BAB 3
ANALISIS DAN PERANCANGAN SISTEM

3.1 Analisis Sistem

Pada analisis sistem akan mencakup analisis masalah, *storyline, gameplay,* analisis tingkat kesulitan, analisis algoritma, analisis, analisis kebutuhan non-fungsional dan analisis kebutuhan fungsional.

3.1.1 Analisis Masalah

Z-Virus merupakan *game* FPS yang dibuat menggunakan *Blender Game Engine* (BEG). *Game* Z-Virus ini memiliki masalah atau kelemahan berupa jalan cerita dan misi akhir *game* yang kurang jelas dan *Non-Player Character* (NPC) musuh yang hanya mempunyai *abilities* (kemampuan) untuk bisa menyerang target sampai target tersebut kalah tanpa memiliki *abilities* lain seperti bertahan ataupun melarikan diri dalam kondisi tertentu.

Game 3D FPS Alien Hunter adalah *game First Person Shooter* yang mempunyai *gameplay* hampir sama dengan Z-Virus akan tetapi memiliki intro cerita dan misi akhir *game* yang jelas. *Game* 3D FPS *Alien Hunter* mempunyai kelebihan atau fitur unik daripada game Z-Virus yang merupakan perbaikan kelemahan game Z-Virus yang telah disebutkan. Pada *game* ini NPC musuh akan diberikan AI (*artificial intelegence*) atau kecerdasan buatan yang dapat memberikan *abilities* (kemampuan) untuk melakukan beberapa opsi keputusan seperti memutuskan terus menyerang, bertahan atau melarikan diri sesuai dengan beberapa kondisi dan pertimbangan yang diberikan.

AI yang digunakan dalam pembangunan *game* ini adalah algoritma A* (A *Star*) dan algoritma *Fuzzy Logic* yang akan diterapkan kepada *Non Playable Character* (NPC). A* menerapkan teknik heuristik dalam membantu penyelesaian persoalan. Heuristik adalah penilai yang memberi harga pada tiap simpul yang memandu A* mendapatkan solusi yang diinginkan. Dengan heuristik yang benar, maka A* pasti akan mendapatkan solusi (jika memang ada solusinya) yang dicari. Dengan kata lain, heuristik adalah fungsi optimasi yang menjadikan

algoritma A* lebih baik dari pada algoritma lainnya. Algoritma ini diberikan pada NPC untuk menemukan jalur tercepat untuk mencapai posisi untuk menyerang target (player). Selain menggunakan metode A* *pathfinding* pada NPC musuh juga diterapkan metode *Fuzzy Logic* (Logika Fuzzy) untuk memberikan *abilities* dalam menentukan keputusan untuk terus menyerang, bertahan atau melarikan diri. *Fuzzy Logic* merupakan algoritma untuk pengambilan keputusan dan kontrol. Bila dibandingkan dengan logika konvensional, kelebihan *fuzzy logic* adalah kemampuannya dalam proses penalaran secara bahasa sehingga dalam perancangannya tidak memerlukan persamaan matematik yang rumit. *Fuzzy Logic* diharapkan dapat membantu mendefinisikan karakteristik operasional sistem dengan lebih baik.

3.1.2 Analisis Game Sejenis

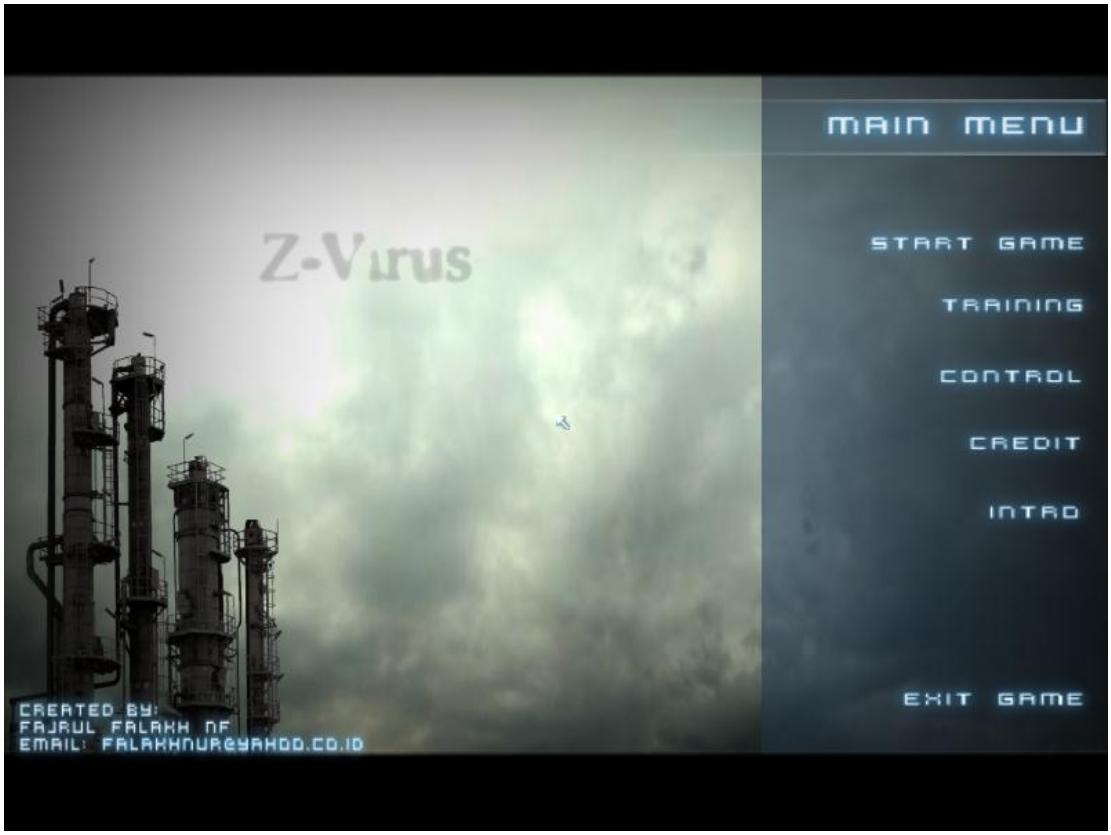
Pada analisis *game* sejenis, akan dilakukan observasi terhadap *game* yang memiliki kesamaan *gameplay* dan *genre* dengan *game* yang akan dibangun. Tujuan dari analisis ini yaitu mendapatkan konsep permainan dan perbandingan dari *game* yang telah ada.

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

a) Pengenalan *game Z-Virus*

Game Z-Virus merupakan game *bergenre First Person Shooter (FPS),* yang dibuat oleh Fajrul Falakh seorang *game maker* dari Indonesia. Game ini dibuat menggunakan *Blender Game Engine (BEG)* yang mempunyai desain 3D (3 dimensi).

Beberapa cuplikan permainan dari *game Z-Virus* dapat dilihat pada gambar 3.1 dan gambar 3.2 :



Gambar 3.1 Menu utama *game Z-Virus* [16]

Gambar 3.1 adalah tampilan menu utama dari *game Z-Virus*, dimana terdapat pilihan menu *start game* untuk memulai permainan, training untuk latihan menembak, *control* untuk info *controller*, *credit* untuk info pengembang, intro untuk melihat *splash screen intro*, dan *exit game* untuk keluar dari *game.*



Gambar 3.2 Layar Permainan *Game Z-Virus* [16]

Gambar 3.2 adalah tampilan layar permainan pada *game Z-Virus*, pada layar permainan dapat dilihat status jumlah peluru dan kondisi *Hit Point (HP)* dan juga terdapat *minimap* yg dapat dilihat pada sudut kanan atas layar permainan.

b) Gameplay game Z-Virus

Game Z-Virus ini menceritakan seorang prajurit yang harus berjuang membasmi semua makhluk hidup yang telah terjangkit virus Z. Level pertama musuh berupa laba-laba yang bergerak menyerang *player* dengan jarak dekat, pemain harus bisa sampai ke lokasi tujuan misi untuk lanjut ke level berikutnya. Pada level kedua musuh berupa makhluk bermata satu berwarna hijau yang biasa dikenal dengan *ogre*. Untuk sukses dalam level dua ini player harus berhasil sampai ke *gate* level 2.

c) Skenario *Game Z-Virus*

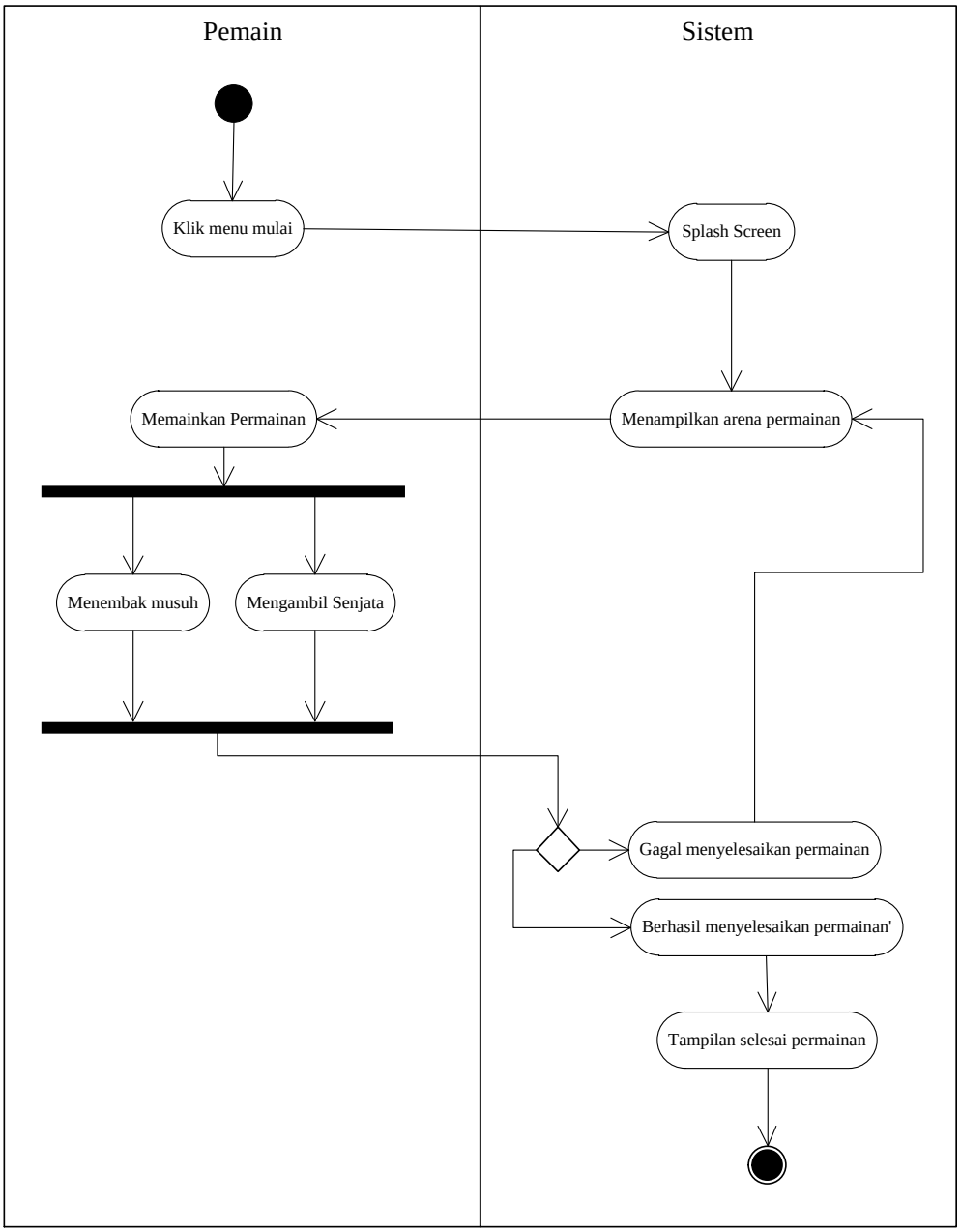
Skenario permainan dalam *game Z-Virus* disusun oleh beberapa hal sebagai berikut :

1. Alur Permaian

   Alur permainan pada *game Z-Virus* adalah sebagai berikut :

   a. Pemain berada pada gurun pasir yang sangat luas
   b. Pemain harus membunuh semua makhluk yang tertular virus Z yang menyerang mereka.
   c. Pemaian harus mampu bertahan hidup dari serangan makhluk tersebut dan harus dapat menemukan *gate* jalan keluar dari setiap level.
   d. Jika pemaian terkena serangan oleh musuh, maka darah / health point akan berkurang dan apabila habis maka permainan diulangi ke awal permainan.

Adapun alur sistem dari permainan *game Z-Virus* dapat di gambarkan melalui *activity diagram* pada Gambar 3.3:



Gambar 3.3 Alur permainan pada *game Z-Virus*

Game Z-Virus memiliki beberapa keterbatasan, yang didapatkan dari hasil review dari beberapa *game maker* pada *website* tempat game ini *dipublikasikan*, yaitu *[13]:*

1. Pada game ini keterangan tentang jalan cerita, target masing-masing level dan tujuan akhir game tidak dijelaskan didalam *game*, baik itu berupa tulisan maupun video.

2. Pada game ini NPC musuh *spawn* (muncul) ditempat yang sama, sehingga *player* dengan mudah dapat membunuh musuh dengan mengarahkan *crossfire* ketempat *spawn* musuh tersebut.
3. Pada game ini NPC musuh hanya dapat menyerang player, tidak mempunyai kemampuan untuk menentukan pilihan berupa menyerang, bertahan atau melarikan diri.

Terlepas dari beberapa keterbatasan tersebut, *game Z-Virus* memiliki kelebihan yaitu *[16]:*

1. *Game Z-Virus* memiliki *interface* (antarmuka) permainan yang bagus dan nyaman.
2. *Game Z-Virus* memiliki desain dan efek 3D yang sudah halus.

3.1.3 Storyline *Alien Hunter*

Game yang dibangun *single player* atau dengan satu pemain yang menjalankan seorang karakter utama berupa seorang *special agent* yang bernama L. Permainan dimulai dengan cerita dimana *agent* L diutus untuk mencuri reaktor nuklir yang terdapat dalam sebuah gudang pada satu kota yang telah dikuasai alien. Untuk bisa masuk kedalam gudang tersebut *Agent* L harus menghadapi prajurit alien penjaga kota bersenjata lengkap dan tower penembak mereka yang ada disekitar kota dengan menggunakan dua jenis senjata yang diberikan, berupa *machine gun* yang *unlimited ammo* (peluru tidak terbatas) dan *rocket louncher* yang *limited ammo* (peluru terbatas) hanya dua puluh rocket.

Setelah mendapatkan reaktor tersebut *agent* L kemudian harus menghancurkan markas komando alien dengan reaktor nuklir yang telah didapatkan sebelumnya. Markas komando alien ini dijaga ketat oleh prajurit-prajurit alien dan juga tower penembak mereka di sekitar markas. *Agent* L harus melewati dan mengalahkan mereka agar dapat menyelesaikan misi terakhir ini.

3.1.4 Gameplay *Alien Hunter*

Tugas utama pemain dalam memainkan *game* ini adalah menyelesaikan dua misi yang diberikan. Misi pertama yaitu pemain harus bisa masuk kedalam gudang penyimpanan alien yang berada di tengah kota serta mendapatkan reaktor

nuklir yang tersembunyi didalamnya. Misi kedua, pemain harus mampu masuk kedalam markas alien dan masuk kedalam ruang kendalinya untuk mengaktifkan reaktor nuklir yang telah didapatkan pada misi sebelumnya untuk menghancurkan markas komando alien tersebut.

Di dalam *game* ini *player* (pemain) dituntut untuk menyelesaikan misi yang ada dengan peralatan dan senjata yang telah diberikan. Didalam game juga tersedia menu *health bar* sebagai informasi *health* (darah) *player, health bar* akan berkurang jika player terkena tembakan *npc* musuh dan jika *health bar* habis, player akan mati dan otomatis misi akan gagal. Akan tetapi di sekitar map atau lokasi game juga akan dapat ditemukan *health pack* yang berguna untuk menambah *healt bar player* dan juga *rocket pack* yang berguna untuk menambah 3 *rocket* kedalam amunisi *rocket louncher.*

3.1.5 Analisis Tingkat Kesulitan

Tingkat kesulitan dari tiap *level* akan dibedakan dari jumlah dan jenis npc musuh, kecepatan, kekuatan tembakan dan *range* tembak yang lebih besar. Untuk *level* pertama NPC musuh berupa spider robot yang akan mengejar *player* dan akan meledakkan diri jika telah dekat dengan player, *white alien* dengan kecepatan, kekuatan dan *range* tembak yang kecil dan beberapa *red alien* yang mempunyai kecepatan, kekuatan dan *range* tembak yang lebih besar dan mempunyai kemampuan untuk melarikan diri dengan kecepatan gerak 1,5 kali lebih cepat dari kecepatan normal dan menambah darah 10 HP per detik pada *healing point* dan juga pada level ini juga terdapat beberapa *sentrygun* (tower penembak) yang akan menembak player jika berada dalam jarak tembak. Pada level kedua NPC berupa spider robot yang akan mengejar *player* dan akan meledakkan diri jika telah dekat dengan player, *white alien* dengan kecepatan, kekuatan dan *range* tembak yang kecil dan lebih banyak *red alien* yang mempunyai kecepatan, kekuatan dan *range* tembak yang besar dan mempunyai kemampuan untuk melarikan diri dengan kecepatan gerak 2 kali lebih cepat dari kecepatan normal dan menambah darah 15 HP per detik pada *healing point* dan juga terdapat banyak *sentrygun* yang akan menembak player jika berada dalam

jarak tembak. Pada level kedua ini alien dan *sentrygun* (tower penembak) memiliki jumlah yang jauh lebih banyak dari level pertama.

