BAB III
ANALISIS DAN PERANCANGAN SISTEM

3.1 Analisis Masalah

Neural network atau lebih dikenal sebagai jaringan saraf tiruan (JST) merupakan jaringan dari sekelompok unit pemroses kecil yang didasari oleh jaringan saraf manusia yang dapat mengubah strukturnya untuk memecahkan masalah berdasarkan informasi eksternal maupun internal yang berada dalam jaringan tersebut.

Adapun inspirasi awal dari pembentukan algoritma ini sendiri yaitu sistem pemrosesan informasi yang ada pada otak manusia, sel syaraf memilii cabang struktur input (*dendrites*), dimana sebuah inti sel dan percabangan struktur output (*axon*) terhubung dengan *dendrites* yang lain melalui sebuah *synapse*. Sehingga ketuka sebuah sel aktif, akan muncul sinyal elektronik pada *axon* denga batasan yang disebut *treshold*.

JST pada umumnya digunakan dalam pemecahan kasus yang dapat dikatakan kurang praktis jika dikerjakan secara manual, seperti pengambilan keputusan, optimasi nilai dan masih banyak yang lainya,sehingga tidak menutup kemungkinan JST dapat digunakan dalam bidang *game*. Namun pada saat ini perkembangan *game* yang menggunakan algoritma neural network didalamnya masih dirasakan kurang berkembang, sehingga perlu diadakan pengembangan lebih lanjut baik oleh pihak developer maupun perseorangan.

Dari segi keunggulan antara sistem jarigan syaraf tiruan dengan sistem konvensional, sistem konvensional sendiri menggunakan pendekatan algoritma, yang dapat didefinisikan bahwa sistem konvensional akan menjalankan sekumpulan perintah untuk melakukan proses pemecahan suatu masalah, dimana saat masalah yang dihadapi memiliki pola baru yang tidak di kenal, maka sistem konvensional tidak dapat memecahkan masalah tersebut, hal ini bertolak belakang dengan sistem jaringan saraf tiruan yang mempelajari pola pola masalah untuk menentukan pola penyelesaian masalah yang dihadapi.

Adapun jaringan syaraf tiruan dan sistem konvensional sendiri tidak saling bersaing, melainkan saling melengkapi, sistem konvensional melakukan proses *control* terhadap jaringan sistem tiruan sehingga memberikan efisiensi yang maksimal, tepat dan memberikan sebuah hasil dengan ketepatan tinggi sesuai dengan tujuan utama penggunaan sistem jaringan syaraf tiruan.

Dari segi performa, JST sudah tidak diragukan lagi dalam pengambilan keputusan dan penentuan perilaku, namun dalam implementasi JST dalam game, tentu saja harus diadakan berbagai macam penyesuaian, baik dari proses, cara kerja, maupun inputan yang diberikan sebagai data awal training, baik secara langsung melalui *gameplay* maupun data yang sudah di implementasikan di dalamnya sebelum *game* dimulai.

3.2 Karakteristik Game

Dalam pembahasan karakteristik *game* akan dijelaskan mengenai *game* sejenis yang menjadi inspirasi dalam proses desain maupun fitur yang terdapat didalamnya, dimana yang menjadi acuan adalah *game* Call Of Duty Modern Warfare 2. Adapun *game* ini sendiri merupakan *game* bergenre *First Person Shooter* (FPS) yang menceritakan tentang perselisihan blok barat dan timur, game ini dilengkapi dengan kemampuan untuk menentukan tingkat kesulitan berdasarkan kemampuan pemain dengan media testing di awal permainan. Berikut adalah beberapa cuplikan permainan dari Call Of Duty Modern Warfare 2 :



Gambar 3.1 tampilan GUI utama Call Of Duty Modern Warfare 2



Gambar 3.2 tampilan *testing range* Call Of Duty Modern Warfare 2

Tingkat kesulitan di dalam *game* ini terbagi menjadi 4, yaitu:

1. Recruit

   Merupakan tingkat kesulitan termudah dalam *game* ini, pemain akan menghadapi BOT dengan tingkat kesulitan termudah dengan *damage* yang kecil dan akurasi yang buruk.

