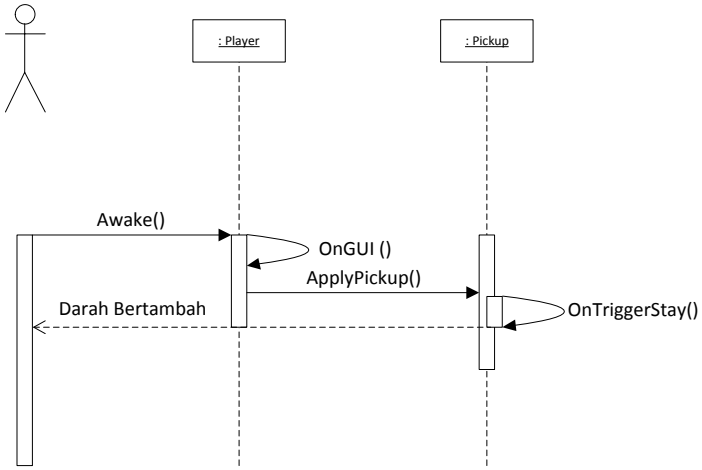
3.2.7.5.9. *Sequence Diagram* Mengambil Nyawa dan Darah

3.2.7.5.10.3. Sequence Diagram Mengambil Nyawa



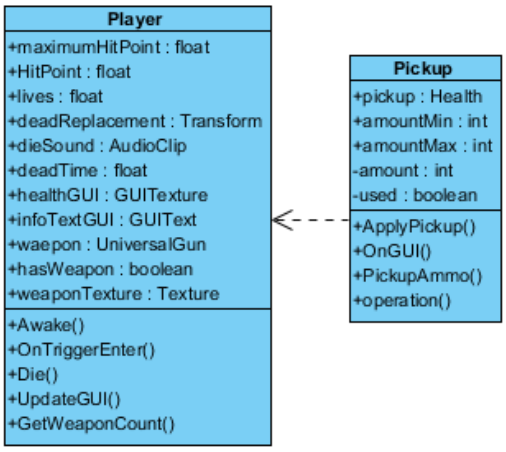
Gambar 3.63 *Sequence Diagram* Mengambil Nyawa

3.2.7.5.10.4. Sequence Diagram Mengambil Darah



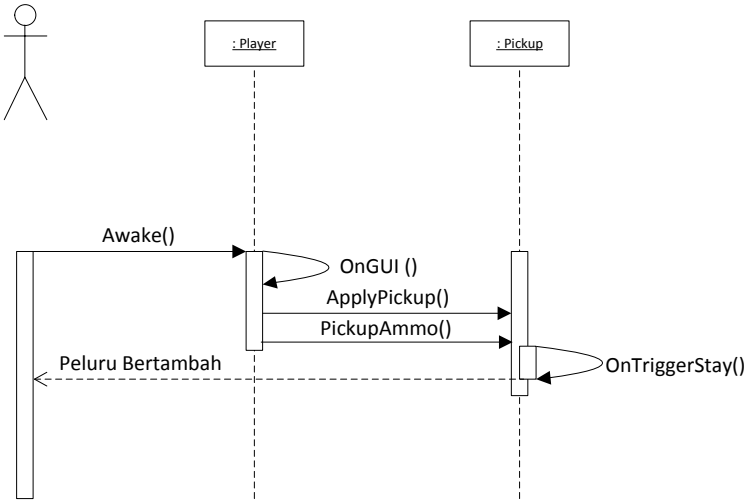
Gambar 3.64 *Sequence Diagram* Mengambil Darah

3.2.7.5.10.5. *Class Diagram* Mengambil Nyawa dan Darah



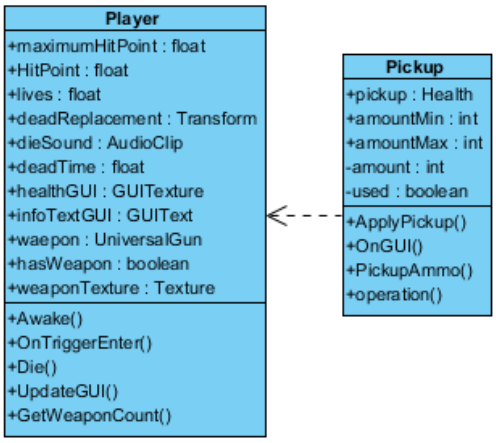
Gambar 3.65 *Class Diagram* Mengambil Darah

3.2.7.5.10. *Sequence Diagram* Mengambil Peluru



Gambar 3.66 *Sequence Diagram* Mengambil Peluru

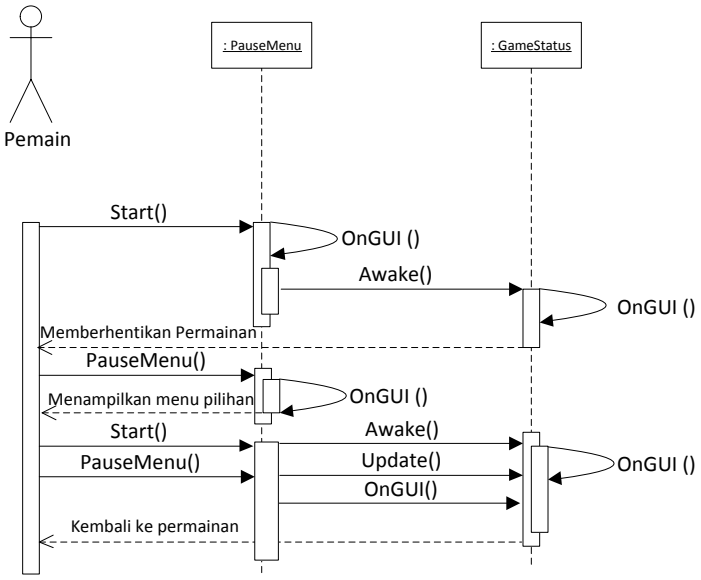
3.2.7.5.11.3. *Class Diagram* Mengambil Peluru