3.1.6 Analisis Scoring

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

3.1.7 Analisis Algoritma

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

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

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

(0.0)	(0.1)	(0.2)	(0.3)	(0.4)	(0.5)	(0.6)
(1.0)	(1.1) A	(1.2)	(1.3)	(1.4)	(1.5) B	(1.6)
(2.0)	(2.1)	(2.2)	(2.3)	(2.4)	(2.5)	(2.6)
F (3.0) G H	(3.1)	(3.2)	(3.3)	(3.4)	(3.5)	(3.6)

Gambar 3.4 Ilustrasi A*

Pada gambar 3.4 diasumsikan bahwa wilayah pencarian dibagi-bagi menjadi kotak-kotak atau disebut *node*, *node* A yang berwarna merah (1.1) merupakan *node* awal dan *node* B yang berwarna hijau (1.5) merupakan *node* tujuan, sedangkan *node* yang berwarna biru (0.3), (0.4), (1.3), (2.3), (2.4) merupakan dinding penghalang atau disebut *collision*. Diasumsikan pencarian jalan terbaik dimulai dari *node* A menuju *node* B dengan *node* biru sebagai *collision*nya.

Nilai F yang akan dihitung untuk setiap *node* terletak disudut kiri atas kotak, nilai G terletak di sudut kiri bawah kotak dan nilai H terletak disudut kanan bawah kotak. Perhitungan nilai F didapat dari persamaan F= G+H. Sedangkan untuk nilai G digunakan nilai 10 untuk gerakan *vertical* atau *horizontal*, angka 10 digunakan hanya untuk memudahkan perhitungan. Sedangkan untuk gerakan *diagonal* maka akan dikalikan dengan √2 yang berarti 10x√2 = 14,14. Untuk memudahkan perhitungan angka 14,14 dibulatkan menjadi 14.

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

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

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

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

1. Pertama, dimulai dari *node* A dan tambahkan ke dalam *open list. Open list* merupakan *list* untuk *node-node* yang harus diperiksa.
2. Ambil *node* A sebagai *current node*, dan *set current node = best node*. Atau dalam artian *node* A merupakan bagian dari jalan terpendek yang sudah didapatkan
3. Periksa apakah *current node* sama dengan *node* B, atau *current node* yang sudah didapatkan merupakan *node* tujuan. Jika iya maka jalan sudah ditemukan, jika tidak maka hapus *node* A dari *open list*, dan tambahkan ke dalam *closed list.*
4. Periksa semua *neighbor node* atau *node* yang berdekatan dengan *node* A apakah *walkable* atau tidak, abaikan *node* dengan *collision* atau penghalang lalu tambahkan semua *node* tersebut ke dalam *open list.* Dan untuk

setiap *node* yang baru ditambahkan, simpan *node* A sebagai "*parent*" yang nantinya akan digunakan untuk menelusuri jalan.

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

Iterasi Pertama:

74 (0.0) 14 ↘ 60	60 (0.1) 10 ↓ 50	54 (0.2) 14 ↙ 50	(0.3)	(0.4)	(0.5)	(0.6)
60 (1.0) 10 → 50	40 (1.1) 0 40	40 (1.2) 10 ← 30	(1.3)	(1.4)	(1.5)	(1.6)
74 (2.0) 14 ↗ 60	60 (2.1) 10 ↑ 50	54 (2.2) 14 ↖ 40	(2.3)	(2.4)	(2.5)	(2.6)
F (3.0) G H	(3.1)	(3.2)	(3.3)	(3.4)	(3.5)	(3.6)

Gambar 3.5 Ilustrasi Iterasi Pertama

Pada gambar 3.5 dapat dilihat iterasi pertama *node* (1,1) dimasukkan ke dalam *open list*, dan selanjutnya memeriksa *neighbor node* atau *node* yang berdekatan dengan *node* (1,1) yaitu *node* {(0,0), (0,1), (0,2), (1,0), (1,2), (2,0), (2,1), (2,2)} dan dimasukkan ke dalam *open list.* Hapus *node* (1,1) dari *open list* dan masukkan ke dalam *closed list.* Ambil *node* dengan nilai F paling kecil sebagai *current node* berikutnya dalam perhitungan diatas didapat *node* (1,2) memiliki nilai F paling rendah dan diberikan tanda dengan kotak berwarna abu-abu.

Node yang menjadi *current node* atau *node* (1,2) diberikan nilai H = 30 yang didapatkan dari perhitungan 3 langkah atau 3 *node* menuju *node* B dan dikalikan 10, 3*10 = 30.

Iterasi pertama menghasilkan *node-node* di dalam *open list* dan *closed list* dijelaskan dalam tabel 3.1.

Tabel 3.1 Perhitungan iterasi pertama

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

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

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

Selanjutnya perhitungan pada iterasi kedua dengan *current node* berada pada *node* (1,2).

74 (0.0) 14 ↘ 60	60 (0.1) 10 ↓ 50	64 (0.2) 20 ↓ 40	(0.3)	(0.4)	(0.5)	(0.6)
60 (1.0) 10 → 50	40 (1.1) 0 40	40 (1.2) 10 ← 30	(1.3)	(1.4)	(1.5)	(1.6)
74 (2.0) 14 ↗ 60	74 (2.1) 24 ↗ 50	64 (2.2) 24 ↑ 40	(2.3)	(2.4)	(2.5)	(2.6)
F (3.0) G H	(3.1)	(3.2)	(3.3)	(3.4)	(3.5)	(3.6)

Gambar 3.6 Ilustrasi Iterasi Kedua

Periksa *neighbor node* atau *node* yang berdekatan dengan *current node*, disebelah kanan, kanan bawah dan kanan atas dari *node* ini adalah *node* penghalang atau *unwalkable* (gambar 3.6) berarti diabaikan, dan disebelah kanan adalah *node* (1,1) yang sudah berada di dalam *closed list*, yang berarti juga diabaikan.

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

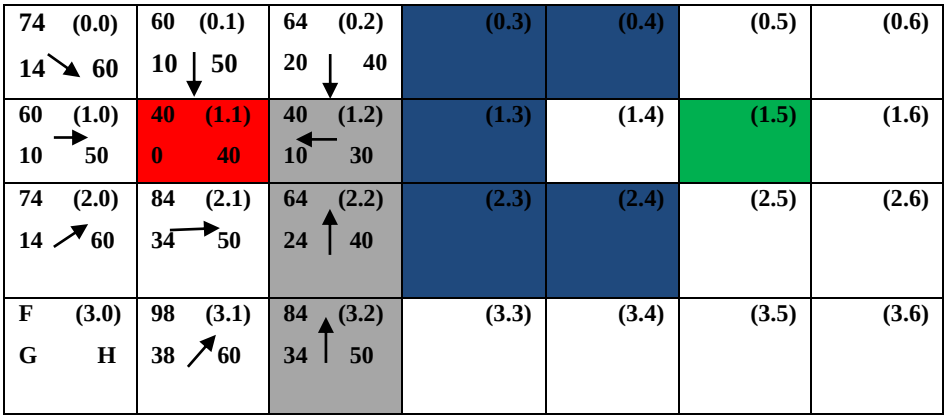
Node (0,2) dan *node* (2,2) memiliki nilai F yang sama, dapat dilihat pada gambar 3.6 diatas, maka sistem akan memilih *node* (2,2) karena yang terakhir masuk ke dalam *open list*. Yang berarti *current node* selanjutnya adalah *node* (2,2).

Iterasi kedua menghasilkan *node-node* di dalam *open list* dan *closed list* dijelaskan dalam tabel 3.2.

Tabel 3.2 Perhitungan iterasi kedua

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

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



Gambar 3.7 Ilustrasi Iterasi Ketiga

Pada gambar 3.7 *node* (3,3) tidak dihitung atau tidak masuk ke dalam *open list*, karena karakter tidak akan bisa bergerak secara *diagonal* menembus dinding atau *collision* sehingga *node* (3,3) otomatis menjadi *unwalkable*.

Looping lagi seperti langkah sebelumnya, hapus dari *open list* dan tambahkan ke dalam *closed list*, periksa semua *node* yang berdekatan, masukkan ke dalam *open list* dan tambahkan *parent*nya, dan periksa lagi jalan terbaik dan *parent*nya. Pada *node* (2,1) *parent*nya mengalami perubahan karena setelah diperiksa nilai G yang dihasilkan dari *current node* lebih kecil, maka nilai F nya juga dirubah.

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

Iterasi ketiga menghasilkan *node-node* di dalam *open list* dan *closed list* dijelaskan dalam tabel 3.3.

Tabel 3.3 Perhitungan iterasi ketiga

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

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

74 (0.0) 14 ↘ 60	60 (0.1) 10 ↓ 50	64 (0.2) 20 ↓ 40	(0.3)	(0.4)	(0.5)	(0.6)
60 (1.0) 10 → 50	40 (1.1) 0 40	40 (1.2) 10 ← 30	(1.3)	(1.4)	(1.5)	(1.6)
74 (2.0) 14 ↗ 60	94 (2.1) 44 ↘ 50	64 (2.2) 24 ↑ 40	(2.3)	(2.4)	(2.5)	(2.6)
F (3.0) G H	104 (3.1) 44 → 60	84 (3.2) 34 ↑ 50	84 (3.3) 44 ↖ 40	(3.4)	(3.5)	(3.6)

Gambar 3.8 Ilustrasi Iterasi Keempat

Iterasi keempat menghasilkan *node-node* di dalam *open list* dan *closed list* dijelaskan dalam tabel 3.4

Tabel 3.4 Perhitungan iterasi keempat

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

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

74 (0.0) 14 ↘ 60	60 (0.1) 10 ↓ 50	64 (0.2) 20 ↓ 40	(0.3)	(0.4)	(0.5)	(0.6)
60 (1.0) 10 → 50	40 (1.1) 0 40	40 (1.2) 10 ← 30	(1.3)	(1.4)	(1.5)	(1.6)
74 (2.0) 14 ↗ 60	94 (2.1) 44 ↘ 50	64 (2.2) 24 ↑ 40	(2.3)	(2.4)	(2.5)	(2.6)
F (3.0) G H	104 (3.1) 44 → 60	84 (3.2) 34 ↑ 50	84 (3.3) 44 ↖ 40	84 (3.4) 54 ← 30	(3.5)	(3.6)

Gambar 3.9 Ilustrasi Iterasi Kelima

Iterasi kelima menghasilkan *node-node* di dalam *open list* dan *closed list* dijelaskan dalam tabel 3.5.

Tabel 3.5 Perhitungan iterasi kelima

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

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

74 (0.0) 14 ↘ 60	60 (0.1) 10 ↓ 50	64 (0.2) 20 ↓ 40	(0.3)	(0.4)	(0.5)	(0.6)
60 (1.0) 10 → 50	40 (1.1) 0 40	40 (1.2) 10 ← 30	(1.3)	(1.4)	(1.5)	(1.6)
74 (2.0) 14 ↗ 60	94 (2.1) 44 ↘ 50	64 (2.2) 24 ↑ 40	(2.3)	(2.4)	(2.5)	(2.6)
F (3.0) G H	104 (3.1) 44 → 60	84 (3.2) 34 ↑ 50	84 (3.3) 44 ↖ 40	84 (3.4) 54 ← 30	84 (3.5) 64 ← 20	(3.6)

Gambar 3.10 Ilustrasi Iterasi Keenam

Node (2,5) menjadi *unwalkable* karena karakter tidak bisa langsung bergerak ke *node* tersebut, sehingga tidak masuk ke dalam *open list*.

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

Tabel 3.6 Perhitungan iterasi keenam

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

Perhitungan pada iterasi ketujuh dengan *current node* berada pada *node* (3,5).

74 (0.0) 14 → 60	60 (0.1) 10 → 50	64 (0.2) 20 → 40	(0.3)	(0.4)	(0.5)	(0.6)
60 (1.0) 10 → 50	40 (1.1) 0 40	40 (1.2) 10 ← 30	(1.3)	(1.4)	(1.5)	(1.6)
74 (2.0) 14 → 60	94 (2.1) 44 → 50	64 (2.2) 24 → 40	(2.3)	(2.4)	84 (2.5) 74 → 10	98 (2.6) 78 → 20
F (3.0) G H	104 (3.1) 44 → 60	84 (3.2) 34 → 50	84 (3.3) 44 → 40	84 (3.4) 54 → 30	84 (3.5) 64 → 20	104 (3.6) 74 → 30

Gambar 3.11 Ilustrasi Iterasi Ketujuh

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

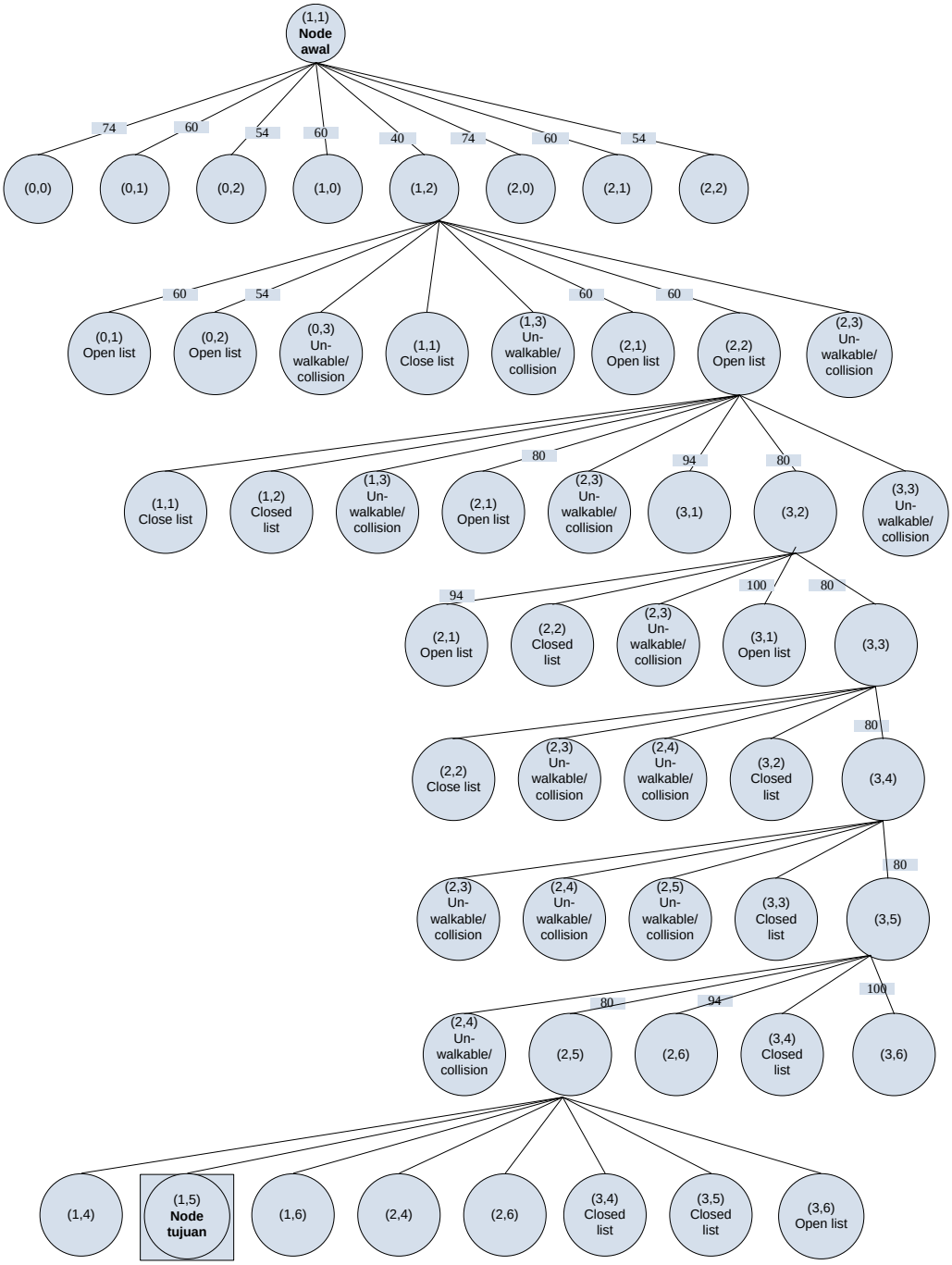
Tabel 3.7 Perhitungan iterasi ketujuh

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

Pada iterasi kedelapan sudah dapat dipastikan *target point* masuk ke dalam *closed list* (gambar 3.11) maka proses perhitungan selesai.

Penentuan jalan terbaik dengan melakukan pemeriksaan dimulai dari *target point* atau *node* (1,5) dengan mengikuti arah *pointer* yang berarti akan membawanya kembali ke *node* awal, maka akan didapatkan jalan terbaiknya.

Langkah-langkah perhitungan dari contoh diatas dapat diilustrasikan dalam *tree* sebagai berikut.



Gambar 3.12 *Tree* ilustrasi perhitungan A*

2. Analisis *Fuzzy Logic*

Fuzzy Logic (logika fuzzy) di dalam *game* akan diterapkan pada NPC (*Non Playable Character*) untuk memberikan *abilities* dalam menentukan keputusan untuk menyerang, bertahan dan melarikan diri. Cara kerja *Fuzzy Logic* di dalam *game Alien Hunter* lebih jelasnya sebagai berikut.