2. Regular

   Tingkat kesulitan dari tingkat ini merupakan "*medium difficulty*" dari *game* ini, merupakan tingkatan yang direkomendasikan bagi *player* yang sudah mengenal game FPS.

3. Hardened

   Tingkat kesulitan yang akan menguji sejauh mana kemampuan *player* dalam menghadapi musuh yang ada, dengan kemampuan BOT yang diklasifikasikan baik, dengan tingkat *damage* yang sama dengan *damage* yang diberikan *player* pada BOT itu sendiri dan dengan tingkat akurasi yang mencapai 80%.

4. Veteran

   Merupakan tingkatan tersulit dalam *game* ini, bentuk dari simulasi *real condition*, yaitu dengan tingkat *health* separuh dari *health* normal, tingkat *damage* yang mencapai 150% *damage* normal dan tingkat akurasi BOT antara 95 – 99%.

Tingkat kesulitan yang akan dimainkan oleh *player* sepanjang alur permainan didapatkan berdasarkan persentase antara waktu penyelesaian *stage* dan akurasi, dengan asumsi semakin singkat waktu penyelesaian *stage* dan semakin baik akurasi yang didapatkan maka akan semakin sulit tingkat kesulitan permainan yang di dapatkan, dan semakin panjang waktu penyelesaian *stage* dan semakin buruk akurasi yang didapatkan, maka akan semakin mudah tingkat kesulitan yang akan didapatkan.

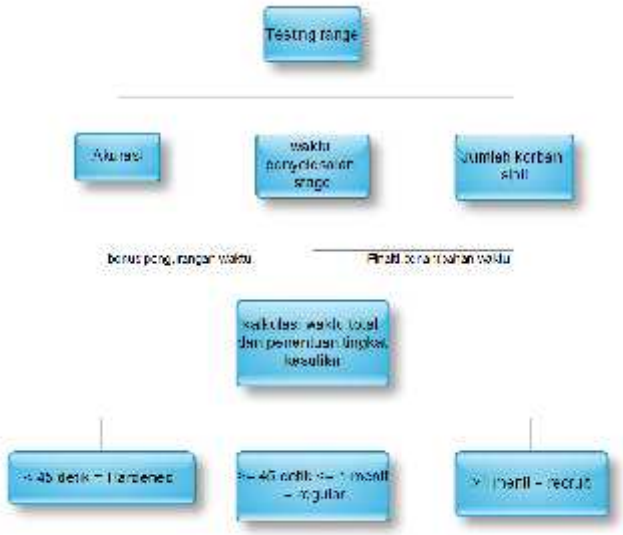
Untuk alur cerita Call Of Duty Modern Warfare 2 yang pada umumnya bertipe *wipe out enemy* game yang dibangun akan menggunakan tipe yang sama untuk alur ceritanya, namun untuk hasil dari *testing range* terdapat perbedaan, yaitu pada game yang dibangun, tingtkat kesulitan tidak didapatkan berdasarkan pengelompokan tingkat kesulitan berdasarkan hasil testing, namun tingkat kesulitan akan disesuaikan dengan kemampuan pemain itu sendiri atau dengan kata lain dinamis menyesuaikan dengan kemampuan pemain itu sendiri.

Untuk input dari *testing arena* dalam game Call Of Duty Modern Warfare 2 sendiri terdiri atas akurasi tembakan, yaitu presentase antara jumlah tembakan yang dilakukan dengan jumlah tembakan yang mengenai target, waktu penyelesaian stage secara keseluruhan, dan juga jumlah korban sipil yang tertembak, dimana dalam kasus ini apabila mengenai target non musuh, maka akan diberikan pinalti berupa penambahan total waktu penyelesaian stage, dan juga otomatis angka korban sipil dalam data test akan bertambah.