Gambar 3.67 *Class Diagram* Mengambil Peluru

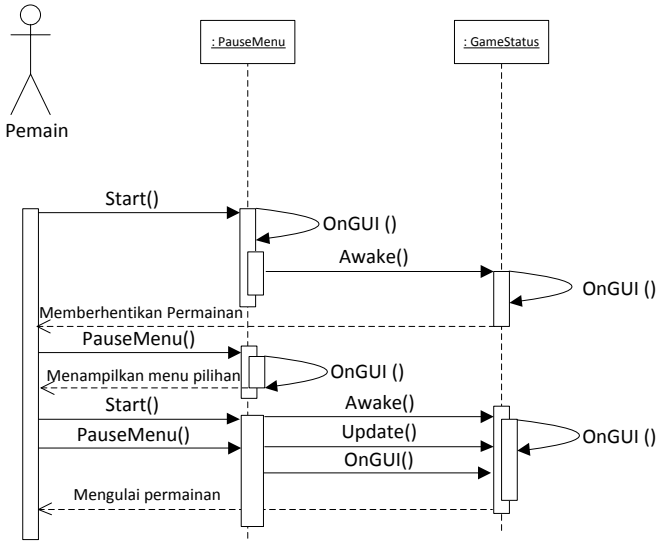
3.2.7.5.11. *Sequence Diagram* Memberhentikan Permainan

3.2.7.5.11.1. *Sequence Diagram* Memberhentikan Permainan memilih *menu* kembali



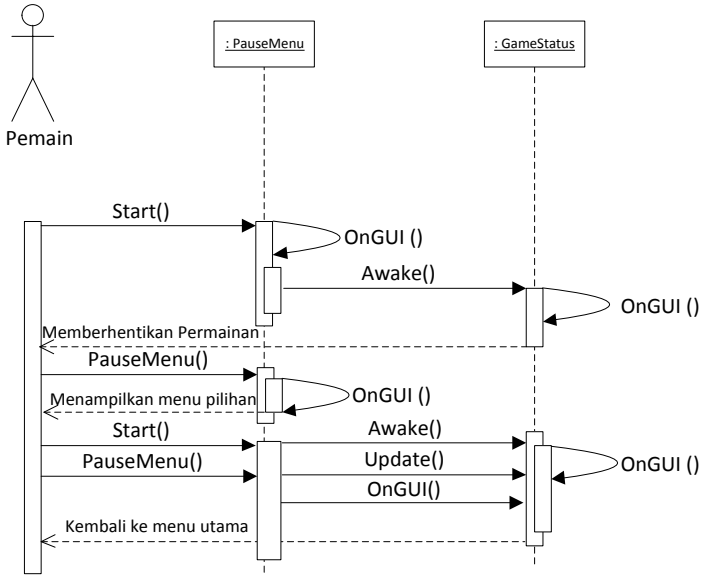
Gambar 3.68 *Sequence Diagram* Memberhentikan Permainan memilih *menu* kembali

3.2.7.5.11.2. *Sequence Diagram* Memberhentikan Permainan memilih *menu* ulangi



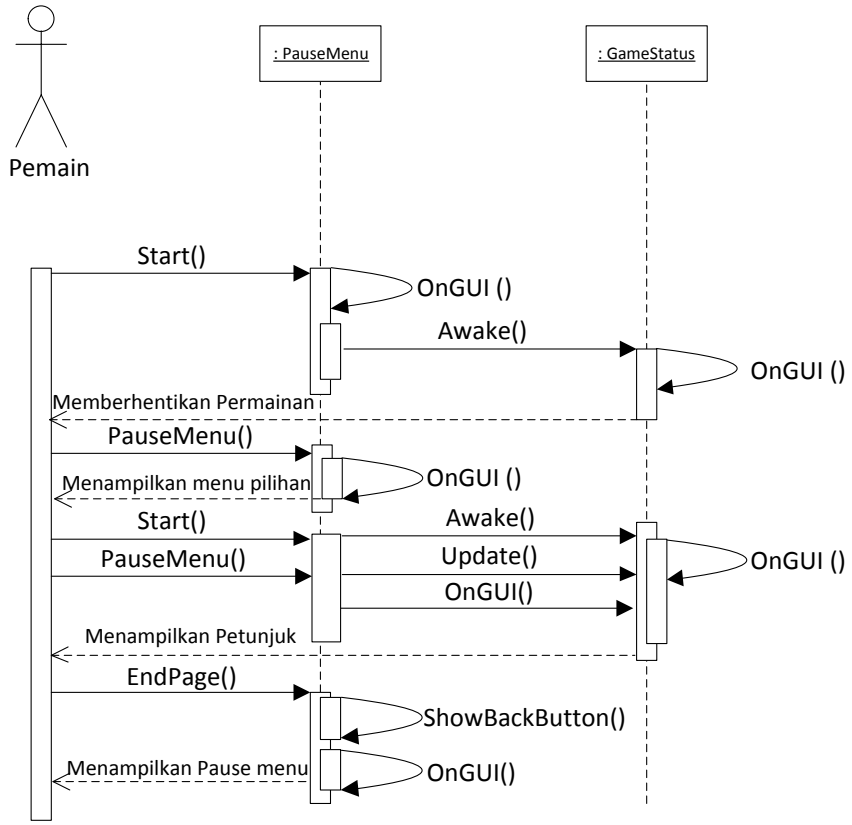
Gambar 3.69 *Sequence Diagram* Memberhentikan Permainan memilih *menu* ulangi