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

1. Variabel Fuzzy

Ada 3 variabel yang digunakan dalam fungsi fuzzy, yaitu sebagai berikut:

1. *Distance* atau jarak antara *player* dengan agen (npc musuh)
2. *HP (Hit Point),* merupakan point darah (kesehatan) dari agen
3. *Decision Maker,* keputusan yang akan diambil oleh agen

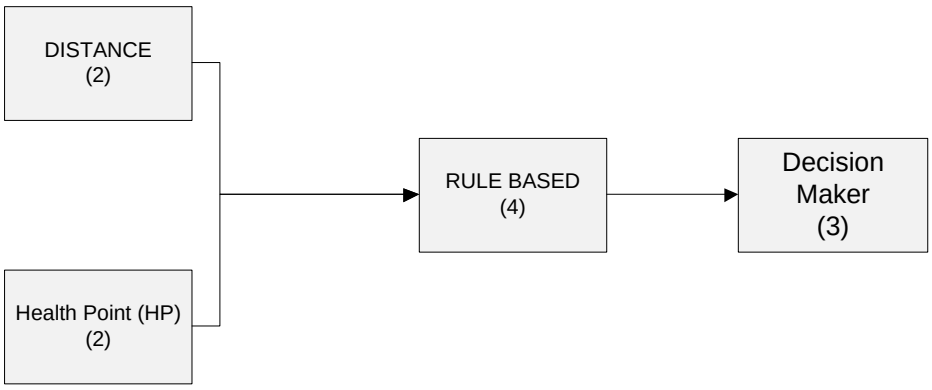
2. Nilai linguistik / terma

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

1. *Distance* : *Near, Far*
2. *HP (Hit Point*) : *Injured, Fit*
3. *Decision Maker* : *Escape, Defend, Attack*

3. Fuzzyfikasi

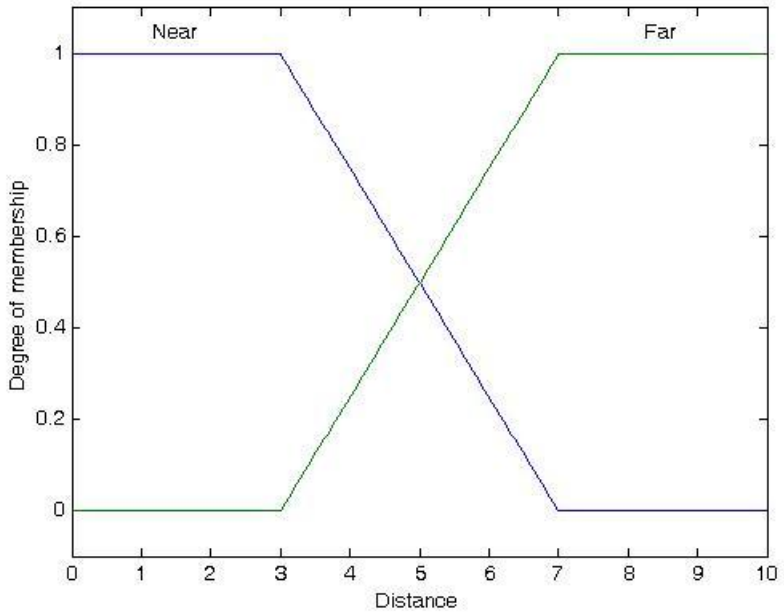
Fuzzyfikasi merupakan proses memetakan nilai *crisp* (numerik) ke dalam himpunan *fuzzy* dan menentukan derajat keanggotaannya. Secara garis besar pemetaan nilai *crisp* ke dalam himpunan *fuzzy* dijelaskan dengan gambar 3.13.



Gambar 3.13 *Fuzzy Interface System*

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

1. Variabel Distance, terbagi menjadi beberapa input yaitu : near, average dan far. Range nilai untuk variabel *distance* diantara 0-10 dengan penjelasan setiap *terma* sebagai berikut
   a) *Terma Near* = 0 – 7
   b) *Terma Far* = 3 – 10



Gambar 3.14 Grafik variabel *Distance*

Perhitugan *Fuzzyfikasi* variabel *Distance*:

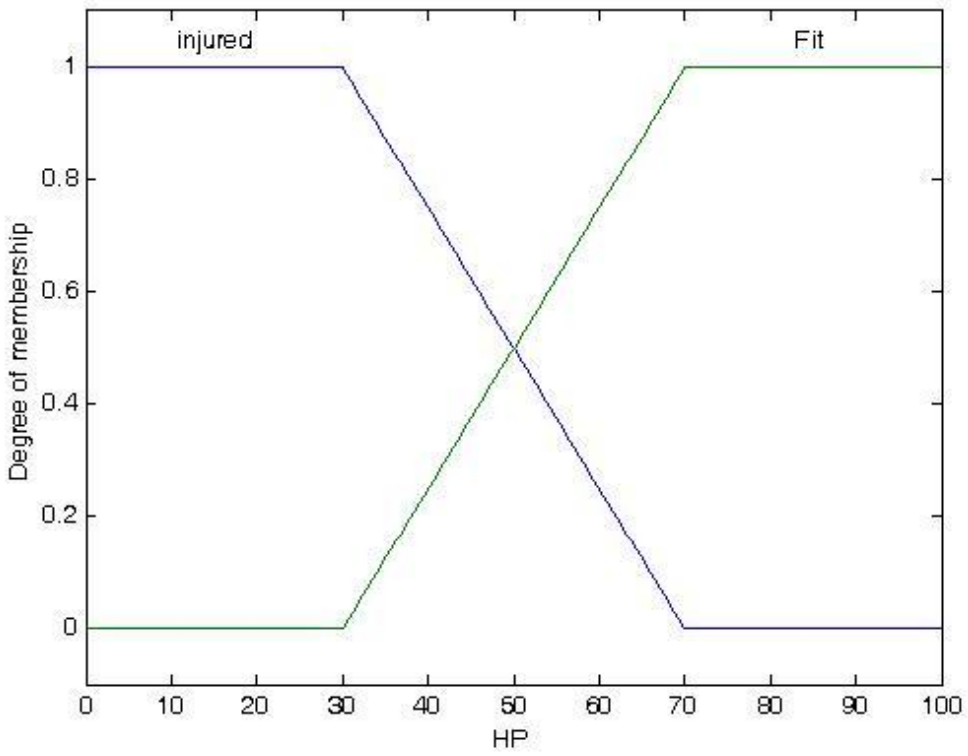
$$\mu DisNear[x] = \begin{cases} 1; x \le 3 \\ \dfrac{7-x}{4}; 3 \le x \le 7 \\ 0; x \ge 7 \end{cases}$$

$$\mu DisFar[x] = \begin{cases} 0; x \le 3 \\ \dfrac{x-3}{4}; 3 \le x \le 7 \\ 1; x \ge 7 \end{cases}$$

μi = derajat keanggotaan masing-masing variabel

2. Variabel *Health Point* (HP), terbagi menjadi beberapa input yaitu : injured dan fit. Range nilai variabel HP diantara 0 – 10 dengan penjelasan *terma* sebagai berikut:

    a) *Terma Injured* = 0 – 70
    b) *Terma Fit* = 30 – 100



Gambar 3.15 Grafik variabel HP

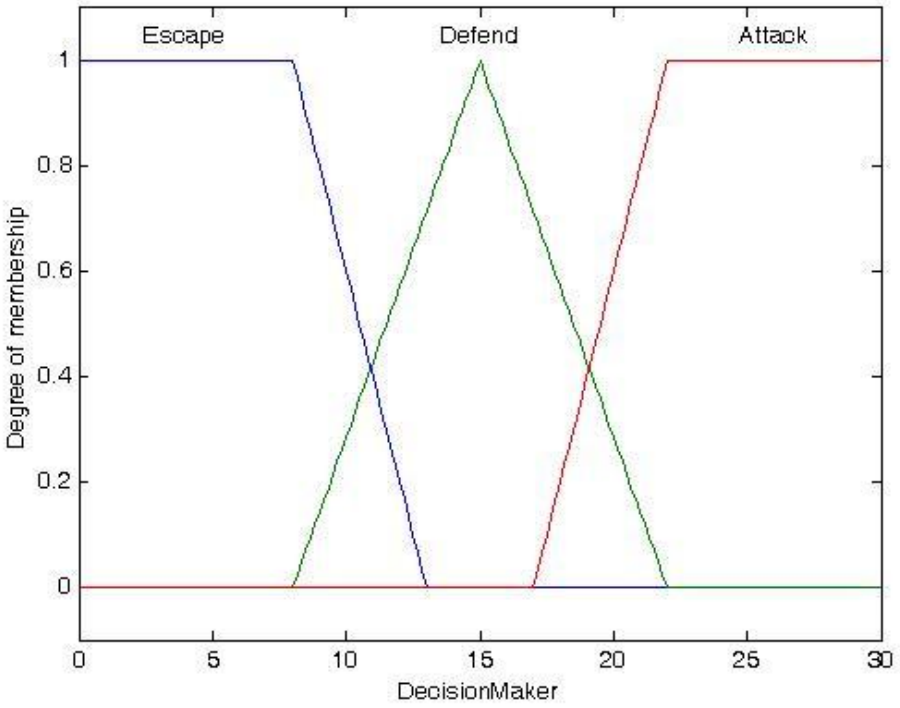
Perhitungan *Fuzzyfikasi* variabel HP:

$$\mu HPInjured[y] = \begin{cases} 1; y \le 30 \\ \dfrac{70-y}{40}; 30 \le y \le 70 \\ 0; y \ge 70 \end{cases}$$

$$\mu HPFit[y] = \begin{cases} 0; y \le 30 \\ \dfrac{y-30}{40}; 30 \le y \le 70 \\ 1; y \ge 70 \end{cases}$$

3. Variabel *Decision maker*, yaitu menyerang, bertahan dan melarikan diri. Range nilai variabel *Desicion Maker* diantara 0-30 dengan penjelasan *terma* sebagai berikut:

   a) Terma Escape = 0 – 13
   b) Terma Defend = 8 – 22
   c) Terma Attack = 17- 30



Gambar 3.16 Grafik output Decisionmaker

Perhitungan fuzzyfikasi Output Decisionmaker

$$\mu DEscape[z] = \begin{cases} 1; z \le 8 \\ \dfrac{13-z}{5}; 8 \le z \le 13 \\ 0; z \ge 13 \end{cases}$$

$$\mu DDefend[z] = \begin{cases} 0; z \le 8 / z \ge 22 \\ \dfrac{z-8}{7}; 8 \le z \le 15 \\ \dfrac{15-z}{7}; 15 \le z \le 22 \end{cases}$$

$$\mu DAttack[z] = \begin{cases} 0; z \le 17 \\ \dfrac{z-17}{5}; 17 \le z \le 22 \\ 1; z \ge 22 \end{cases}$$

4. Kaidah fuzzy atau *Rule Based*

Kaidah *fuzzy (rules)* atau aturan-aturan yang diterapkan dalam penentuan perilaku Decision Maker ada enam *rules* yaitu:

Tabel 3.8 *Rule-Based* tindakan agen

	DISTANCE		**HP**		**Decision Maker**
IF	Near	AND	Injured	THEN	**Escape**
IF	Near	AND	Fit	THEN	**Attack**
IF	Far	AND	Injured	THEN	**Defend**
IF	Far	AND	Fit	THEN	**Attack**

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)

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

z*=titik pusat nilai *crisp*

μi =derajat keanggotaan masing-masing variabel

zi = nilai domain pada variabel linguistik z

Contoh Penyelesaian Kasus

Misalkan variabel distance diberikan nilai *crisp* sebesar 6, dan variabel hp diberikan nilai sebesar 40, maka proses agregasi yang menentukan keluaran sesuai dengan *rules* yang sudah dibuat akan dijelaskan seperti berikut.

[Distance= 6 HP=40]

Fuzzyfikasi:

$$\mu DisNear[6] = \frac{7-6}{4} = 0.25$$

$$\mu DisFar[6] = \frac{6-3}{4} = 0.75$$

$$\mu HPInjured[40] = \frac{70-40}{40} = 0.75$$

$$\mu HPFit[40] = \frac{40-30}{40} = 0.25$$

Operasi Logika Fuzzy dan Implikasi

a) Rule 1 : IF Distance is *Near* and HP is *Injured* Then *Escape*

→ min(0.25 , 0.75) = 0.25

b) Rule 2 : IF Distance is *Near* and HP is *Fit* Then *Attack*

→ min(0.25 , 0,25) = 0.25

c) Rule 3 : IF Distance is *Far* and HP is *Injured* Then *Defend*

→ min(0.75 , 0.75) = 0.75

d) Rule 4 : IF Distance is *Far* and HP is *Fit* Then *Attack*

→ min(0.75 , 0.25) = 0.25

Defuzzy

a) Rule 1: $\frac{13-z}{5}=0.25$

$z=13-1.25=11.75$

b) Rule 2: $\frac{z-17}{5}=0.25$

$z=1.25+17=18.25$

c) Rule 3: $\frac{z-8}{7}=0.75$

$z=5.25+8=13.25$

d) Rule 4: $\frac{z-17}{5}=0.25$

$z=1.25+17=18.25$

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

➔ $$z^*=\frac{0.25\times11.75+0.25\times18.75+0.75\times13.25+0.25\times18.25}{0.25+0.25+0.75+0.25}$$

➔ $$z^*=\frac{2.9+4.6+9.9+4.6}{0.25+0.25+0.75+0.25}$$

➔ $$z^*=\frac{22}{1.5}=14,66$$

Jadi keputusan output yang diambil pada Decisionmaker bernilai 14,66 (*Defend*) sesuai dengan grafik decisionmaker dan sesuai dengan Rules 3.

Dengan Distance = 6 dan HP = 40 maka decisionmaker yang diambil adalah 14,66 (Defend)

3.1.8 Analisis User

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

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

   Game alien hunter ini bisa digunakan oleh kalangan apapun, tetapi pengetahuan dan pengalaman akan memudahkan *user* dalam penggunaanya. Terutama pengetahuan dan pengalaman dalam memainkan

game action atau *first person shooter*. Berikut ini klasifikasi *knowledge and experience* dari pengguna applikasi:

Tabel 3.9 Analisis klasifikasi *knowledge and experience*

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

2. Users Physical Characteristic

Keadaan fisik seseorang mungkin akan berpengaruh pada penggunaan aplikasi *game* ini. Ada hal-hal yang harus diperhatikan juga

terhadap user dari karakteristik fisiknya untuk dapat menggunakan aplikasi ini yaitu, Color Blind, Handednes, dan Gender.

Tabel 3.10 Analisis *Users Physical Characteristic*

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

3.1.9 Analisis Kebutuhan Non-Fungsional

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

3.1.9.1 Analisis Kebutuhan Perangkat Lunak

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

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

1. Sistem operasi Windows Seven.
2. Unity versi 3.5.0
3. 3ds Max 2011
4. Graphic card
5. DirectX 9

Perangkat lunak yang dibutuhkan pengguna untuk memainkan game ini adalah :

1. Sistem operasi minimal Windows Xp
2. Graphic card
3. DirectX 9

3.1.9.2 Analisis Kebutuhan Perangkat Keras

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

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

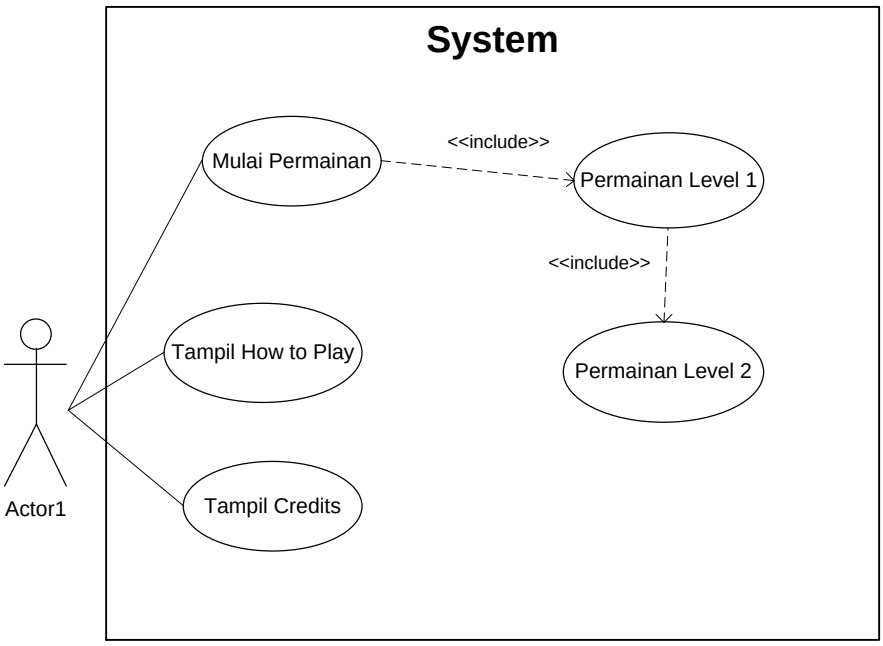
1) Processor Dual Core 1,80 Ghz
2) RAM 1 GB
3) VGA 512 MB
4) Harddisk kosong tersedia 300 MB
5) Mouse dan Keyboard

3.1.10 Analisis Kebutuhan Fungsional

Tahap pertama yang dilakukan dalam melakukan analisis berorientasi objek menggunakan UML adalah menentukan aktor atau pengguna sistem. Kata aktor dalam konteks UML, menampilkan peran (*rules*) pengguna atau sesuatu diluar sistem yang dikembangkan yang dapat berupa perangkat keras, end user, sistem yang lain , dan sebagainya [17].

1. Use Case Diagram

Use case diagram merupakan pemodelan untuk mendeskripsikan interaksi antara satu atau lebih aktor dengan sistem yang akan dibuat. Berikut ini adalah perancangan proses-proses yang terdapat pada *game FPS 3D Alien Hunter* :



Gambar 3.17 *Use Case Diagram*

1.1 *Definisi Actor*

Definisi *Actor* berfungsi untuk menjelaskan *actor* yang terdapat pada *use case diagram*. Definisi *actor* diterangkan pada tabel 3.11.

Tabel 3.11 *Definisi Actor*

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

1.2 *Definisi Use Case*

Definisi *use case* berfungsi untuk menjelaskan fungsi *use case* yang terdapat pada *use case diagram*. Definisi *use case* diterangkan pada tabel 3.12.

Tabel 3.12 *Definisi Use Case*

No.	*Use Case*	Deskripsi
1.	*Mulai Permainan*	Proses untuk memulai *game.*
2.	*Tampil How To Play*	Proses untuk menampilkan petunjuk dan panduan bermain.
3.	*Tampil Credits*	Proses untuk melihat informasi pembuat *game*.
4.	*Permainan Level 1*	Permainan Misi 1 (*level* 1).
5.	*Permainan Level 2*	Permainan Misi 2 (*level* 2).