Untuk output yang dihasilkan, setelah player menyelesaikan rangkaian test, maka akan muncul rekomendasi tingkat kesulitan yang didapatkan melalui perhitungan dari input yang diperoleh dari hasil testing. Penentuan tingkatan kesulitan dalam permainan Call Of Duty Modern Warfare 2 sangat bergantung pada

waktu penyelesaian stage, hal ini dikarenakan semakin cepat waktu penyelesaian stage, maka semakin baik akurasi dari player dan semakin sedikit korban sipil yang tertembak.

Penentuan keputusan pergerakan dari BOT dalam game Call Of Duty Modern Warfare 2, tidak digunakan nilai parameter tertentu, hanya menggunakan sistem *cover point* dan BOT akan melakukan keputusan pergerakan secara random hanya saja tingkat kecepatan dan rangkaian aksi yang akan dilakukan bot akan berbeda mengikuti tingkat kesulitan yang dipilih oleh player.



Gambar 3.3 Gambaran kalkulasi penentuan tingkat kesulitan Call Of Duty Modern Warfare 2

Penerapan AI yang digunakan dalam game ini terbagi menjadi 2 kelompok, yaitu *enemy AI* dan *friendly AI*, dimana *enemy AI* merupakan AI yang akan menyerang player dan merupakan AI yang harus dieliminasi oleh player, sedangkan *friendly AI* merupakan support bagi *player*, AI yang akan membantu *player* selama berjalanya permainan, baik menunjukan *waypoint* yang harus dilalui maupun membantu *player* dalam mengeliminasi *enemy AI*, penerapan AI di sini bersifat murni, atau dengan kata lain AI sepenuhnya dikendalikan oleh sistem, baik dari segi penentuan keputusan dalam bergerak, menyerang maupun menggunakan senjata yang ada, seluruhnya ditentukan oleh sistem secara random berdasarkan tingkatan kesulitan yang dipilih *player*.

3.3 Analisis Game

Pada analisis *game neural network* digunakan untuk pengenalan pola permainan bagi AI , baik dalam hal penentuan pola gerakan maupun akurasi dalam menyerang yang tentu saja memerlukan data awal sebagai starter untuk dimulainya proses learning yang disebut dengan *data training*.

Data training dalam *game* ini sendiri diperoleh melalui proses testing awal yang dilakukan pada menu awal, dimana selama proses ini terjadi pengumpulan data status dari pemain, baik total tembakan, akurasi, waktu rata-rata, maupun *health point* dari pemain itu sendiri.

3.3.1 Analisis Proses Testing

Proses testing merupakan sebuah tahapan inisialisasi *data training* yang akan digunakan untuk menentukan keputusan AI BOT dalam permainan utama, data training ini hanya digunakan sebagai awalan sebelum *player* melakukan tembakan di dalam permainan utama, karena data training akan kembali di *update* dengan variabel-variabel serupa yang ada di dalam permainan utama.

a. Total Tembakan (X1)

Nilai dari total tembakan yang dilakukan dalam proses testing merupakan sebuah variabel yang didapatkan melalui proses penghitungan dari jumlah peluru yang ditembakan selama berlangsungnya proses testing, baik peluru yang mengenai target sasaran maupun yang meleset, variabel ini didapatkan melalui proses pembacaan input *player* berupa *mouse click event*, atau dalam kata lain pembacaan setiap klik kiri pada mouse yang dilakukan oleh *player* selama berlangsungnya proses testing yang kemudian akan tersimpan secara otomatis ke dalam data training saat *player* melakukan mouse click.