3.2.7.5.11.3. *Sequence Diagram* Memberhentikan Permainan memilih *main menu*



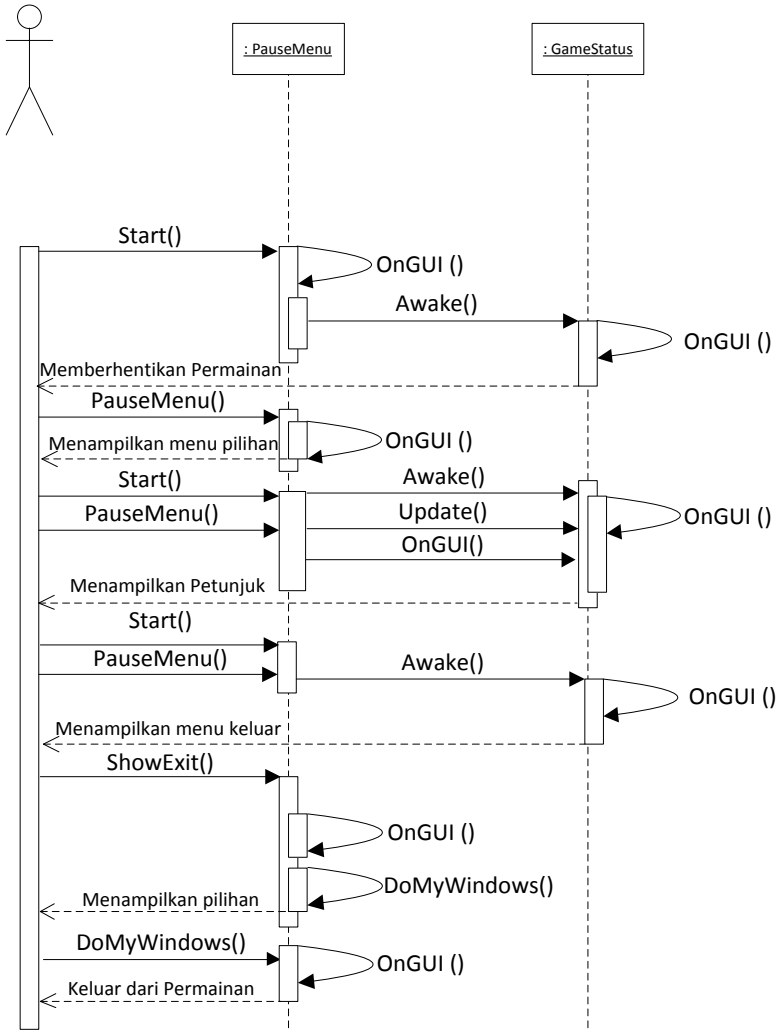
Gambar 3.70 *Sequence Diagram* Memberhentikan Permainan memilih *main menu*

3.2.7.5.11.4. *Sequence Diagram* Memberhentikan Permainan memilih *menu* Petunjuk



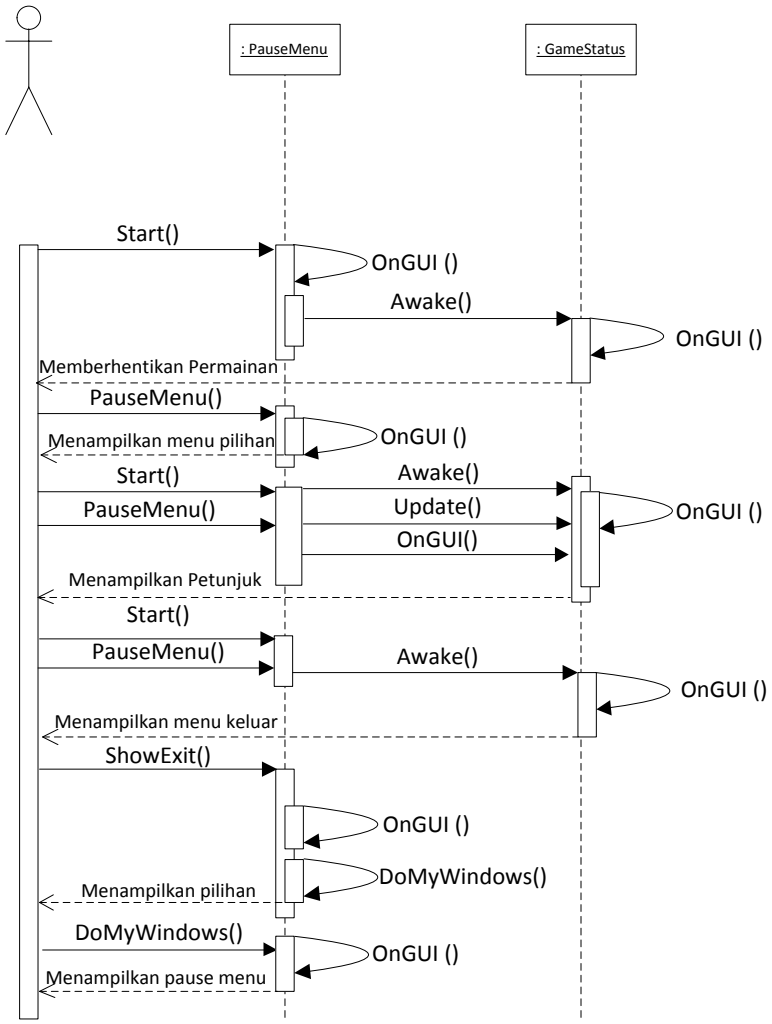
Gambar 3.71 *Sequence Diagram* Memberhentikan Permainan memilih *menu* Petunjuk

3.2.7.5.11.5. *Sequence Diagram* Memberhentikan Permainan memilih *menu* keluar memilih Ya



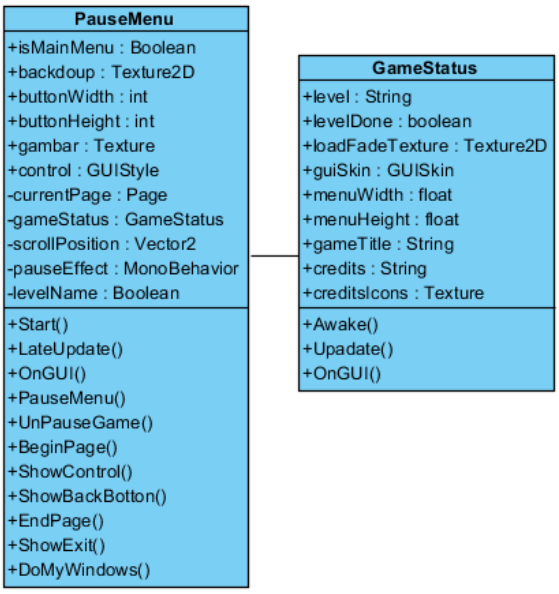
Gambar 3.72 *Sequence Diagram* Memberhentikan Permainan memilih *menu* Keluar memilih Ya