1.3 *Scenario Use Case*

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

Tabel 3.13 *Scenario Use Case* Mulai Permainan

Identifikasi		
Nomor	1	
Nama	*Mulai Permainan*	
Tujuan	*Memulai Permainan*	
Deskripsi	Proses untuk memulai *game*	
Aktor	Pengguna	
Skenario Utama		
Kondisi awal	Pengguna berada di *menu* utama	
Aksi Aktor		Reaksi Sistem
Memilih menu *Play*		*Masuk Intro scene dan Start game level* 1
Kondisi Akhir	*Intro Scene* dan *Level* 1 dimainkan	

Tabel 3.14 *Scenario Use Case Tampil How To Play*

Identifikasi		
Nomor	2	
Nama	Tampil *How To Play*	
Tujuan	Menampilkan petunjuk dan panduan bermain	
Deskripsi	Proses untuk menampilkan petunjuk dan panduan bermain	
Aktor	Pengguna	
Skenario Utama		
Kondisi awal	Pengguna berada di *menu* utama	
Aksi Aktor		Reaksi Sistem
Memilih menu *how to play*		Menampilkan informasi petunjuk bermain
Kondisi Akhir	Menampilkan petunjuk dan panduan bermain	

Tabel 3.15 *Scenario Use Case Tampilan Credits*

Identifikasi		
Nomor	3	
Nama	*Tampil Credits*	
Tujuan	Menampilkan informasi pembuat *game*	
Deskripsi	Proses untuk melihat informasi pembuat *game.*	
Aktor	Pengguna	
Skenario Utama		
Kondisi awal	Pengguna berada di *menu* utama	
Aksi Aktor		Reaksi Sistem
Memilih menu *credits*		Menampilkan informasi pembuat *game.*
Kondisi Akhir	Menampilkan informasi pembuat *game.*	

Tabel 3.16 *Scenario Use Case Permainan Level* 1

Identifikasi		
Nomor	7	
Nama	*Permainan Level* 1	
Tujuan	Menampilkan *level* 1 permainan	
Deskripsi	Tingkatan *level* 1(Misi 1).	
Aktor	Pengguna	
Skenario Utama		
Kondisi awal	Pengguna berada di *menu* utama	
Aksi Aktor		Reaksi Sistem
Memilih menu *Play*		Menampilkan *level* 1
Kondisi Akhir	Menampilkan *level* 1	

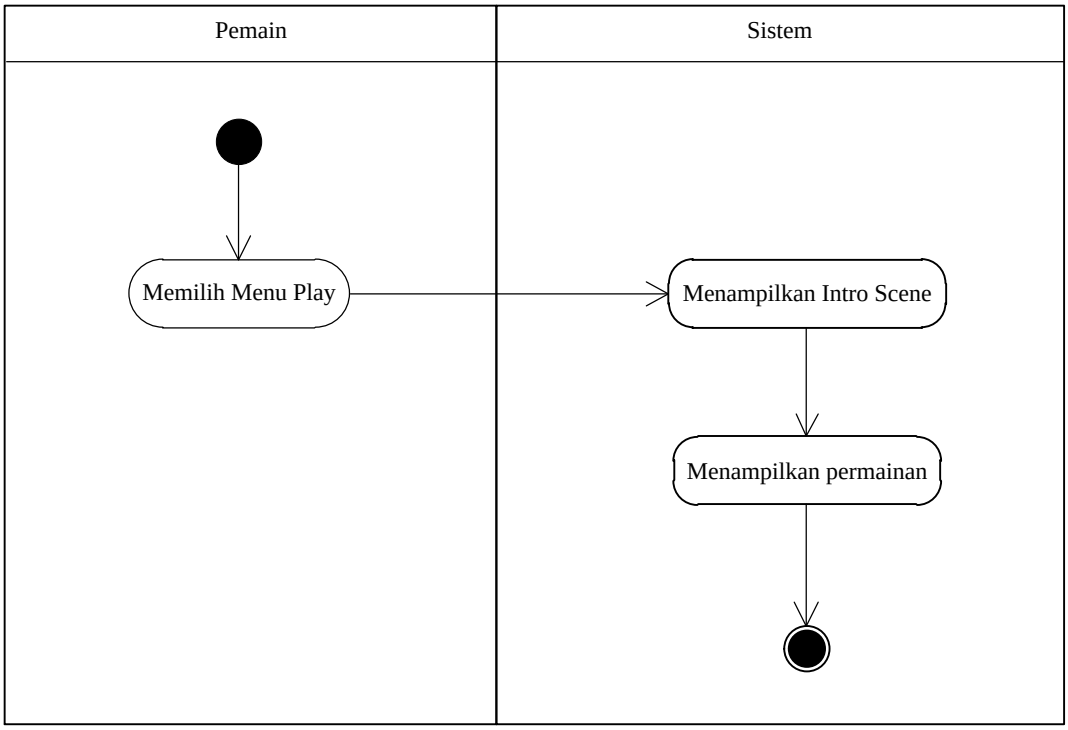
Tabel 3.17 *Scenario Use Case Permainan Level* 2

Identifikasi		
Nomor	8	
Nama	*Permainan Level* 2	
Tujuan	Menampilkan *level* 2 (Misi 2)	
Deskripsi	Tingkatan *level* 2	
Aktor	Pengguna	
Skenario Utama		
Kondisi awal	Pengguna memainkan *game*	
Aksi Aktor		Reaksi Sistem
Pengguna menyelesaikan *level* 1 (misi 1 success)		Menampilkan Permainan *level* 2 (Misi 2)
Kondisi Akhir	Menampilkan *level* 2	

2. Activity Diagram

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

A. *Activity Diagram* mulai permainan

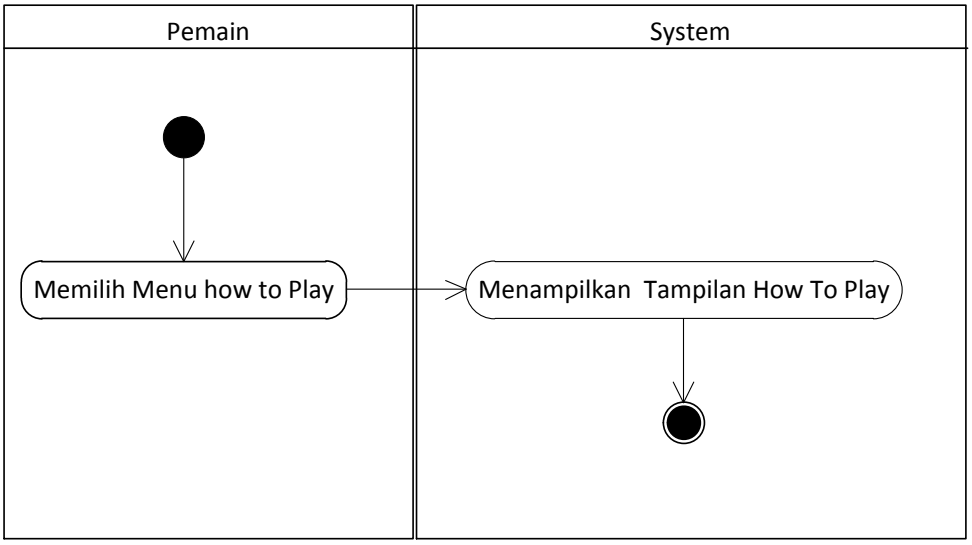


Gambar 3.18 *Activity diagram* Proses Mulai Permainan

Tabel 3.18 Penjelasan Proses Mulai permainan

Deskripsi	Menu *Play* merupakan menu yang digunakan untuk mulai memainkan permainan.
Normal Flow	1. Pemain memilih menu *play* di menu utama 2. Sistem menampilkan *scene* atau cerita sebelum masuk ke permainan 3. Sistem menampilkan permainan

B. *Activity Diagram* Proses Tampilan *How to Play*

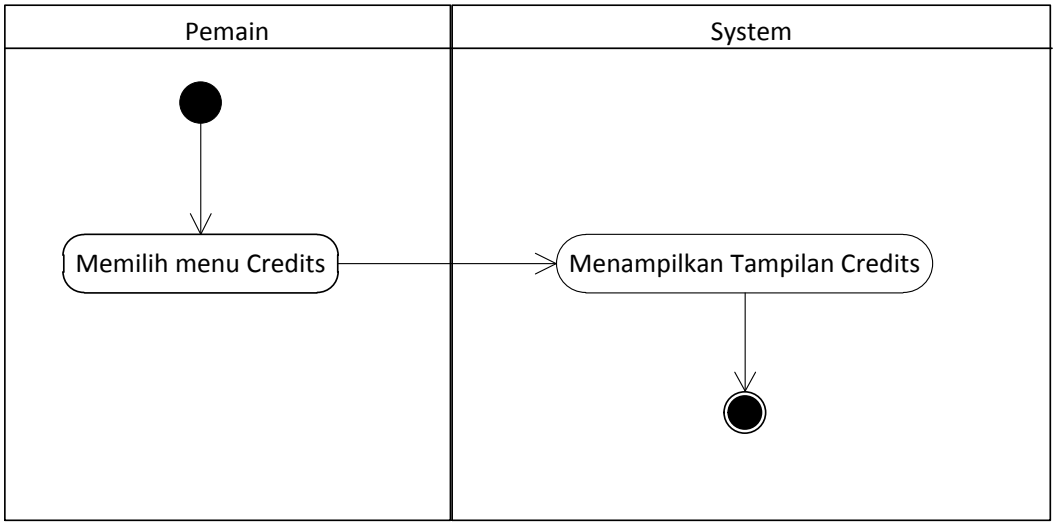


Gambar 3.19 *Activity diagram* Proses Tampilan *How to Play*

Tabel 3.19 Penjelasan Tampilan *How to Play*

Deskripsi	Tampilan ini menampilkan start menu setelah menjalankan game, pemain diberikan beberapa sub pilihan.
Normal Flow	1. Pemain menjalankan game 2. Pada start menu pemain memilih menu how to play untuk melihat tampilan *how to play*.

C. *Activity Diagram* Proses Tampilan *Credits*

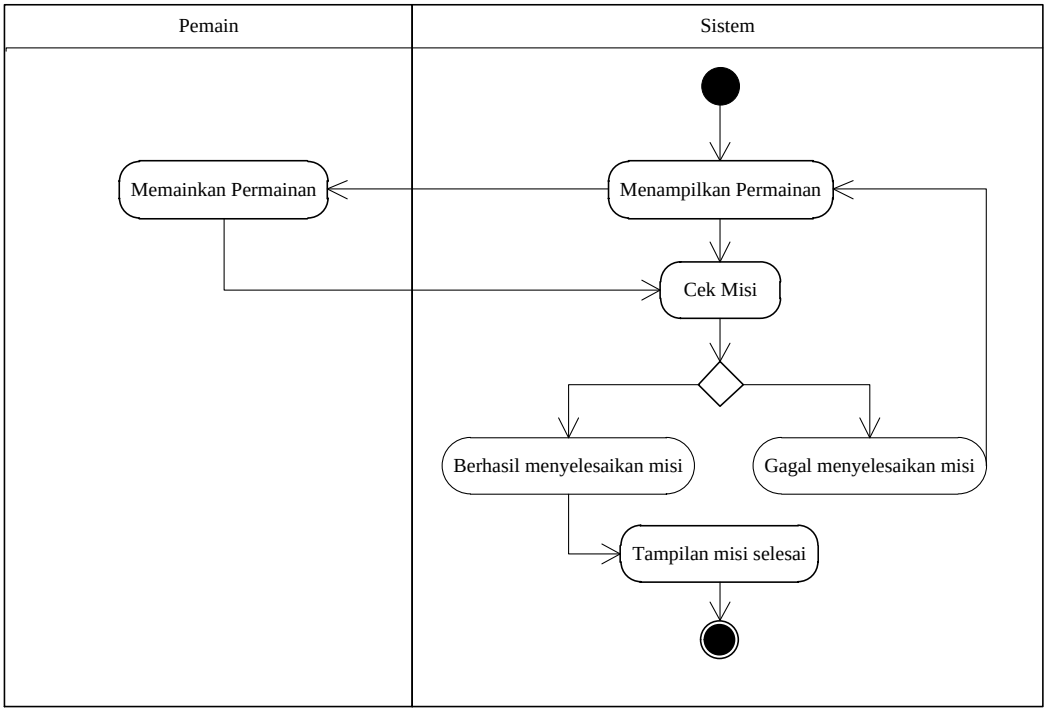


Gambar 3.20 *Activity diagram* Proses Credits

Tabel 3.20 Penjelasan Tampilan *Credits*

Deskripsi	Tampilan ini menampilkan start menu setelah menjalankan *game*, pemain diberikan beberapa sub pilihan.
Normal Flow	1. Pemain menjalankan game 2. Pada start menu pemain memilih menu *Credits* untuk melihat tampilan *Credits*

D. *Activity Diagram* permainan

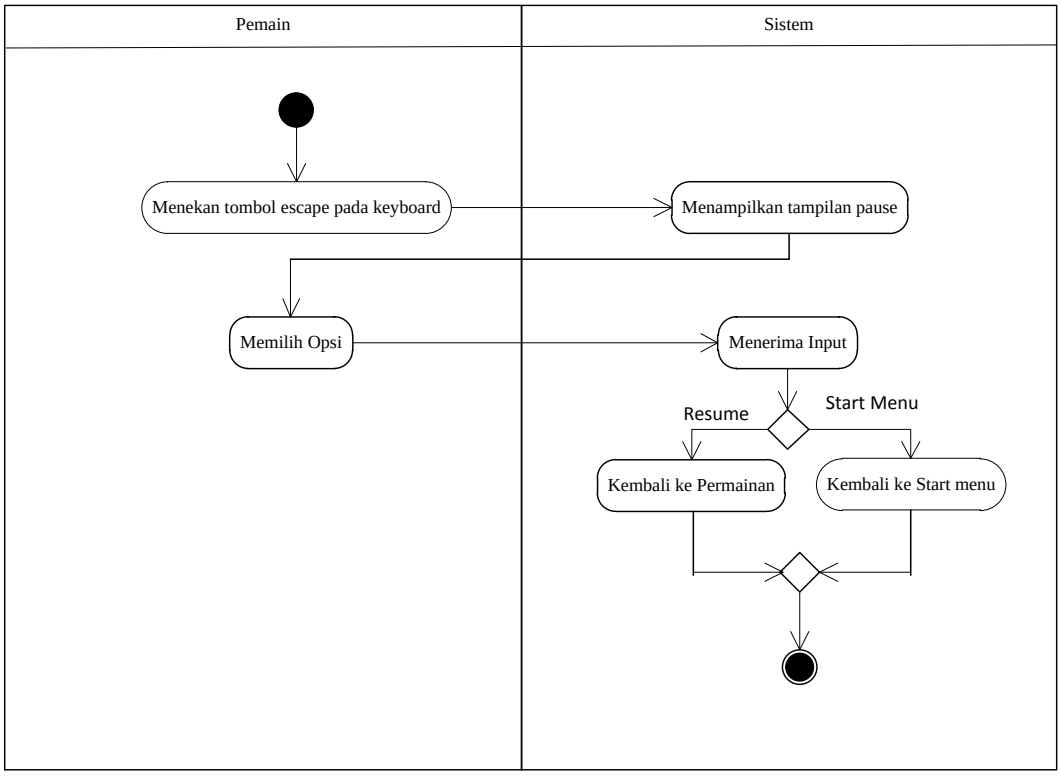


Gambar 3.21 *Activity diagram* Proses Permainan

Tabel 3.21 Penjelasan Proses permainan

Deskripsi	Tampilan ini menampilkan *start menu* setelah menjalankan *game*, pemain diberikan beberapa sub pilihan.
Normal Flow	1. System menampilkan permainan 2. Pemain mulai memainkan permainan 3. Jika berhasil menyelesaikan akan menampilkan tampilan *mission success* 4. Jika gagal menyelesaikan misi akan kembali menjalankan game dari awal.

E. *Activity Diagram Pause*

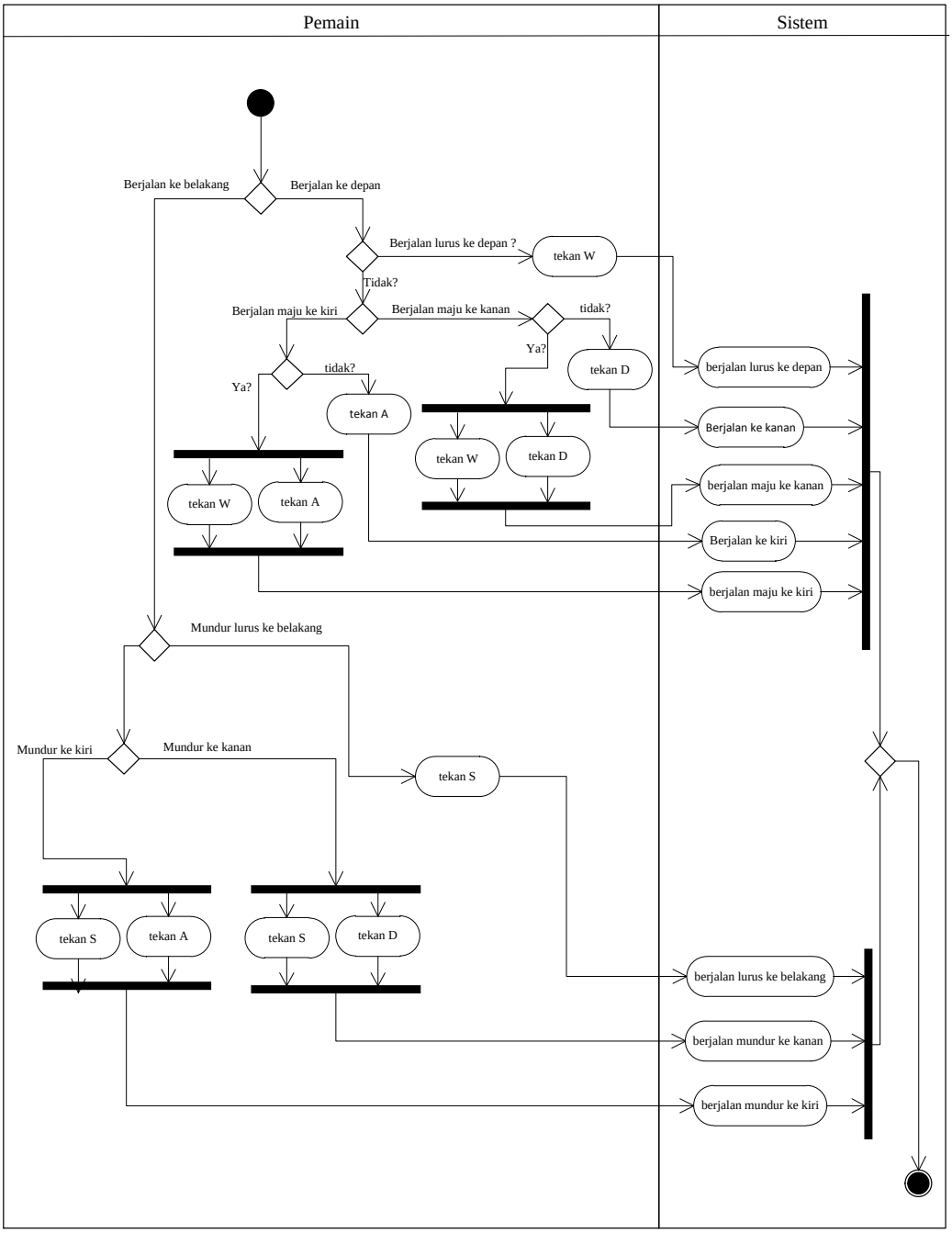


Gambar 3.22 *Activity Diagram Pause*

Tabel 3.22 Penjelasan Proses *Pause*

Deskripsi	Proses ketika pemain ingin memberhentikan sejenak permainan atau kembali *ke start menu*
Normal Flow	1. Pemain menekan tombol *escape* 2. Sistem menampilkan informasi *pause menu*. 3. Sistem akan kembali ke permainan apabila pemain memilih opsi resume 4. Sistem akan keluar permainan dan kembali ke *start menu* apabila pemain memilih opsi back to *start menu.*