Variabel ini dipilih karena total tembakan yang dilakukan menggambarkan bagaimana *style* serangan tembakan yang dimiliki oleh *player*, yang merupakan salah satu faktor penting dalam penentuan keputusan yang akan diambil oleh AI BOT. Sebagai gambaran, terdapat beberapa *style* yang umum digunakan dalam game bertipe FPS, diantaranya:

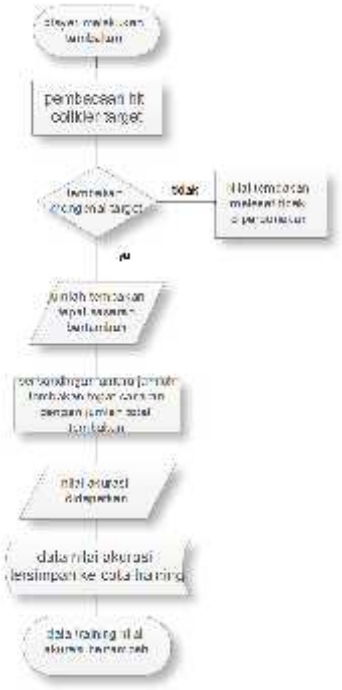
a. Tipe tembakan *on target*, yaitu tembakan secara langsung ke arah target dengan arah yang jelas dan terarah.
b. Tipe tembakan spekulasi, yaitu tipe tembakan dengan menggunakan tembakan percobaan, yang akan menjadi acuan untuk melakukan tembakan selanjutnya.
c. Tipe tembakan *lure to kill*, yaitu tipe tembakan yang dilakukan secara beruntun hingga mengenai target dalam satu waktu.



Gambar 3.4 proses perhitungan jumlah tembakan yang dilakukan *player* dalam proses testing.

b. Akurasi tembakan

Nilai dari akurasi pada proses testing merupakan perbandingan antara tembakan yang mengenai target dengan tembakan yang tidak mengenai target selama proses testing berlangsung, variabel ini diperoleh melalui perbandingan hasil pembacaan *target colider* yang terkena tembakan dengan jumlah tembakan yang dilakukan selama berlangsungnya proses testing yang kemudian akan tersimpan secara otomatis ke dalam data training saat player melakukan *mouse click*, variabel ini dipilih karena akurasi merupakan sebuah "identitas" bagi seorang *player* dalam game FPS selain tipe tembakan, hal ini dikarenakan semakin tinggi tingkat akurasi yang dimiliki oleh seorang player, maka akan semakin sulit untuk melawan player tersebut,yang akan mempengaruhi bagaimana BOT AI menentukan lokasi dan cara bersembunyi dari *player* tersebut yang tentu saja terlepas dari tingkat *awarenes* dari *player* itu sendiri.



Gambar 3.5 proses perhitungan akurasi tembakan yang dilakukan *player* dalam proses testing.

c. Waktu rata rata

Waktu rata-rata pada proses testing merupakan perbandingan antara jumlah stage dalam proses testing dengan total waktu yang diperlukan untuk menyelesaikan keseluruhan *stage*, variabel ini diperoleh dengan cara membaca waktu total keseluruhan yang diperlukan oleh *player* dalam menyelesaikan *stage* testing, dibagi dengan jumlah *stage* yang berjumlah 4 *stage*.

Variabel waktu rata-rata ini dipilih karena waktu yang dibutuhkan dalam menyelesaikan setiap *stage* memberikan gambaran mengenai seberapa cepat dan tepat *player* dalam menembak target yang telah diatur sedemikian rupa dengan pengaturan posisi dan tingkat kesulitan yang berbeda.



Gambar 3.6 proses perhitungan waktu rata-rata penyelesaian stage dalam proses testing

d. Health point

Nilai *health point* dalam proses testing merupakan nilai *health point* yang dimiliki oleh *player* selama proses testing, variabel ini diperoleh melalui proses pembacaan nilai *health point* dari *player* setelah dikurangi dengan kerusakan yang dialami oleh *player* apabila menyentuh objek rintangan yang ada dalam proses testing, variabel ini dipilih karena nilai *health point* yang dimiliki *player* akan menggabarkan kemampuan *player* dalam mengendalikan pergerakan karakter yang dimainkan, dimana *player* dalam hal ini mengendalikan karakter agar tidak mengenai objek rintangan