3.2.7.5.11.6. *Sequence Diagram* Memberhentikan Permainan memilih *menu* Petunjuk memilih Tidak



Gambar 3.73 *Sequence Diagram* Memberhentikan Permainan memilih *menu* Keluar memilih Tidak

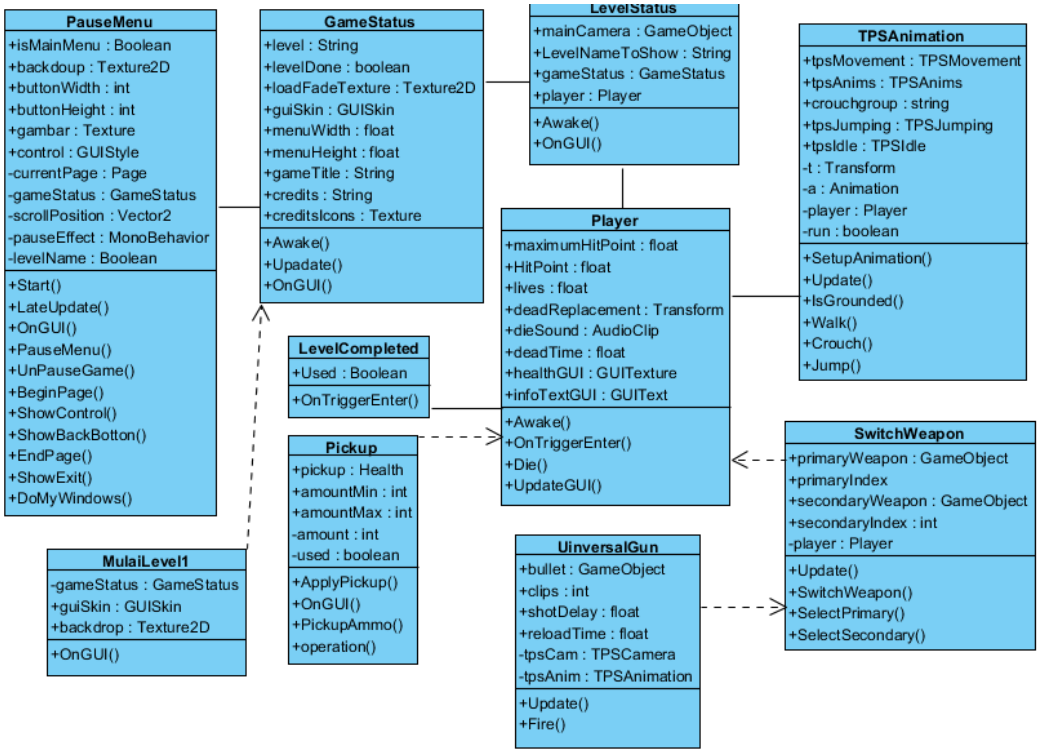
3.2.7.5.11.7. *Class Diagram* Memberhentikan Permainan



Gambar 3.74 *Class Diagram* Memberhentikan Permainan

3.2.7.6. *Class Diagram* Keseluruhan

Class adalah sebuah spesifikasi yang jika diinstansiasi akan menghasilkan sebuah objek dan merupakan inti dari pengembangan dan desain berorientasi objek. *Class* menggambarkan keadaan (atribut/properti) suatu sistem, sekaligus menawarkan layanan untuk memanipulasi keadaan tersebut (metoda/fungsi). *Class diagram* menggambarkan struktur dan deskripsi *class, package* dan objek beserta hubungan satu sama lain seperti *containment*, pewarisan, asosiasi dan lain-lain. berikut adalah *class diagram* dari aplikasi *game*.



Gambar 3.75 *Class Diagram* Keseluruhan

3.3. Perancangan Sistem

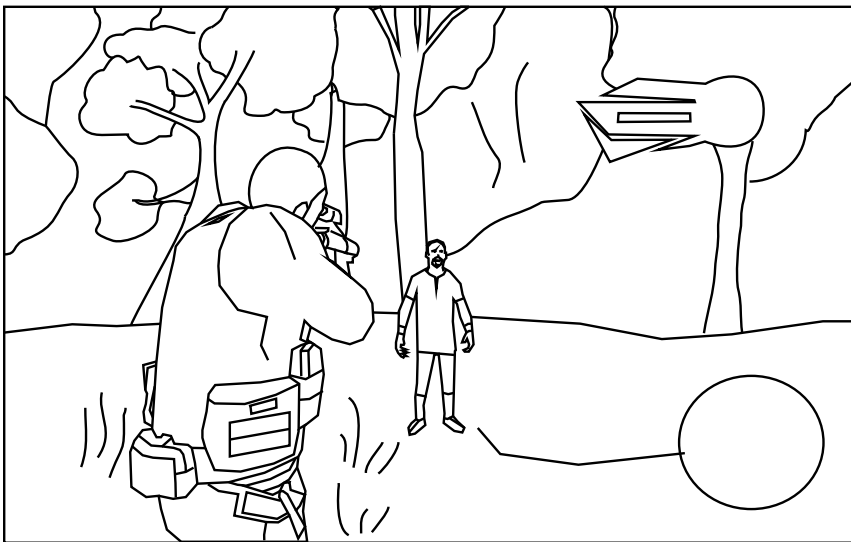
Perancangan adalah suatu bagian dari metodologi pengembangan pembangunan suatu perangkat lunak yang dilakukan setelah tahapan untuk memberikan gambaran secara terperinci. Berdasarkan uraian diatas perancangan sistem merupakan tahapan dari siklus pengembangan sistem yang didefinisikan dari kebutuhan-kebutuhan fungsional dan persiapan untuk rancang bangun implementasi yang menggambarkna bagaimana suatu sistem dibentuk yang dapat berupa penggambaran, perancangan dan pembuatan sketsa atau pengaturan dari beberapa elemen yang terpisah ke dalam satu kesatuan yang utuh dan berfungsi juga menyangkut konfigurasi dari komponen-komponen perangkat keras dan perangkat lunak dari suatu sistem.