F. *Activity Diagram* Berjalan

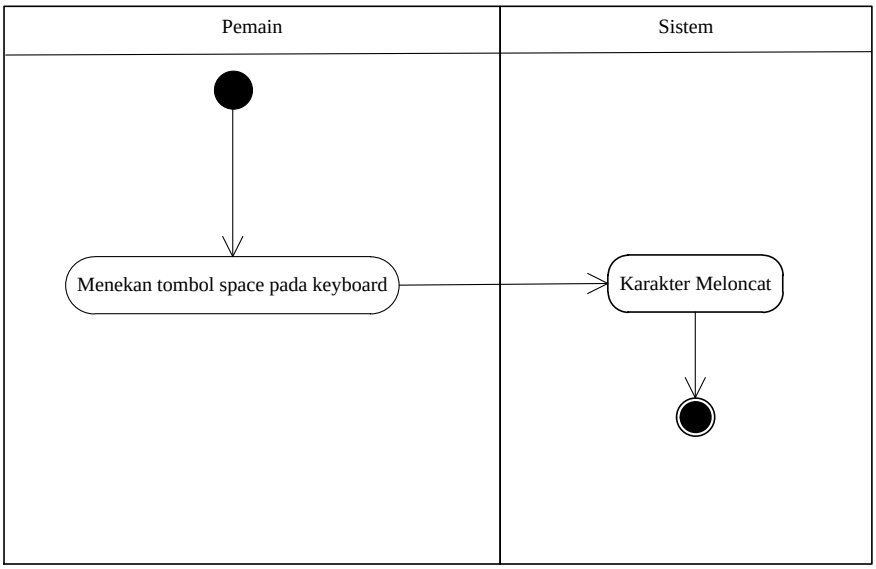


Gambar 3.23 *Activity Diagram* Berjalan

Tabel 3.23 Penjelasan Proses Berjalan

Deskripsi	Proses untuk menggerakan karakter atau tokoh utama yang dimainkan dalam permainan menggunakan tombol *keyboard.*
Normal Flow	1. Pemain menekan tombol W 2. Karakter berjalan ke depan 3. Pemain menekan tombol W dengan A secara bersamaan 4. Karakter berjalan maju ke sebelah kiri 5. Pemain menekan tombol A 6. Karakter berjalan ke kiri 7. Pemain menekan tombol W dengan D secara bersamaan 8. Karakter berjalan maju ke kanan 9. Pemain menekan tombol D 10. Karakter berjalan ke kanan 11. Pemain menekan tombol S 12. Karakter berjalan lurus mundur 13. Pemain menekan tombol S dengan D secara bersamaan 14. Karakter berjalan mundur ke kanan 15. Pemain menekan tombol S dengan A secara bersamaan 16. Karakter berjalan mundur ke kiri

G. *Activity Diagram* Meloncat

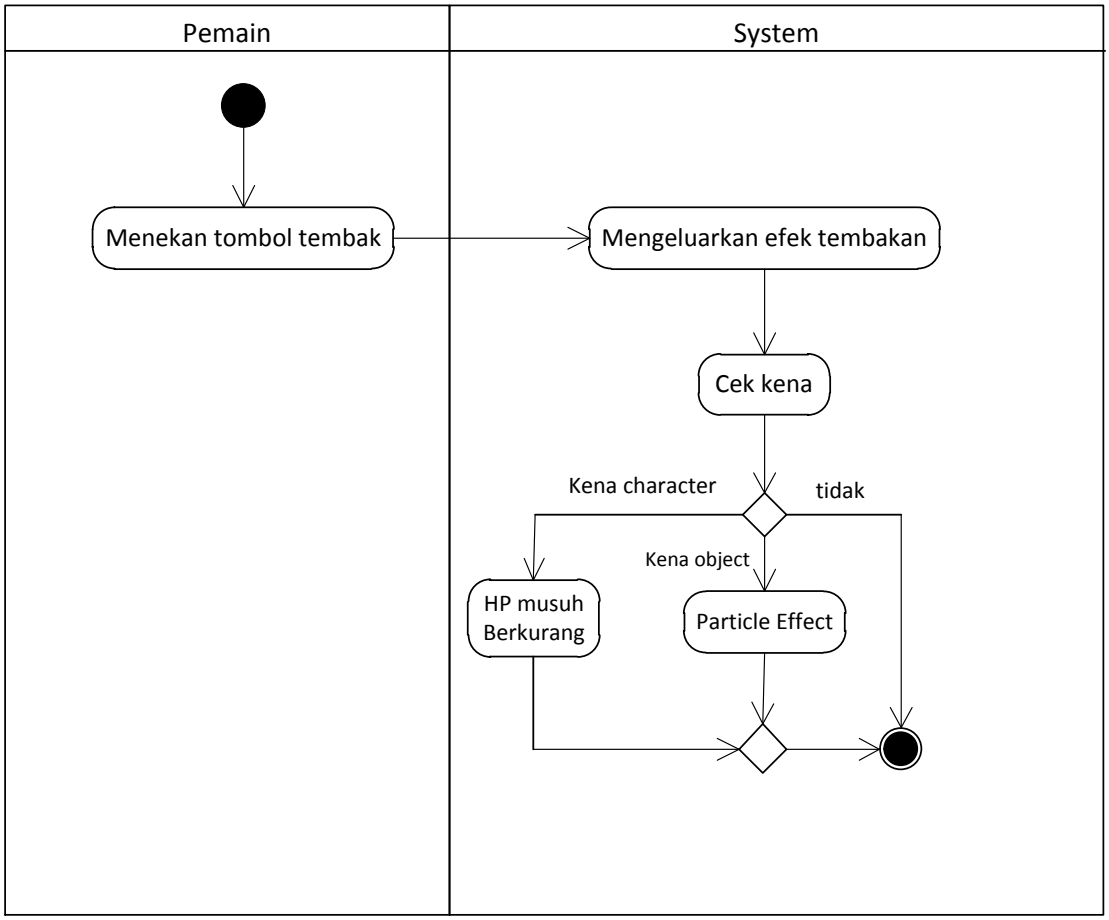


Gambar 3.24 *Activity Diagram* Meloncat

Tabel 3.24 Penjelasan Proses Meloncat

Deskripsi	Proses untuk menggerakan karakter atau tokoh utama yang dimainkan dalam permainan menggunakan tombol *keyboard.*
Normal Flow	1. Pemain menekan tombol space pada *keyboard* 2. Sistem menampilkan karakter yang meloncat sesuai inputan pemain

H. *Activity Diagram* Menembak *Player*

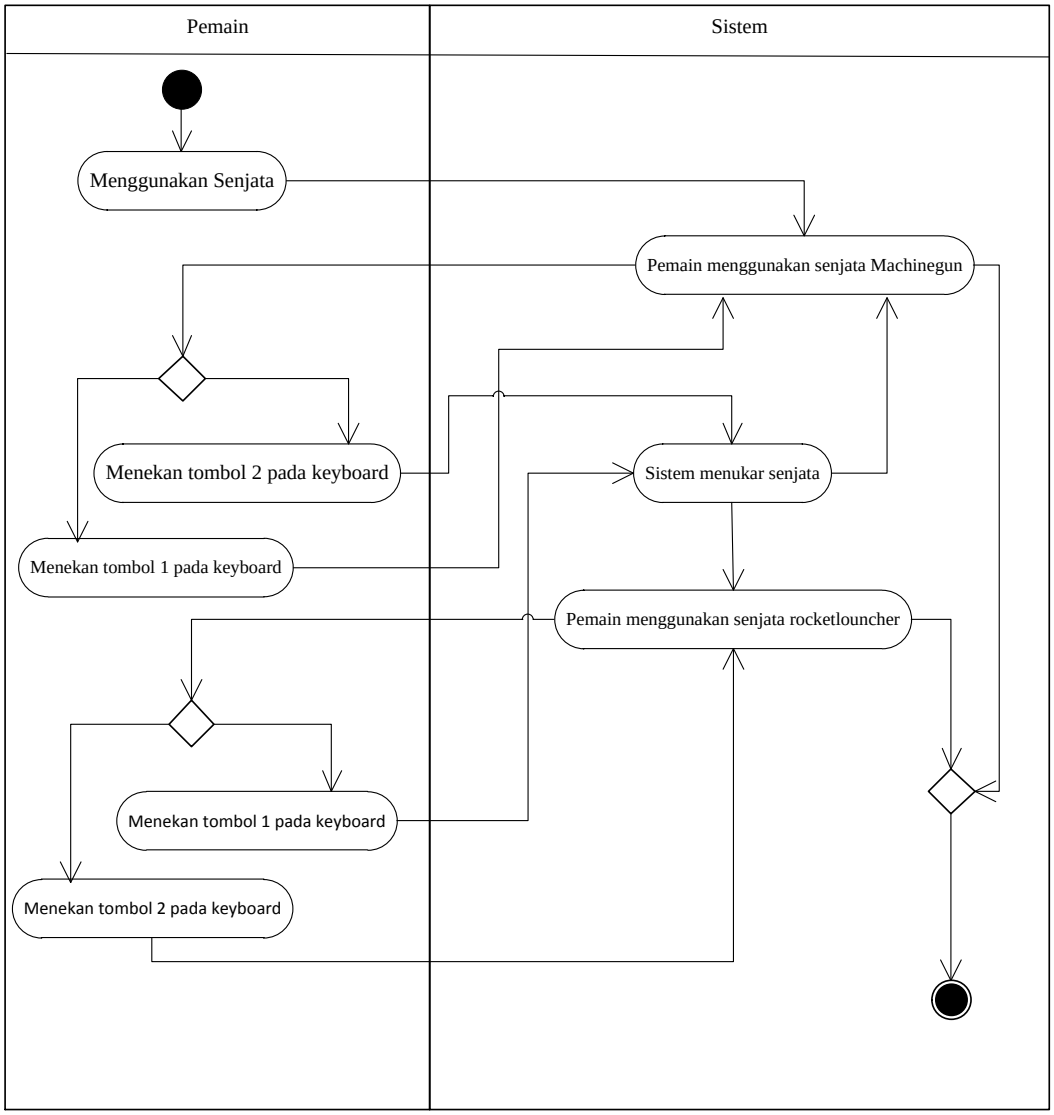


Gambar 3.25 *Activity Diagram* Menembak *Player*

Tabel 3.25 Penjelasan Proses Menembak

Deskripsi	Proses untuk melakukan tembakan dalam permainan
Normal Flow	1. Pemain menekan tombol kiri pada *mouse* untuk menembak 2. Sistem mengeluarkan efek tembakan 3. Jika mengenai character maka HP *character* berkurang jika mengenai objek menghasilkan *particle* api. 4. Peluru berkurang

I. *Activity Diagram* Mengganti Senjata

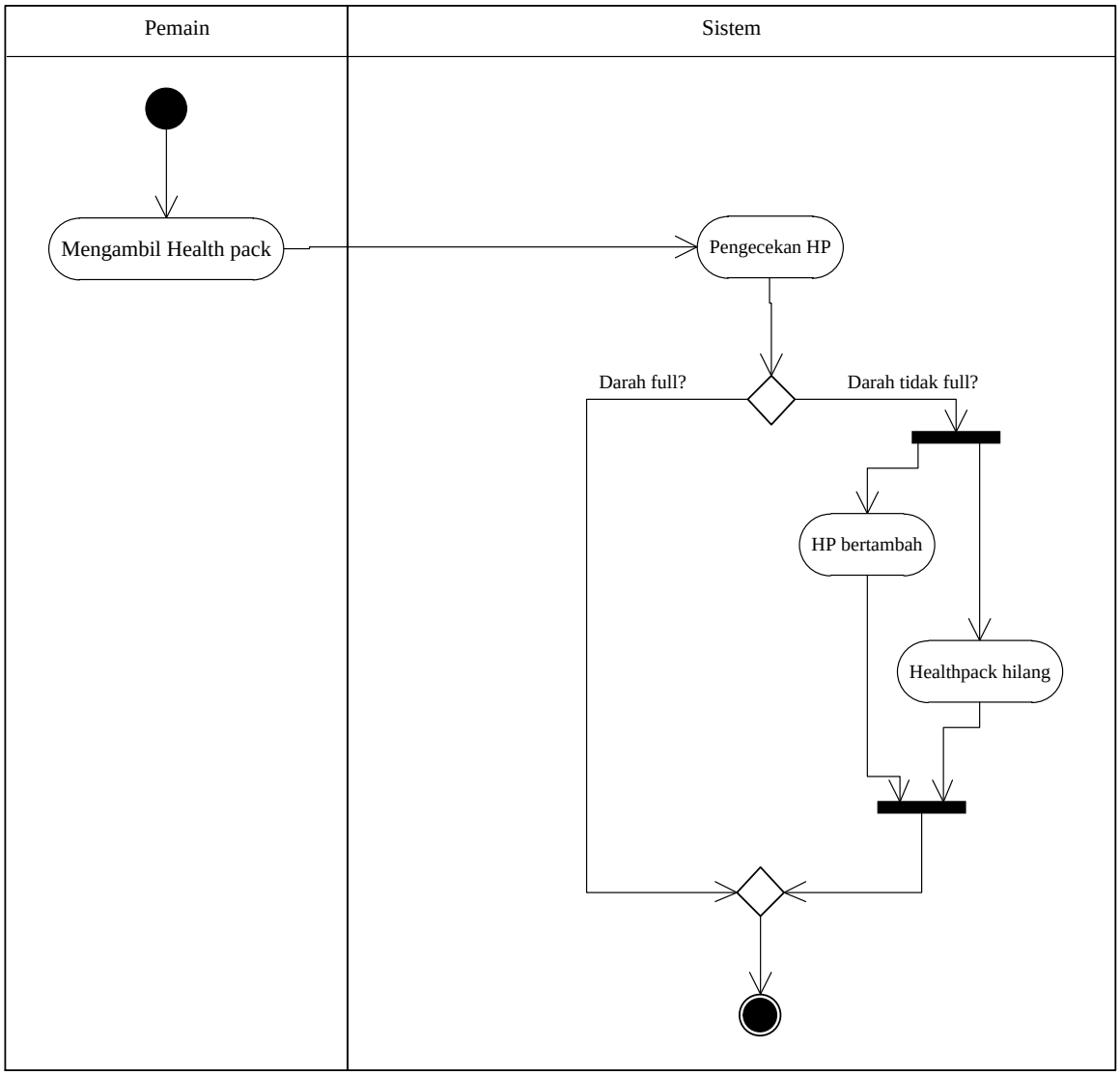


Gambar 3.26 *Activity Diagram* Mengganti Senjata

Tabel 3.26 Penjelasan Proses Mengganti Senjata

Deskripsi	Proses mengganti senjata dengan pilihan senjata *machinegun* atau senjata *rocketlouncher*
Normal Flow	1. Pemain menggunakan senjata 2. Sistem menampilkan pemain menggunakan *Machinegun* 3. Pemain menekan tombol 1 pada *keyboard* 4. Sistem tidak mengganti senjata 5. Pemain menekan tombol 2 pada *keyboard* 6. Sistem menukar senjata 7. Sistem menampilkan pemain menggunakan senjata *Rocketouncher* 8. Pemain menekan tombol 2 pada *keyboard* 9. Sistem tidak mengganti senjata 10. Pemain menekan tombol 1 pada *keyboard* 11. Sistem menukar senjata 12. Sistem menampilkan pemain menggunakan sejata *Machinegun*

J.*Activity Diagram* Mengambil *Health Pack* (darah)

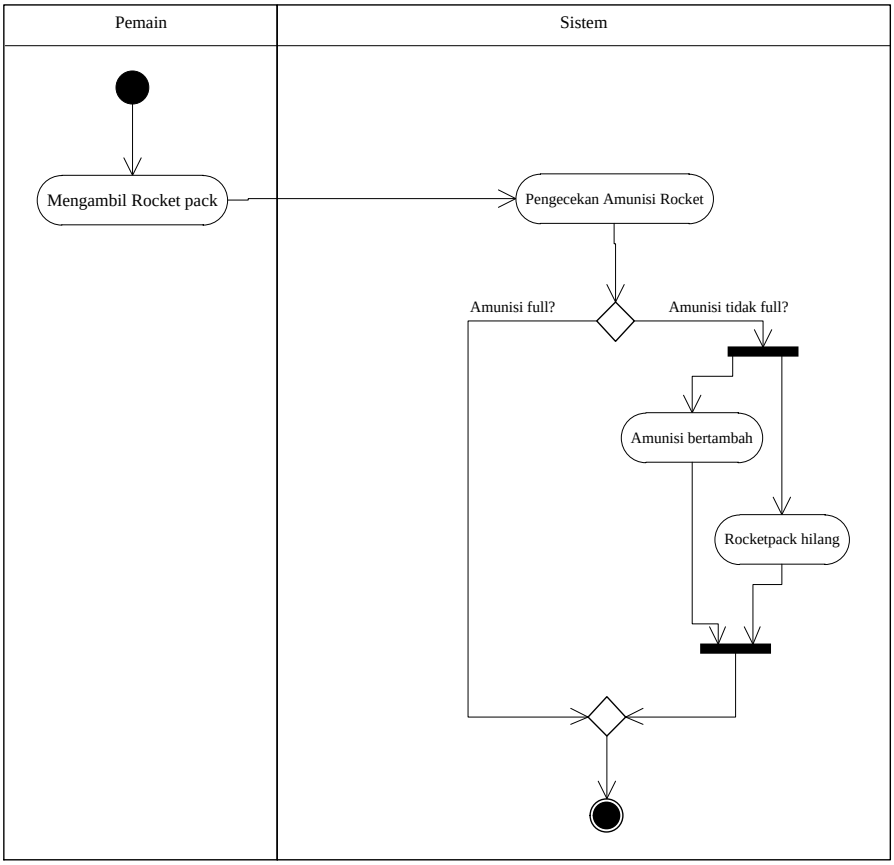


Gambar 3.27 *Activity Diagram* Mengambil *Health Pack*

Tabel 3.27 Penjelasan Proses Mengambil Health Pack

Deskripsi	Proses dimana pemain mengambil *Health pack*
Normal Flow	1. Pemain mengambil *Health pack* 2. Sistem melakukan pengecekan HP Player 3. Apabila darah pemain *full* (penuh), sistem tidak mengalami perubahan 4. Apabila darah pemain tidak *full*, sistem menambahkan HP pemain sesuai nilai *health pack* dan menampilkan *health bar* pemain yang bertambah, *health pack* menghilang