Gambar 3.7 proses perhitungan *health point* player dalam proses testing

3.3.2 Analisis Proses *In Game*

Dalam proses *in game*, juga terdapat proses pengumpulan data training, memiliki parameter variabel yang sama dengan proses training namun memiliki beberapa proses yang berbeda untuk mendapatkanya, diantaranya:

a. Total tembakan

nilai total tembakan yang dilakukan dalam proses *in game*, variabel ini diperoleh melalui proses perhitungan jumlah *mouse click event* yang dilakukan oleh *player* selama berlangsungnya permainan utama.



Gambar 3.8 proses perhitungan jumlah tembakan yang dilakukan *player* dalam permainan utama

b. Akurasi Tembakan

Nilai akurasi pada permainan utama adalah perbandingan antara jumlah tembakan yang mengenai BOT AI dengan tembakan yang meleset, variabel ini diperoleh melalui perhitungan hasil pembacaan jumlah tembakan yang mengenai *collider* BOT AI dengan tembakan yang meleset, variabel ini dipilih karena akurasi selama berlangsungnya permainan utama didapatkan dengan cara menembak target yang bergerak, yang tentu saja memiliki tingkat kesukaran yang lebih bila dibandingkan dengan proses testing.



Gambar 3.9 proses perhitungan akurasi tembakan dalam permainan utama

c. Waktu rata-rata

Nilai waktu rata-rata dalam permainan utama merupakan waktu yang diperlukan *player* untuk melakukan tembakan yang mengenai BOT AI, variabel ini diperoleh melalui proses pembacaan nilai timer waktu pada saat tembakan player mengenai *collider* BOT AI. Pada proses ini *timer* akan kembali pada posisi awal (nilai 00:00) pada saat tembakan *player* mengenai collider BOT AI untuk kemudian dimulai kembali hingga tembakan *player* kembali mengenai collider BOT AI.



Gambar 3.10 proses perhitungan waktu rata-rata dalam permainan utama

d. Health Point

Nilai *health point* pada permainan utama merupakan nilai *health point* yang dimiliki *player* selama permainan utama berlangsung, variabel ini diperoleh melalui proses perhitungan nilai *health point player* dikurangi dengan *damage* dari serangan BOT AI, variabel ini dipilih karena *Health point* dari *player* selama berlangsungnya permainan akan menentukan tingkat *awareness* dan kejelian *player* dalam menghindari serangan BOT AI maupun dalam serangan yang dilakukan *player* terhadap BOT AI.



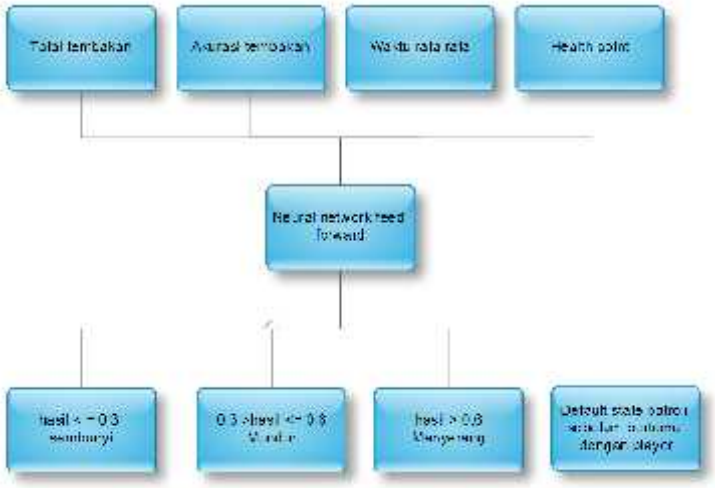
Gambar 3.11 proses perhitungan *health point player* dalam permainan utama

Data yang diperoleh kemudian akan diolah melalui algoritma *neural network multi level feedforward* sehingga dapat muncul *output* yang akan dijadikan acuan dalam penentuan keputusan perilaku bot.*State* yang diapatkan diambil berdasarkan pada pola pada *data training* yang dikombinasikan dengan *data testing*, sebagai gambaran, apabila pada *data training* kondisi total tembakan bejumlah banyak, akurasi rendah, waktu rata rata tinggi dan *health poin* pemain rendah, maka respon yang akan diberikan bot dalam penentuan keputusan akan menjadi cenderung untuk menyerang pemain, hal ini dikarenakan status permainan pemain yang lemah.