3.3.1. Perancangan Komponen Permainan

Perancangan Komponen Permainan meliputi karakter dan *storyboard* yang didalamnya menjelaskan karakter beserta senjata apa saja yang dimiliki pemain maupun NPC dan *storyboard* sebagai gambaran perancangan pembuatan *game* Selamatkan Orangutan.

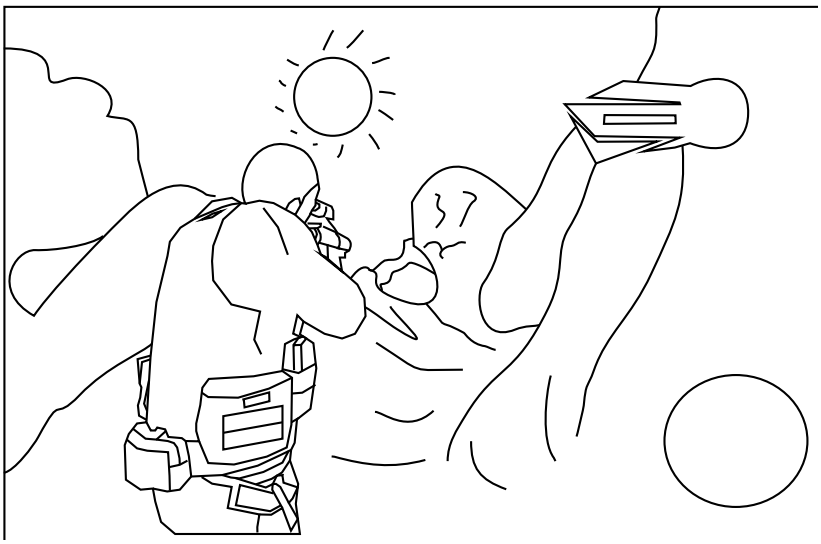
3.3.1.1. *Storyboard*

Storyboard adalah sketsa gambar yang disusun berurutan sesuai dengan naskah, dengan *storyboard* dapat menyampaikan ide carita kepada orang lain dengan lebih mudah, karena dapat menggiring khayaln seseorang mengikuti gambar-gambar yang tersaji, sehingga menghasilkan persepsi yang sama pada ide carita. Adapun *storyboard* untuk *game* Selamatkan Orangutan. ini adalah sebagai berikut.



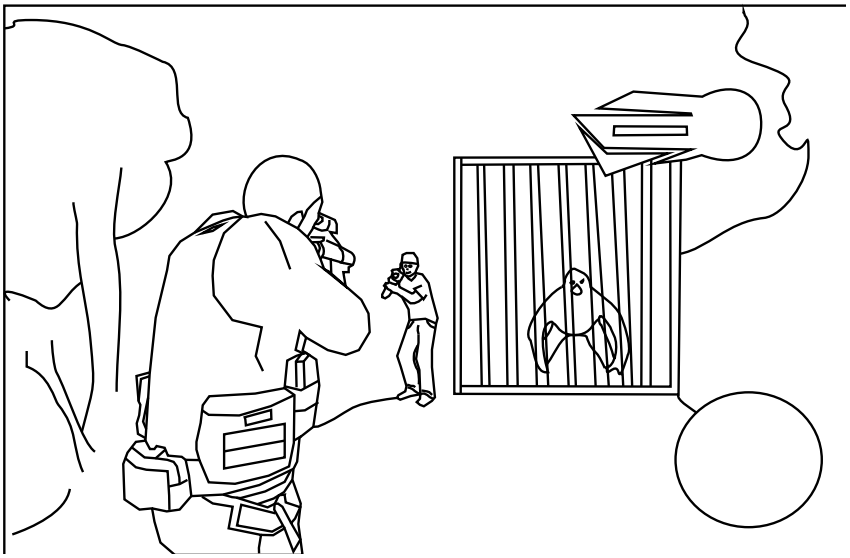
Gambar 3.76 *Storyboard Level* 1

Gambar 3.76 Menceritakan ketika pemain memulai permainan di awal *level* dengan berada disebuah hutan di daerah Kalimantan dan disinilah awal dari permainan. Pemain akan bertemu dengan seorang warga desa yang sedang mencari kayu.



Gambar 3.77 *Storyboard Level* 1 dalam permainan

Gambar 3.77 Menceritakan tentang pemain yang harus mencari sebuah desa dengan menempuh rintangan tanjakan yang curam.



Gambar 3.78 *Storyboard Level* 2

Gambar 3.78 Menceritakan tentang pemain yang harus menyelamatkan orangutan dari kurungan dan harus membawa orangutan kesebuah penangkaran.



Gambar 3.79 *Storyboard Level* 3

Gambar 3.79 Menceritakan tentang pemain yang diserang oleh boss yang mempunyai senjata. Pemain harus mencari orangutan dan menyelamatkannya sehingga misi terselesaikan.

3.3.1.2. Pengenalan Karakter

Karakter utama dalam g*ame* ini bernama sonny, sonny merupakan seorang polisi hutan yang ditugaskan pergi ke hutan untuk menyelamatkan habitat orangutan. karakter dirancang untuk melawan musuh-musuh yang ada disetiap *level*. Karakter utama ini dilengkapi dengan persenjataan berupa snapan dan *sniper*. *Game* ini merupakan *genre third person shooter* yang mempunyai sudut pandang orang ketiga, bentuk karakter dan *properties* yang dimiliki oleh karakter utama dapat dilihat pada gamabr 3.80, 3.81, 3.82 Dan Tabel 3.34.