K. *Activity Diagram* Mengambil *Rocket Pack* (amunisi rocketlouncher)

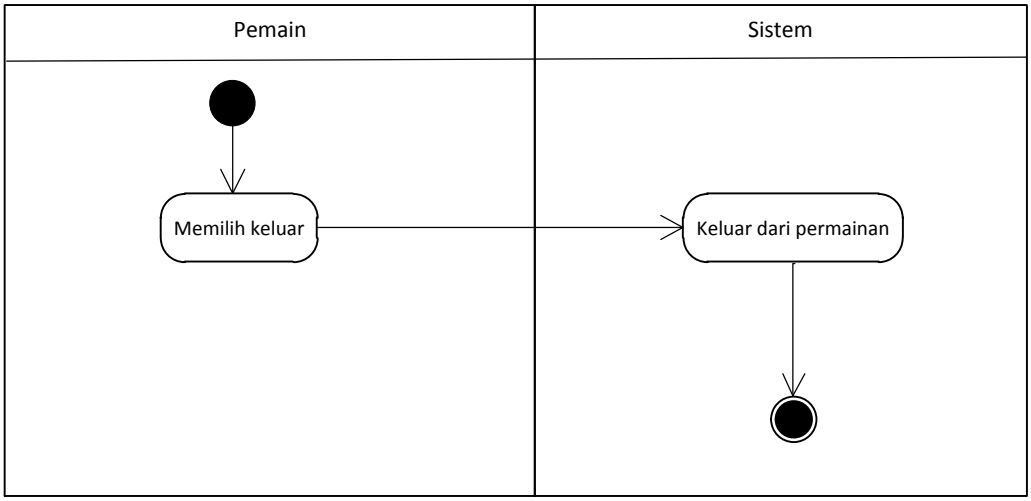


Gambar 3.28 *Activity Diagram* Mengambil *Rocket pack*

Tabel 3.28 Penjelasan Proses Mengambil *Rocket Pack*

Deskripsi	Proses dimana pemain mengambil *Rocket pack*
Normal Flow	1. Pemain mengambil *Rocket pack* 2. Sistem melakukan pengecekan Amunisi *Rocketlouncher* 3. Apabila amunisi *full* (penuh), sistem tidak mengalami perubahan 4. Apabila amunisi tidak *full*, sistem menambahkan amunisi pemain sesuai nilai *rocket pack* dan menampilkan amunisi pemain yang bertambah, menghilang

L. *Activity Diagram Exit Game*



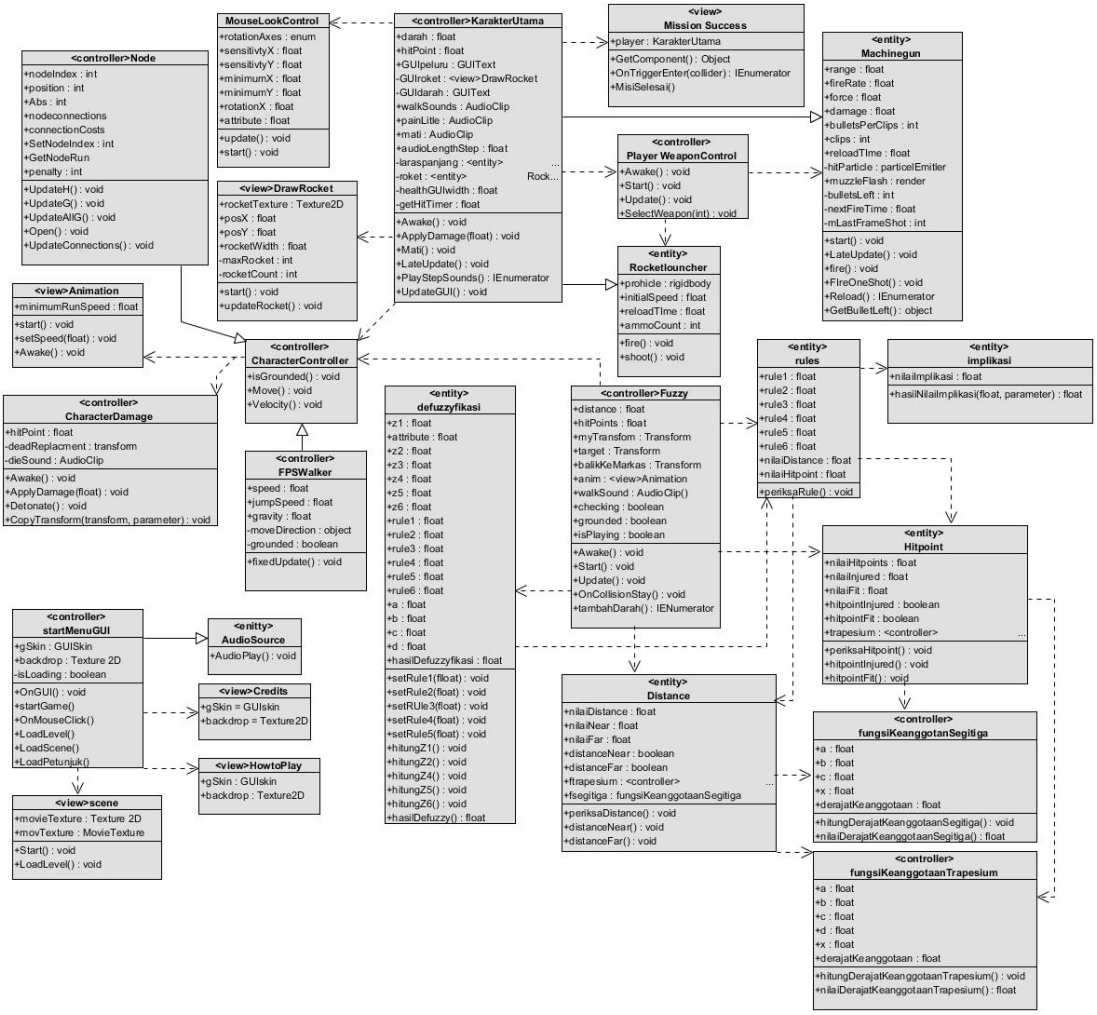
Gambar 3.29 *Activity Diagram Exit Game*

Tabel 3.29 Penjelasan Proses *Exit Game*

Deskripsi	Proses dimana pemain Keluar dari *game*
Normal Flow	1. Pemain memilih menu *exit* pada *start menu* 2. Sistem akan mengakhiri permainan dan exit applikasi

3. Class Diagram

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

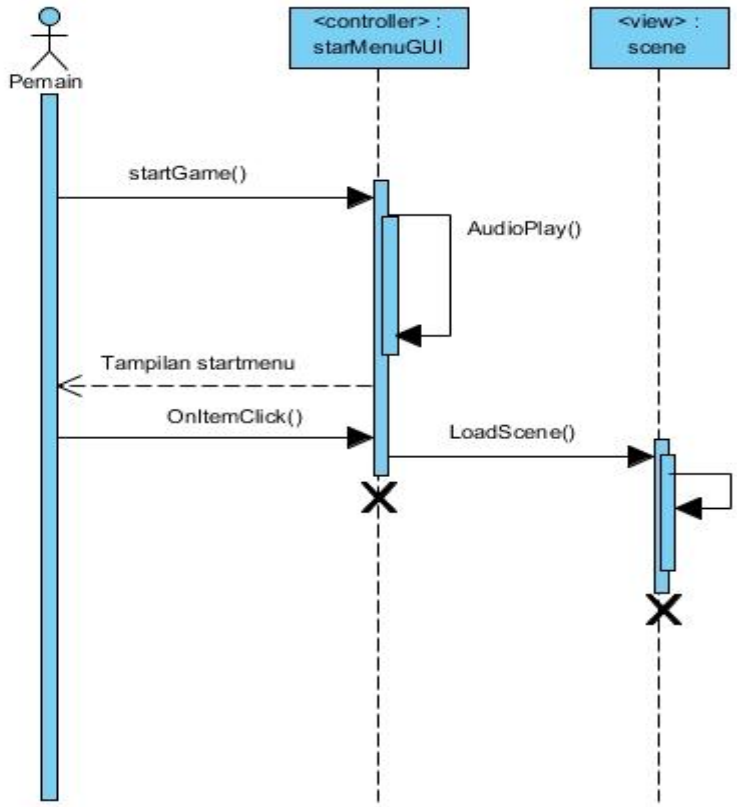


Gambar 3.30 *Class diagram game Alien Hunter*

4. Sequence Diagram

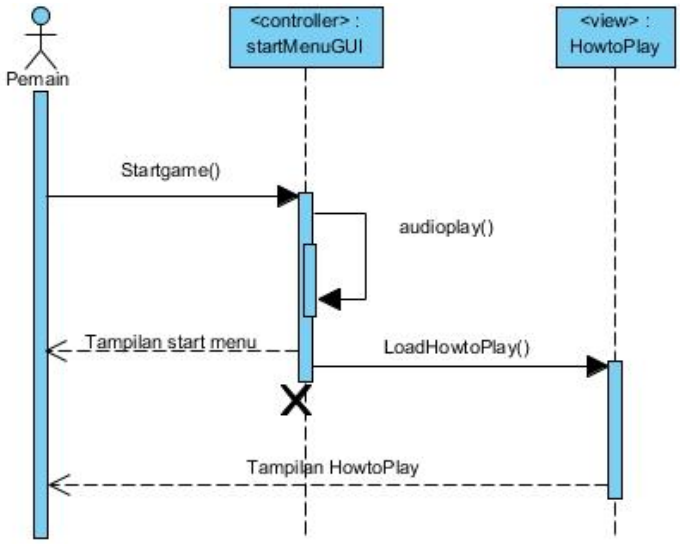
Sequencial diagram menggambarkan interaksi antar objek di dalam dan disekitar sistem berupa *message* yang digambarkan terhadap waktu. Berikut ini beberapa *sequencial diagram* yang terdapat pada *game Alien Hunter* yaitu :

A. *Sequence Diagram* Scene Permainan



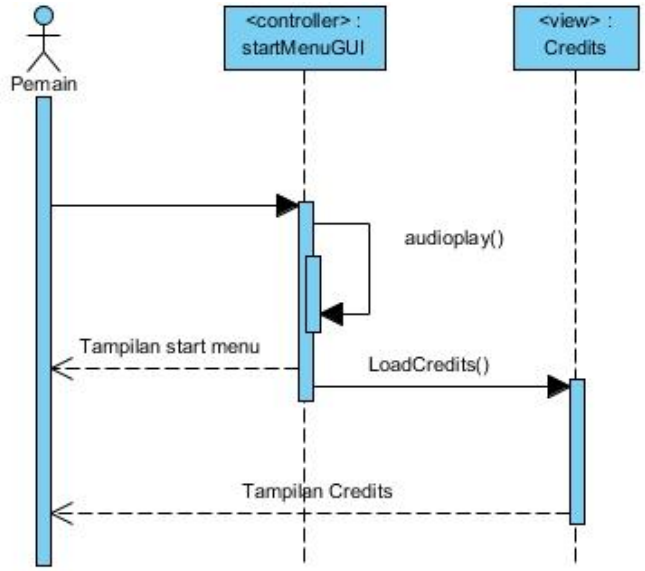
Gambar 3.31 *Sequence Diagram* Mainkan

B. *Sequence Diagram Howtoplay*



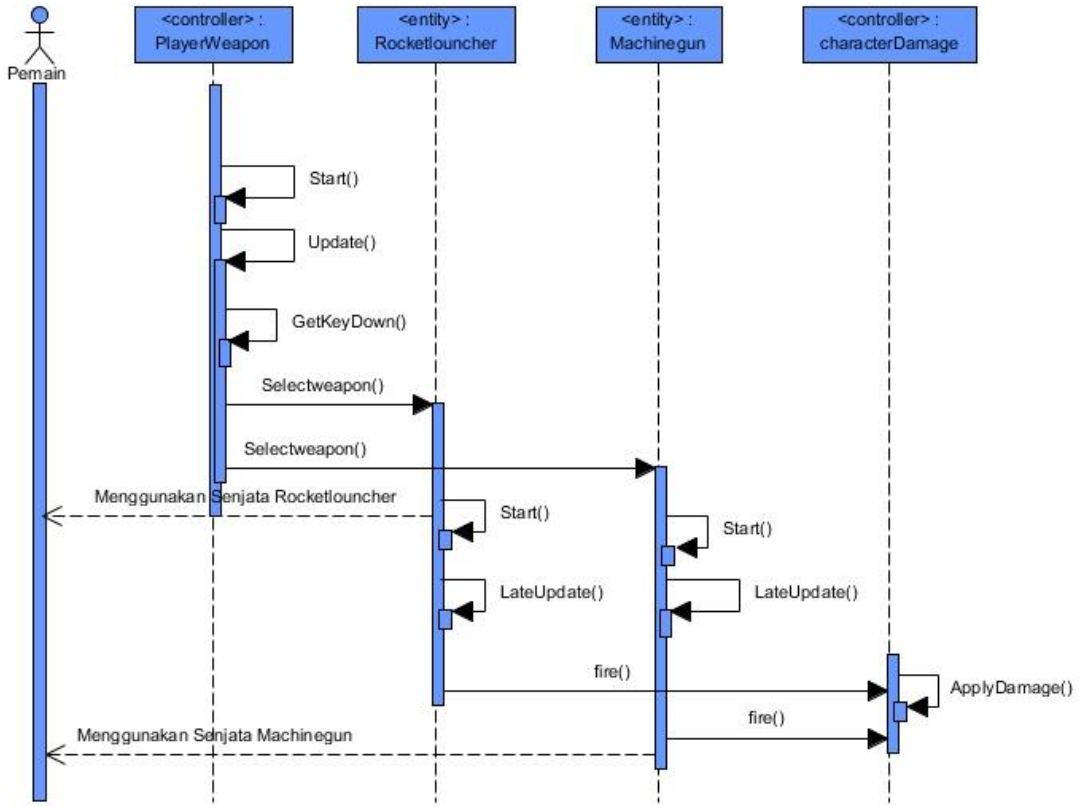
Gambar 3.32 *Sequence Diagram Howtoplay*

C. *Sequence Diagram Credits*



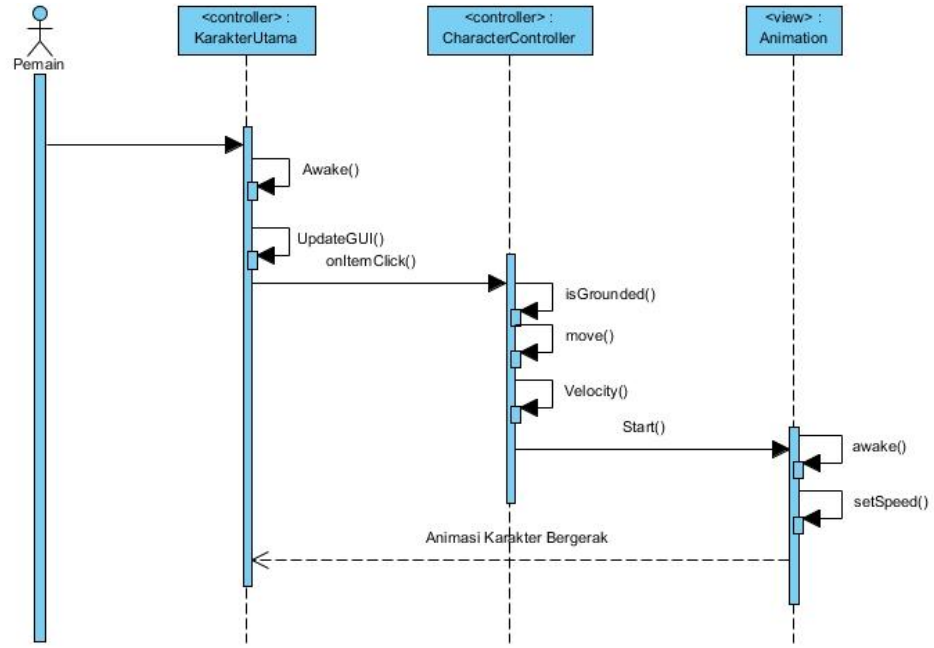
Gambar 3.33 *Sequence Diagram Credits*

D. *Sequence Diagram* Menembak



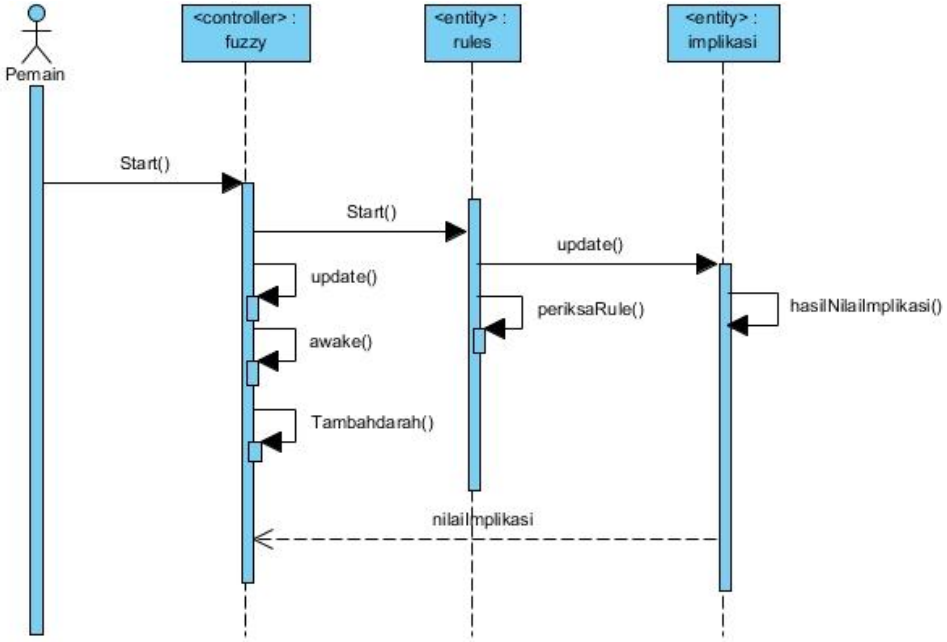
Gambar 3.34 *Sequence Diagram* Menembak

E. *Sequence Diagram* Bergerak



Gambar 3.35 *Sequence Diagram* Bergerak

F. *Sequence Diagram* Fuzzy



Gambar 3.36 *Sequence Diagram* Fuzzy

3.2 Perancangan Sistem

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

3.2.1 Perancangan Komponen Permainan

3.2.1.1 Karakter

A. Karakter Utama

Karakter utama dalam game ini bernama agen L. Karakter ini memiliki seragam pasukan khusus yang dirancang untuk melawan alien yang berwarna coklat tua. Agen L juga memakai sepatu bot khusus dan juga dilengkapi dengan *helm* pengaman. Dengan Semua peralatan yang ada agen L siap untuk menghadapi seluruh serangan alien.