Gambar 3.80 Karakter Utama Tampak Depan



Gambar 3.81 Karakter Utama Tampak Belakang



Gambar 3.82 Karakter Utama Tampak Sisi

Tabel 3.34 Detail Karakter

Nama	Sonny
Nyawa	3
Darah	100
Kemampuan	Berjalan, berlari, melompat, menembak dan merunduk
Senjata	*Forest Guard Gun* PM2-V2 dan *Forest Guard Gun* PM2-V3

Senjata yang digunakan oleh Sonny adalah senjata *Forest Guard Gun* PM2-V2 yang memiliki peluru sebanyak 30 buah dan senjata *Forest Guard Gun* PM2-V3 yang memilki peluru 20 buah yang dapat dilihat pada Gambar 3.83, 3.84 dan *properties* senjata pada karakter dapat dilihat pada tabel 3.35



Gambar 3.83 Senjata *Forest Guard Gun* PM2-V2



Gambar 3.84 Senjata *Forest Guard Gun* PM2-V3

Tabel 3.35 Detail Senjata

Properti	***Forest Guard Gun* PM2-V2**	***Forest Guard Gun* PM2-V3**
Kekuatan	10	20
Peluru	10	10
Jumlah box peluru	3	2

Musuh yang terdapat pada *game* ini adalah manusia. Musuh akan memiliki tingkat kesulitan berbeda-beda untuk dikalahkan. Musuh dapat dilihat pada Gambar 3.85, 3.6 dan 3.87.



Gambar 3.85 Karakter Musuh *Boss*

Tabel 3.36 Detail Karakter Musuh *Boss*

Nama	*Boss*
Darah	100
Kemampuan	Berjalan, berlari, menembak, menyerang , menghidar dan mengambil darah
Senjata	*Machine Gun*



Gambar 3.86 Karakter Musuh Pemburu



Gambar 3.87 Karakter Musuh Anak buah *Boss*

Tabel 3.37 Detail Karakter Musuh Pembru

Nama	Pemburu dan Anak buah
Darah	50
Kemampuan	Berjalan, berlari, dan menembak
Senjata	*Machine Gun*



Gambar 3.88 Karakter Orangutan



Gambar 3.89 Karakter Warga Desa 1



Gambar 3.90 Karakter Kepala Desa



Gambar 3.91 Karakter Warga desa 2

3.3.1.3. Pengenalan *Item*

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

Tabel 3.38 Tabel penjelas *Item* pada *game*

Gambar *Items*	Nama *Items*	Keterangan
	Darah	*Item* yang bisa diambil oleh karakter utama untuk menambah darah atau *hitpoints*. Satu kali ambil bertambah darah 10 *hitpoints.*
	Nyawa	*Item* yang bisa diambil oleh karakter utama untuk menambah nyawa atau *live*. Satu kali ambil bertambah darah 1 *live*.

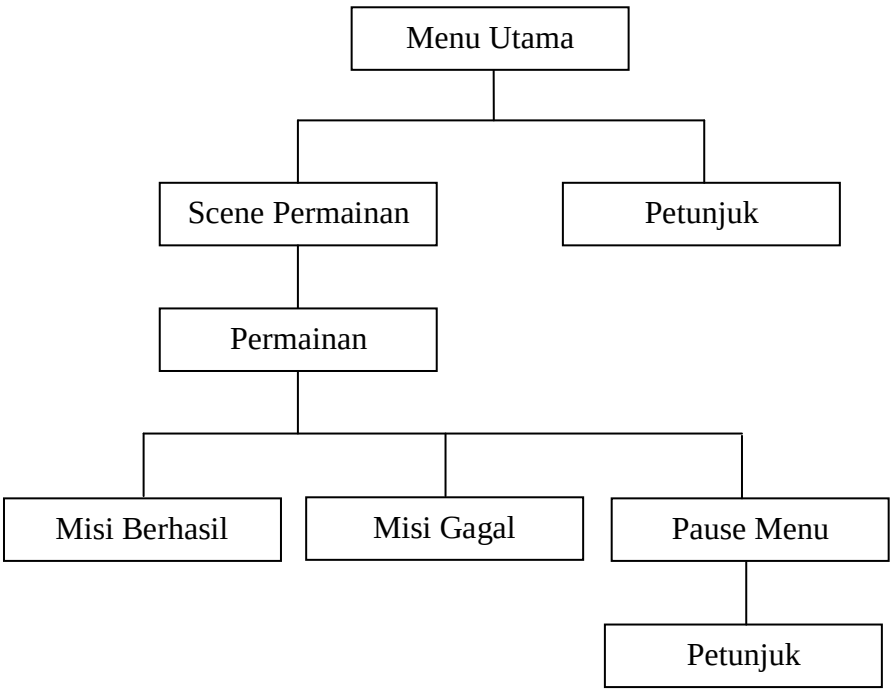
	*Box* Peluru	*Item* yang bisa diambil oleh karakter utama apabila peluru sudah habis.

3.3.2. 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.

3.3.2.1. Perancangan Struktur Menu

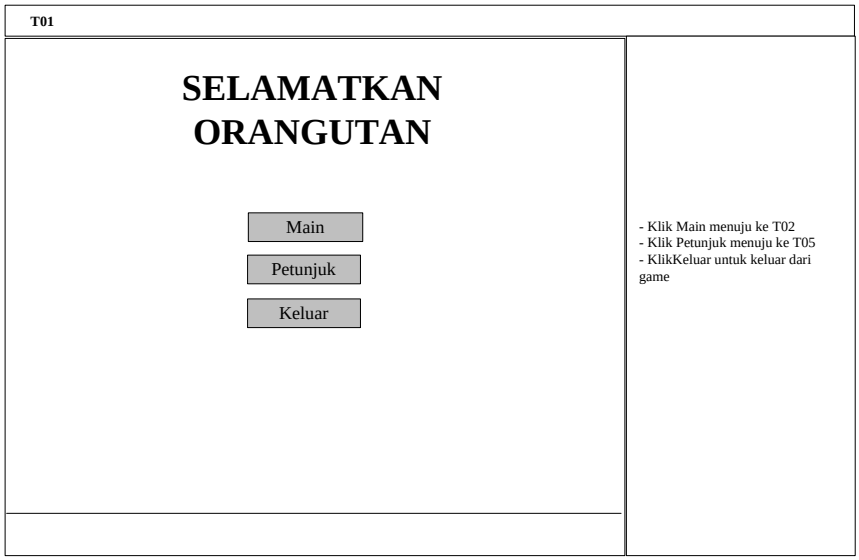
Perancangan menu dibuat dibuat agar pengguna dapat menggunakan plikasi *Game* dengan mudah, sehingga pengguna dapat dengan mudah memilih menu aplikasi yang sedang berjalan. Stuktur *menu* dapat dilihat pada gambar 3.85.