Game Alien Hunter adalah game yang *bergenre* FPS *(First Person Shooter)* yang mempunyai sudut pandang *camera* dalam game yang tidak memperlihatkan pergerakan karakter utama secara keseluruhan. Oleh sebab itu tubuh, kepala, kaki dan tangan dari karakter utama tidak diperlihatkan tetapi hanya memperlihatkan bagian moncong senjatanya saja akan. Pada saat di dalam game *player* hanya akan melihat senjata yang dipakai dan *crossfire* yang berada di tengah layar.

***Properties* karakter utama**

Properties dari karakter utama dijelaskan pada tabel 3.30.

Tabel 3.30 *Properties* karakter utama

Nick Name	Agen L
Hit Points (HP)	250
Abilities	Berlari, melompat, menembak
Gun	*Machine gun, Rocket louncher*

B. Senjata

Ada dua macam senjata yang dapat digunakan oleh agen L, yang pertama adalah *machinegun* yang dapat dilihat pada gambar 3.37 yang mempunyai *unlimited ammo* (peluru yang tidak terbatas) dan yang kedua adalah *Rocket Louncher* yang dapat dilihat pada gambar 3.38 yang *limited ammo* peluru dibatasi hanya 20 peluru dan dapat dilihat distatus peluru pada layar permainan.



Gambar 3.37 Senjata *Machinegun*



Gambar 3.38 Senjata *Rocket Louncher*

Perbandingan *Properties* Senjata

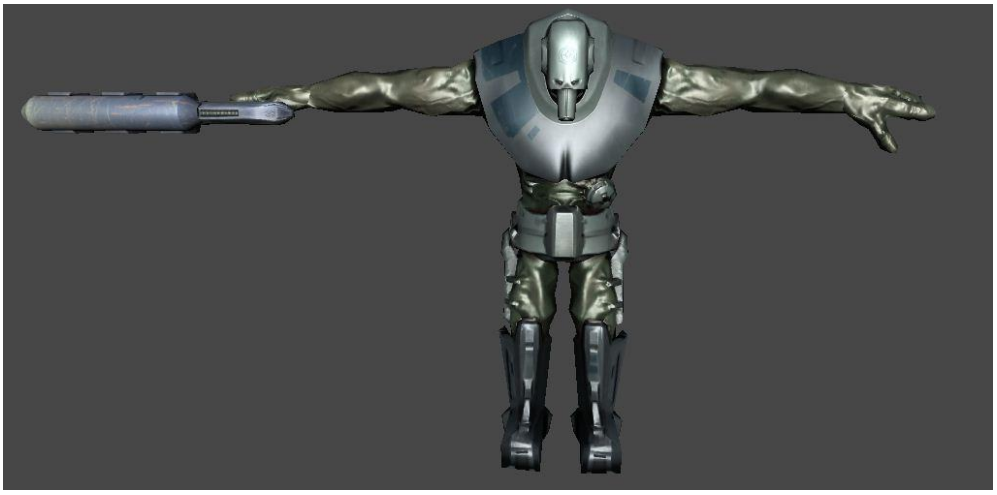
Perbandingan *properties* dari dua senjata yang ada dijelaskan pada tabel 3.31.

Tabel 3.31 Perbandingan *properties* senjata

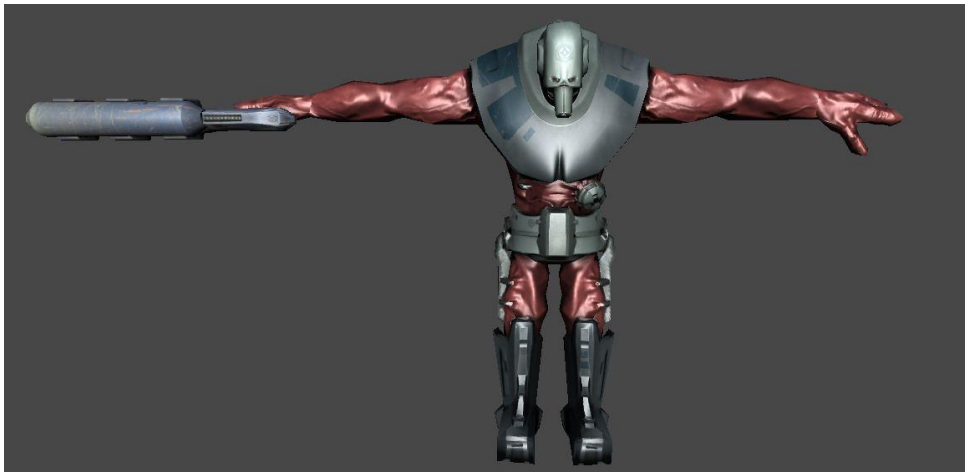
Properties	***Machine Gun***	***Rocket Louncher***
Damage	5	40
Range	50	100
Reload Time	0.5	-
Bullet	Unlimited	20

C. Enemy dan *Monster*

Makhluk alien ini memiliki tinggi yang sama karakter utama. Dalam *game* ini terdapat dua jenis alien, *white alien* seperti pada gambar 3.39 dan *red alien* seperti pada gambar 3.40. Bentuk keduanya sama akan tetapi alien yang berwarna merah lebih agresif dan memiliki kekuatan serangan yang melebihi *white alien.* Kedua alien ini menyerang dengan menggunakan senjata *rocket* yang memiliki tenaga nuklir. *Red alien* akan banyak ditemukan di level 2, sedangkan di level 1 terdapat lebih sedikit red alien. Dalam *game* ini juga terdapat *sentrygun* yang dapat dilihat pada gambar 3.42 yang akan menembak otomatis ketika target dalam jangkauan jarak tembak mereka dan juga terdapat *spider robot* seperti pada gambar 3.41 yang akan mengejar *player* dan secara otomatis meledakkan diri jika telah dekat dengan *player*.



Gambar 3.39 Enemy White alien



Gambar 3.40 Enemy Red alien



Gambar 3.41 Spider robot



Gambar 3.42 *Sentrygun*

Perbandingan *Properties Enemy*

Perbandingan *properties* dari *enemy* yang ada didalam *game* dijelaskan pada tabel 3.32.

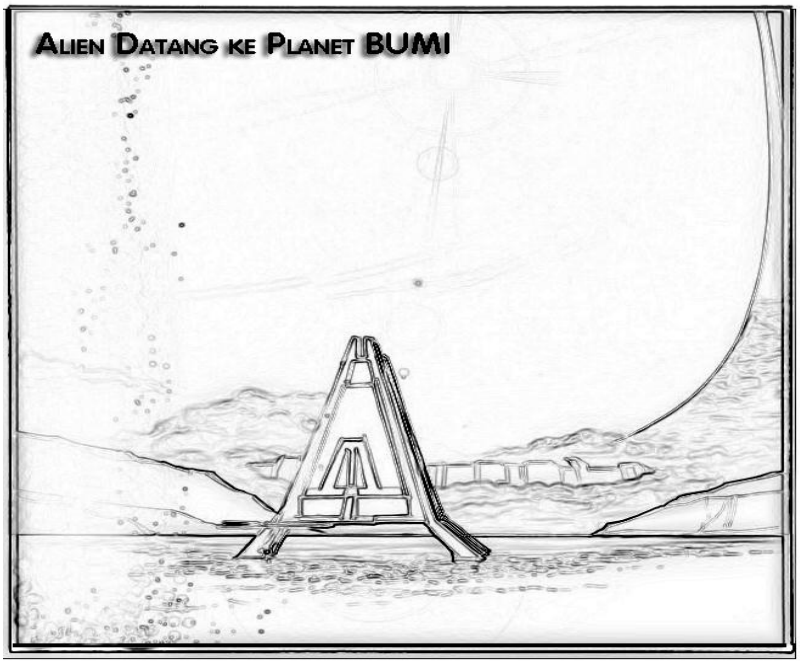
Tabel 3.32 Perbandingan properties enemy

Properties	***White alien***	***Red alien***	***Sentry gun***	***Spider robot***
Speed	3	5	-	7
Range Shoot	15	20	25	-
Delay Shoot	0.5	0.4	0.1	-
Damage	30	30	5	50
HP	100	100	100	40

3.4.1.2 Story Board

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

Berikut *storyboard* dari *Game* FPS 3D Alien Hunter yaitu :



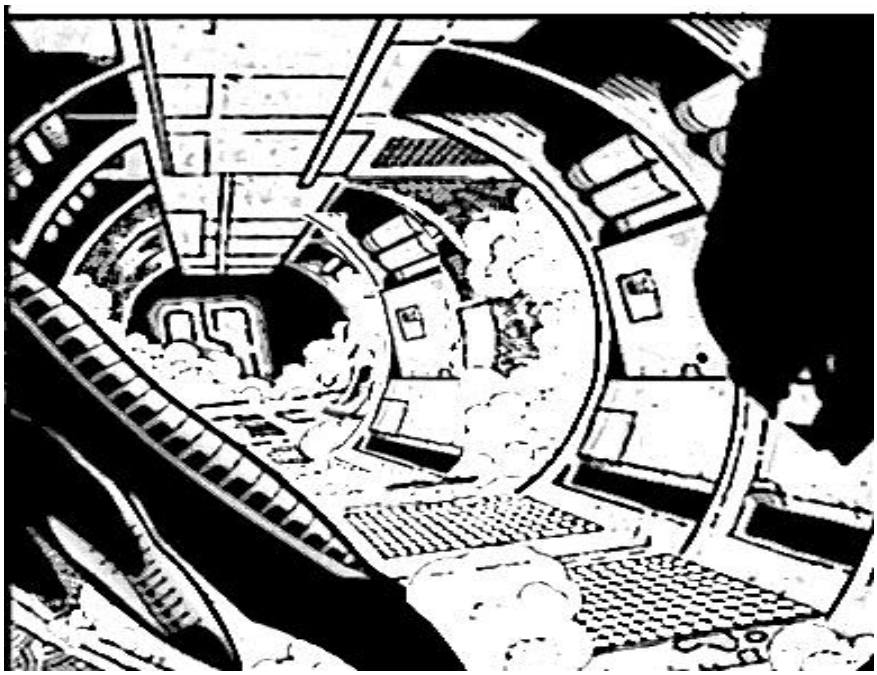
Gambar 3.43 *Story Board 1*

Gambar 3.43 menceritakan awal mula alien datang kebumi yang dapat dilihat pada *intro scene* yang ada didalam *game*.



Gambar 3.44 *Story Board 2*

Gambar 3.44 memperlihatkan kondisi tempat *player* memainkan *level* pertama permainan. Berada pada sebuah kota dengan gedung-gedung tinggi yang telah dikuasai alien. Pada kota ini terdapat sebuah gudang yang menyimpan reaktor nuklir yang harus diambil oleh *player* untuk menyelesaikan misi pertama.



Gambar 3.45 *Story Board 3*

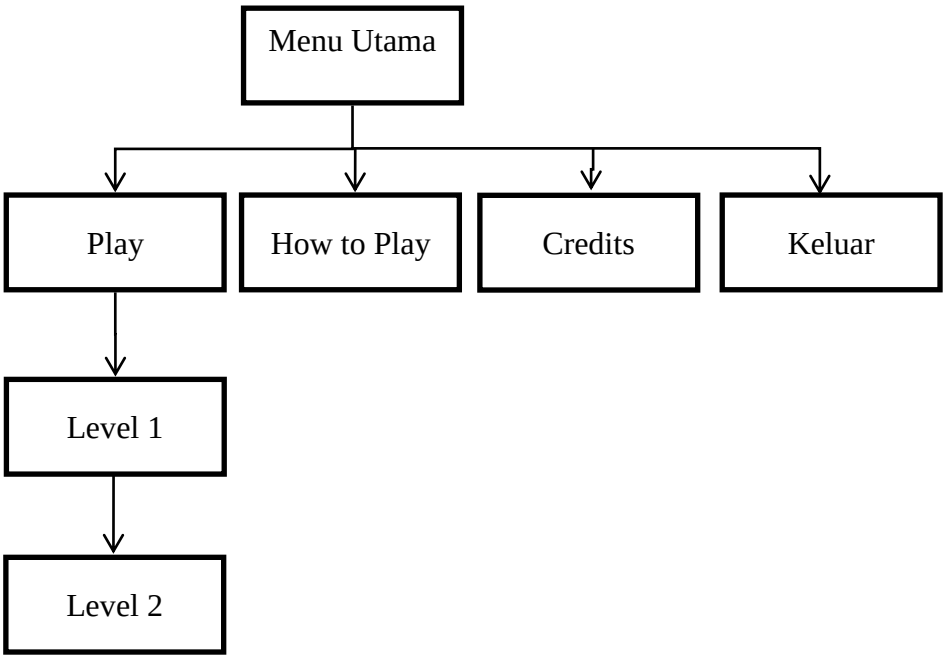
Gambar 3.45 memperlihatkan kondisi tempat *player* memainkan *level* kedua permainan. Berada pada sebuah wilayah yang didalamnya terdapat markas komando alien. *Player* harus bisa masuk kedalam markas komando alien dan mencari ruang kendalinya untuk mengaktifkan reaktor nuklir yang telah didapatkan pada misi pertama.

3.2.2 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,

3.2.2.1 Perancangan Struktur Menu

Perancangan struktur menu dirancang untuk menyediakan fungsi-fungsi yang akan digunakan dalam *game Alien Hunter* .

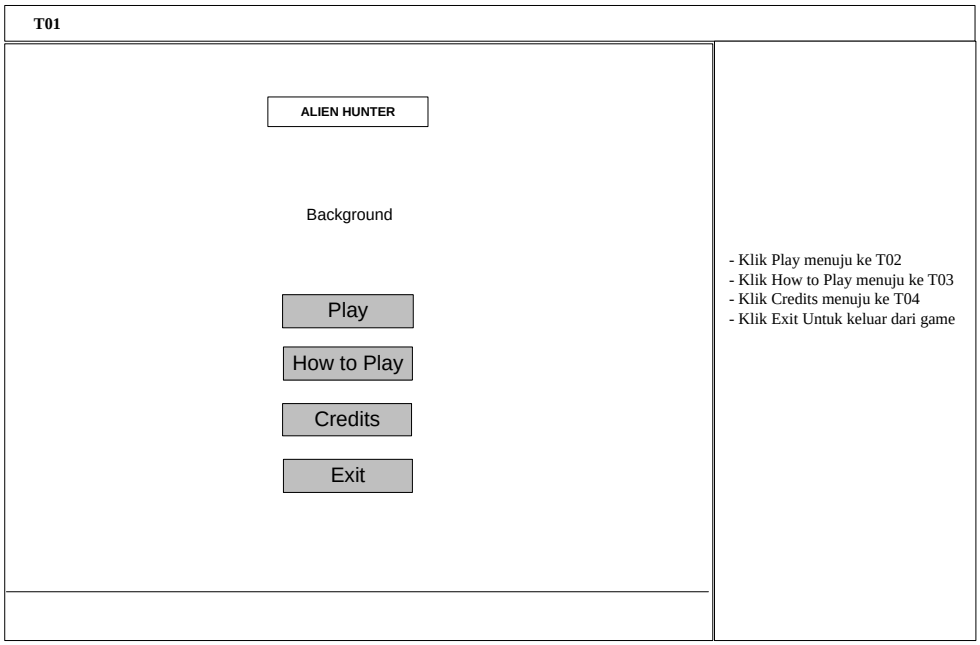


Gambar 3.46 Struktur Menu Utama

3.2.3 Perancangan Antarmuka

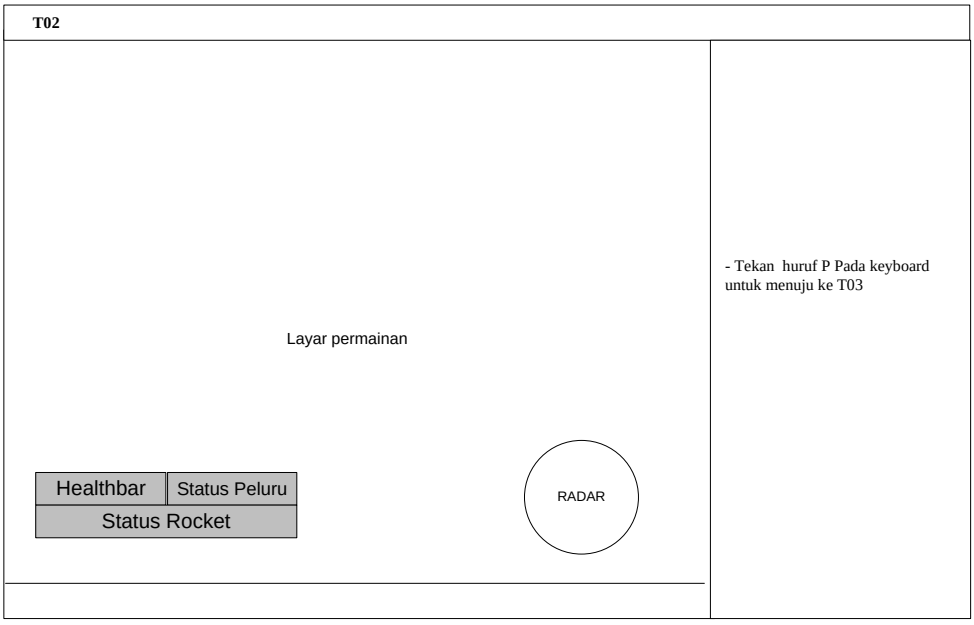
Perancangan antarmuka dibutuhkan untuk mawakili keadaan sebenarnya dari aplikasi yang akan dibangun. Berikut ini beberapa perancangan antarmuka dari aplikasi yang akan dibangun yaitu :