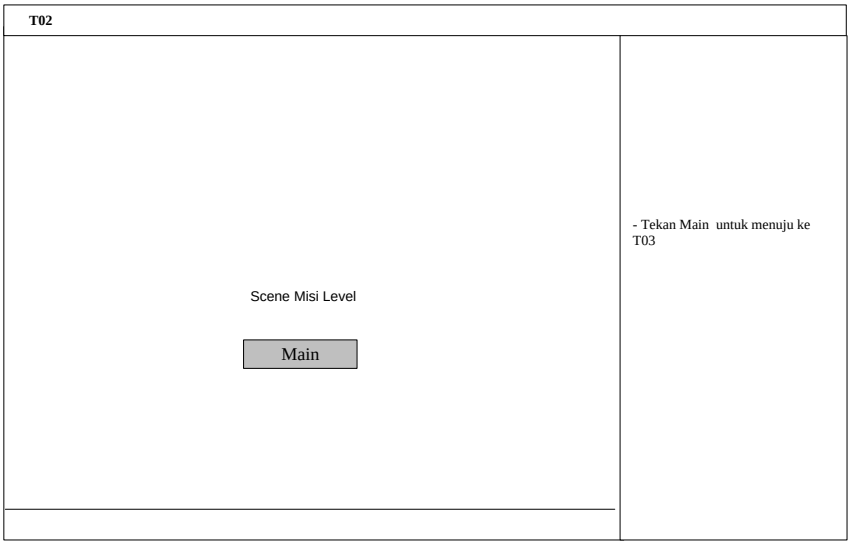
Gambar 3.92 Struktur *Menu* Utama

3.3.2.2. Perancangan Antar Muka

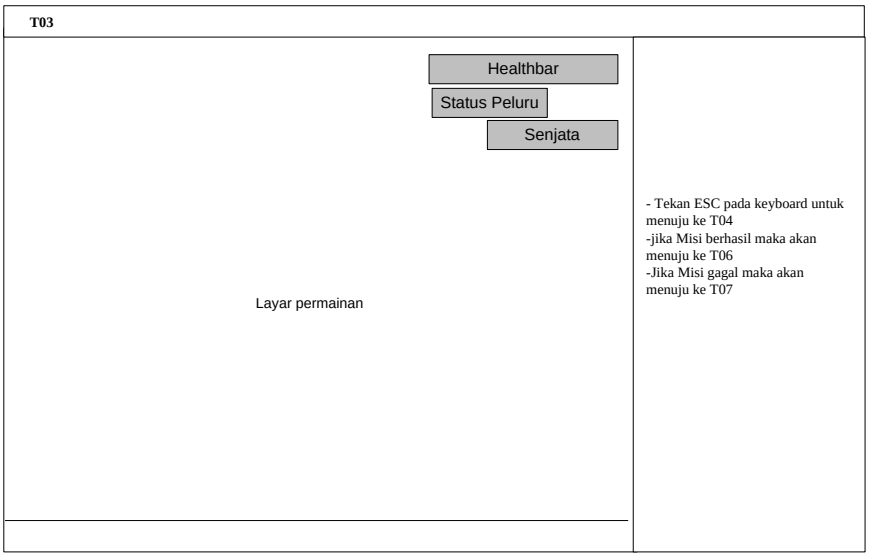
Setelah melakukan perancangan data pada sistem yang dibangun, maka dilakukanlah perancangan arsitektur. Perancangan arsitektur yang telah dibuat meliputi beberapa perancangan diantaranya struktur *menu*, perancangan antarmuka.