1. Menu Utama



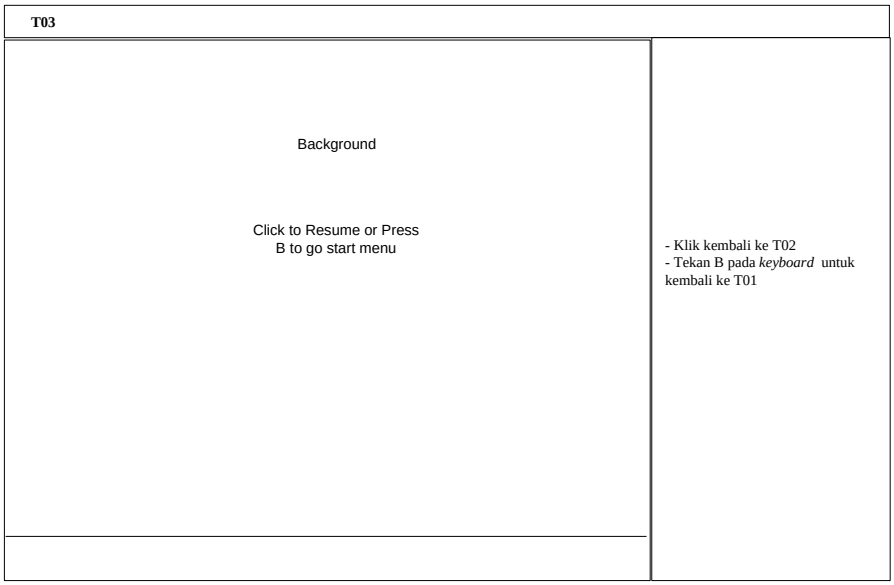
Gambar 3.47 Tampilan Menu Utama

2. Permainan



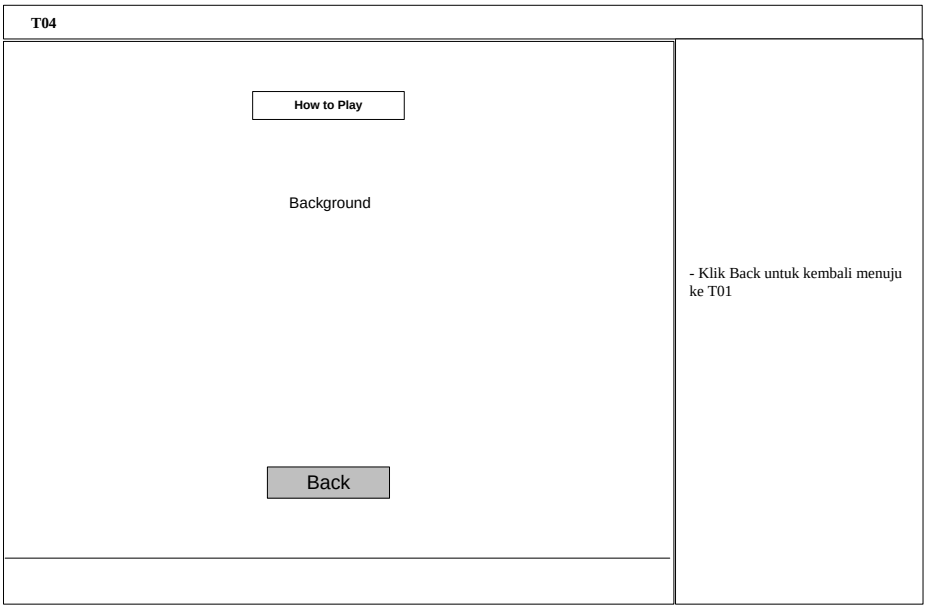
Gambar 3.48 Tampilan Permainan

3. Pause Screen



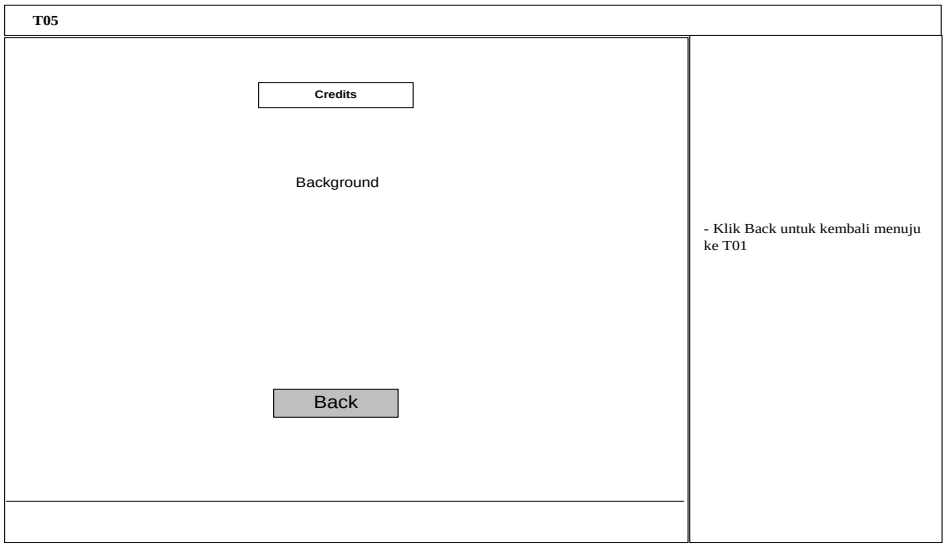
Gambar 3.49 Tampilan Pause Screen

4. Menu How to Play



Gambar 3.50 Tampilan Menu HowtoPlay

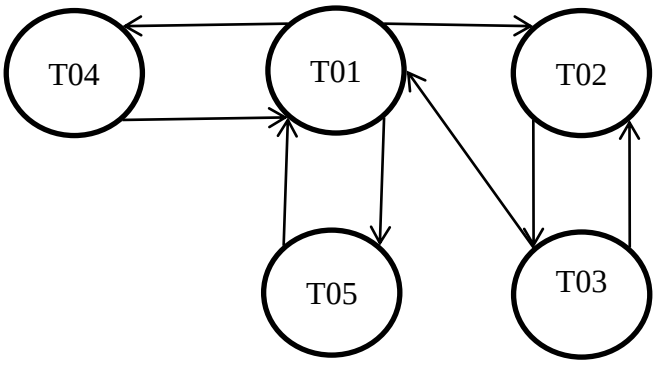
5. Menu Credits



Gambar 3.51 Tampilan Menu Credits

3.2.3.1 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.



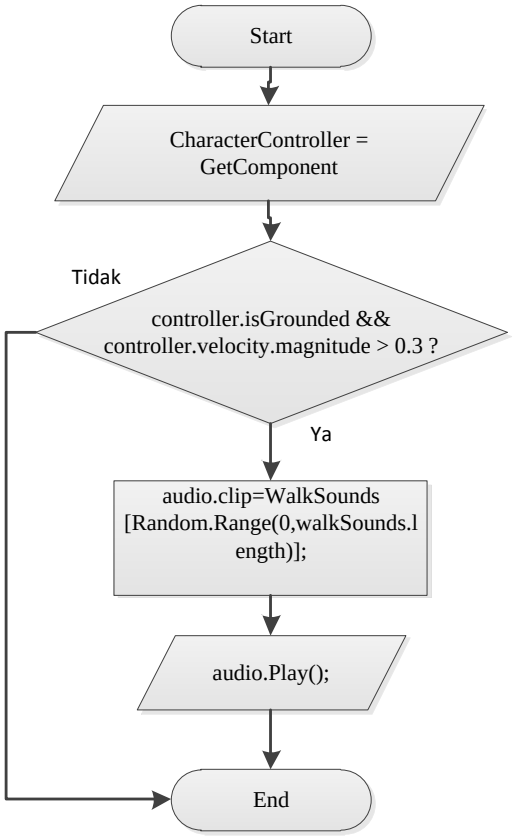
Gambar 3.52 Jaringan Semantik Pengguna

3.2.4 Perancangan *Method*

Perancangan *method* merupakan perancangan yang berfungsi untuk mendeskripsikan *method-method* yang berada di dalam aplikasi. *Method* dapat dipanggil dengan menyertakan variabel, baik hanya satu variabel, banyak variabel atau bahkan tidak ada sama sekali. Adapun *method-method* yang terdapat dalam *game Alien Hunter* yang akan dibangun adalah sebagai berikut :

1. ***Method PlayStepSounds***

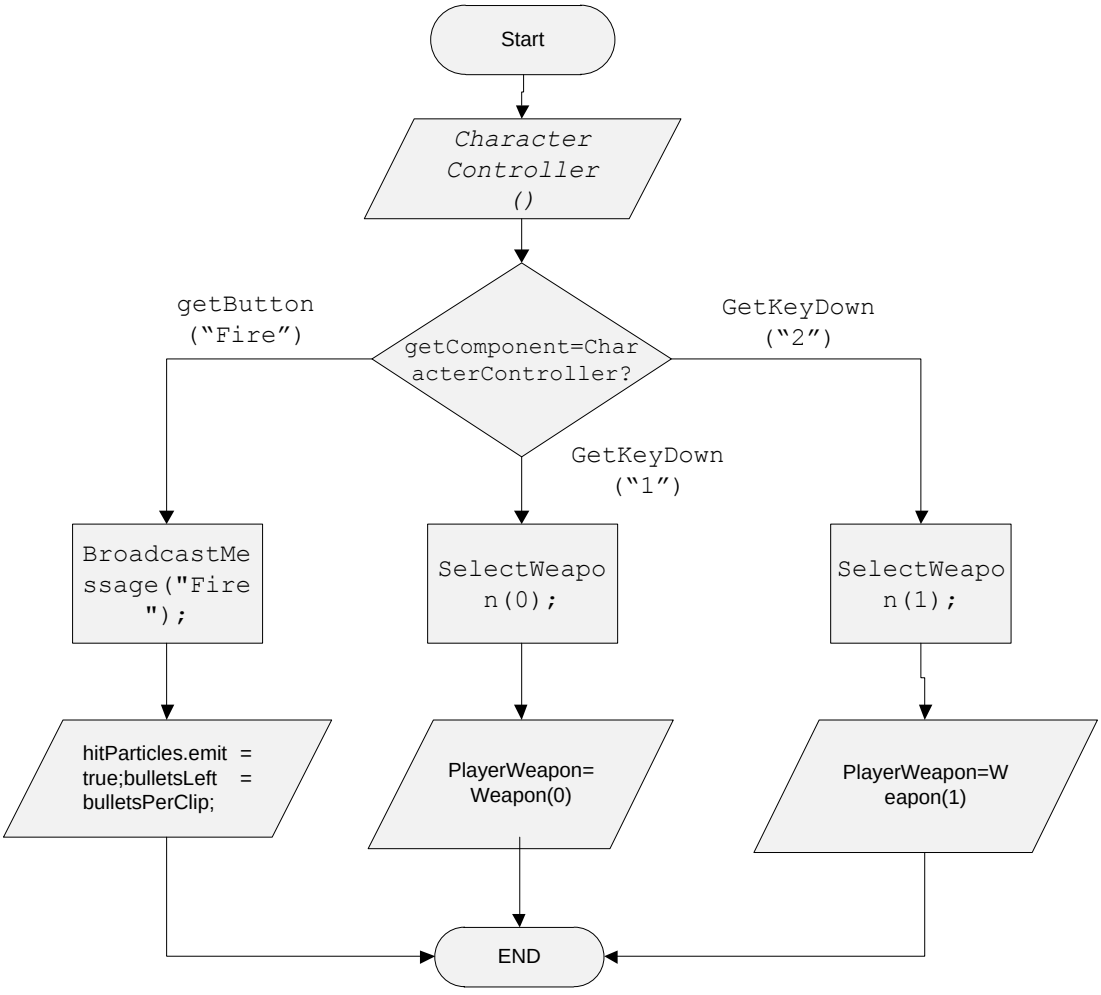
   Method ini digunakan untuk membuat efek suara langkah jika *FPS player* bergerak dalam game. Untuk lebih jelasnya dapat dilihat pada gambar *flowchart* dibawah ini :



Gambar 3.53 *Flowchart Method PlayStepSounds*

2. *Method Player Weapon*

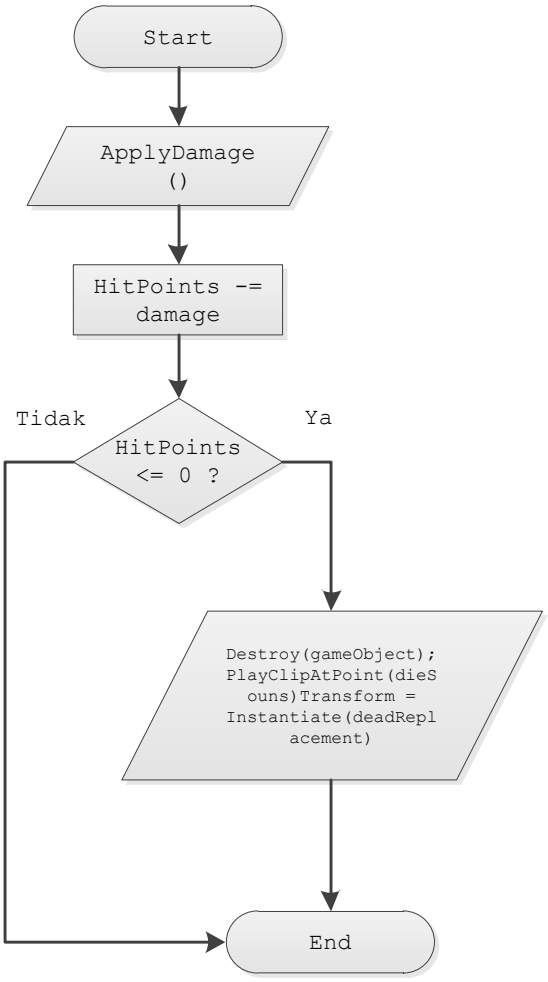
Method Player Weapon digunakan untuk menampung semua atribut-atribut senjata dari yang ada pada *FPS Player*. Untuk lebih jelasnya dapat dilihat pada gambar *flowchart* dibawah ini:



Gambar 3.54 *Flowchart Method Player Weapon*

3. *Method Character Damage*

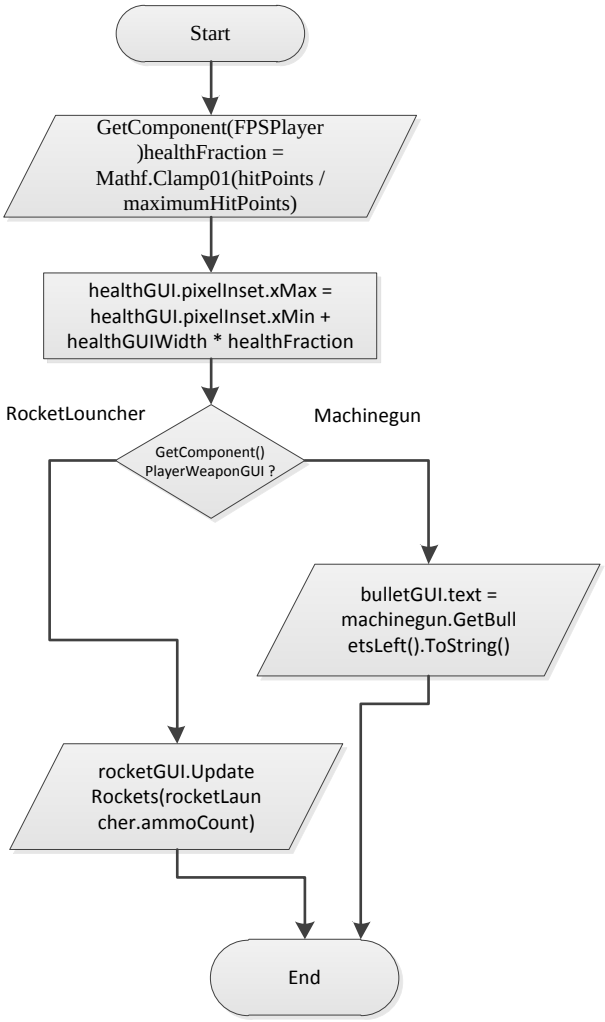
Method character damage digunakan untuk membuat character menerima damage yang akan mengurangi *hitpoints*. Untuk lebih jelasnya dapat dilihat pada gambar *flowchart* dibawah ini:



Gambar 3.55 *Flowchart Method Character Damage*

4. *Method UpdateGUI*

Method ini akan melakukan perubahan terhadap GUI berupa pengurangan peluru ketika pemain menembak, baik itu menggunakan senjata *machine gun* atau *rocket louncher*. Untuk lebih jelasnya dapat dilihat pada gambar *flowchart* dibawah ini:



Gambar 3.56 *Flowchart Method UpdateGUI*