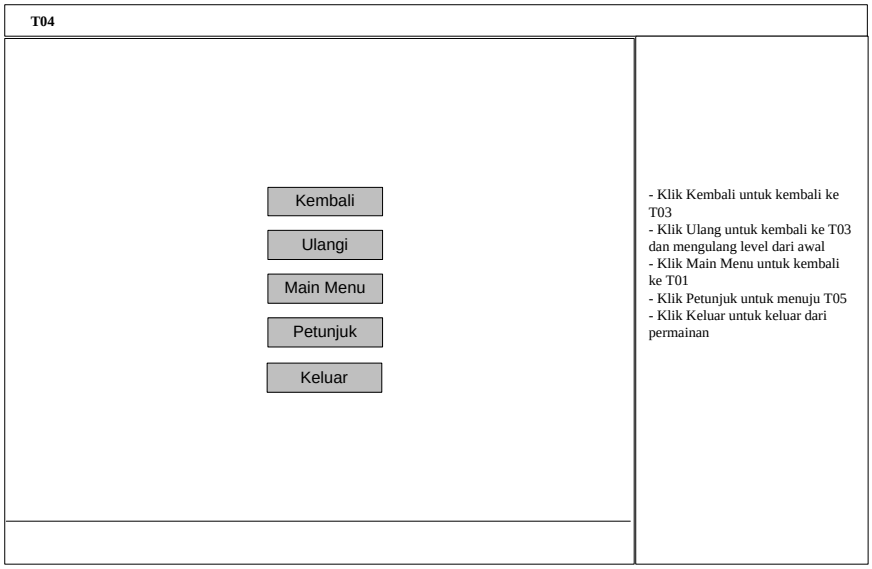
Gambar 3.93 Antarmuka *Menu* Utama



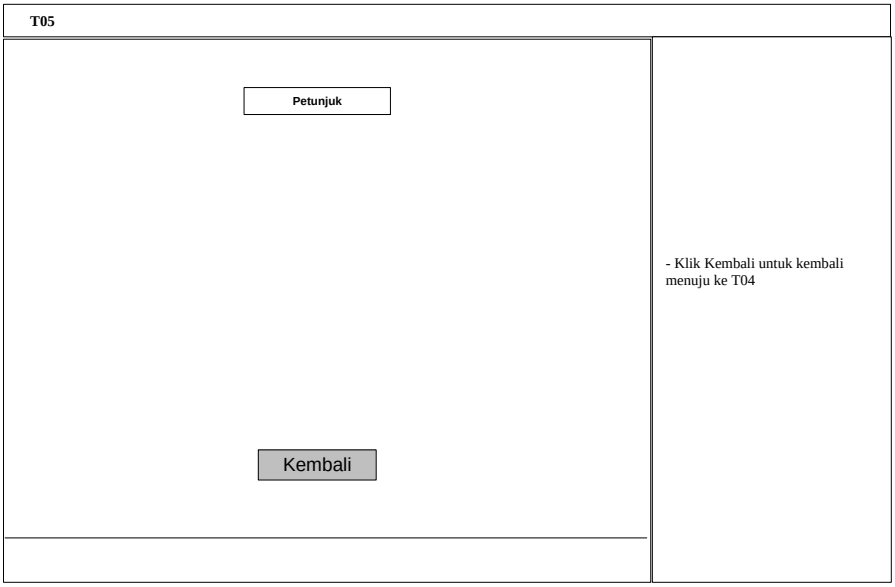
Gambar 3.94 Antarmuka *Scene* Permainan



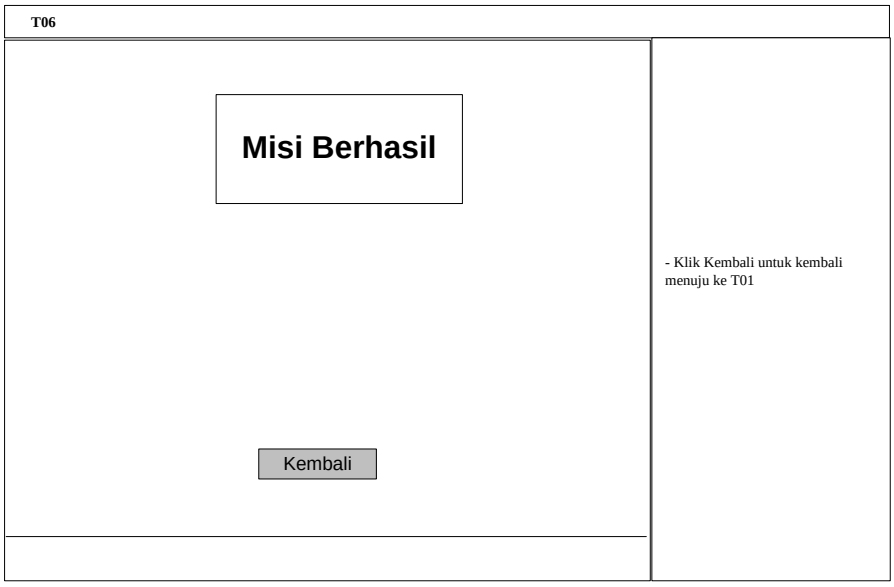
Gambar 3.95 Antarmuka Permainan



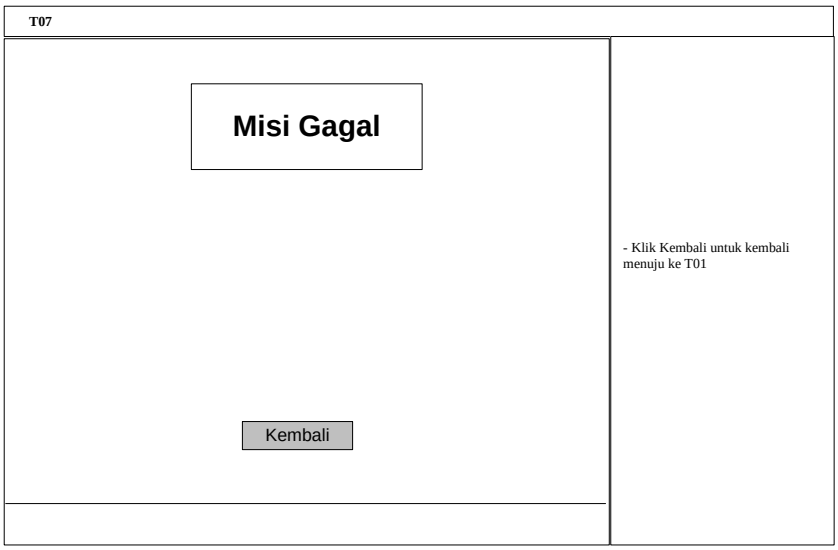
Gambar 3.96 Antarmuka *Pause Menu*



Gambar 3.97 Antarmuka Petunjuk



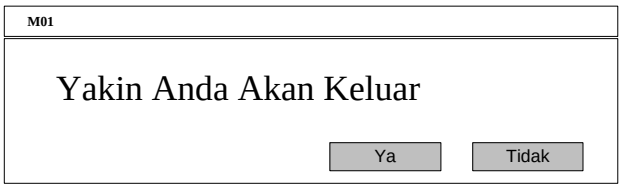
Gambar 3.98 Antarmuka tampilan misi berhasil



Gambar 3.99 Antar Muka tampilan misi gagal

3.3.2.3. Perancangan Pesan

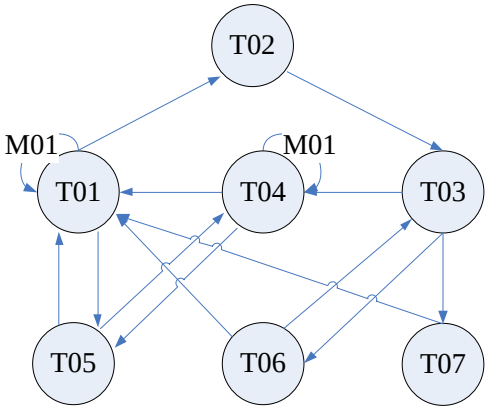
Perancangan pesan muncul sebagai informasi kepada pengguna dalam mengoperasikan sistem.



Gambar 3.100 Perancangan Pesan Keluar

3.3.3. Jaringan Semantik

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



Gambar 3.101 Jaringan Semantik