Tentu saja hal ini akan terbalik apabila status permainan pemain kuat, yaitu jumlah tembakan sedikit, akurasi tinggi, waktu rata rata rendah dan *health poin* tinggi, maka bot akan cenderung bertahan untuk mengimbangi kemampuan dari pemain itu sendiri, hal ini dimaksudkan untuk memberikan efek *balance* selama proses permainan berlangsung, yang pada umumnya hanya di organisir dengan *sistem dificulty* , atau dalam kata lain dengan tingkat kesulitan yang sudah ditentukan.

Dalam implementasinya algoritma JST akan berperan penuh dalam pengambilan keputusan BOT dalam bergerak, yang dalam kasus ini memiliki 3 parameter, yaitu menyerang, bertahan dan berlari. Namun secara keseluruhan proses pengambilan keputusan sangat ditentukan oleh waktu penyelesaian *stage* dan akurasi.

Selain dari penggunaan JST dalam penentuan keputusan, komponen lain akan ditangani oleh bantuan sistem, termasuk penentuan akurasi BOT, gerakan yang dilakukan dan kecepatan gerakan BOT, hal ini dikarenakan pada kasus ini AI yang dibangun hanya terfokus untuk penentuan keputusan yang akan diambil BOT saat bertemu dengan *player*.



Gambar 3.12 Gambaran kalkulasi penentuan aksi BOT dengan menggunakan JST *Multi level feed forward*

Adapun di dalam permainan, terdapat paramater lain yang digunakan sebagai pendukung neural network, salahsatu di antaranya adalah parameter jarak serang, parameter ini digunakan sebagai batas jarak *trigger* AI BOT untuk melakukan interaksi dengan player, dengan kata lain sebagai jarak batas minimal antara player dan AI BOT agar AI BOT memberikan respon, yang bertujuan untuk membatasi jarak serang AI BOT . Berikut adalah gambaran flowchart interaksi AI BOT dan Player selama permainan berlangsung:



Gambar 3.13 Gambaran proses terjadinya interaksi antara AI BOT dan player

3.3.3 Analisis Masukan

Dalam implementasi algoritma neural network feed forward pada AI game FPS yang akan menjadi inputan adalah akurasi menembak dan waktu respon dalam proses *testing* game untuk mendapatkan data input *learning*.

Adapun penjelasan mengenai inputan yang akan dijadikan bahan dalam proses learning yaitu :

a. Total tembakan (X1)

Total tembakan diperoleh melalui proses perhitungan total seluruh tembakan yang dilakukan selama proses *testing* dengan pembagi 100 dalam sebuah stage, berikut adalah contoh perbandingan yang didapatkan:

5 kali menembak dalam 1 stage = 5/100 (0.05)

12 kali menembak dalam 1 stage = 12/100 (0.12)

15 kali menembak dalam 1 stage = 15/100 (0.15) Dst.

Adapun untuk nilai yang didapatkan dalam proses *In Game* **akan ditambahkan secara langsung ke dalam nilai total tembakan** yang didapatkan dalam proses testing, misal:

30 kali menembak dalam proses *testing* = 30/100 (0.3)

7 kali menembak dalam proses *In Game* = (30+7)/100 (0.37)

b. Akurasi tembakan (X2)

Akurasi didapatkan melalui proses penghitungan perbandingan antara jumlah tembakan meleset dan jumlah tembakan total yang dilakukan dalam sebuah *stage*, berikut adalah beberapa contoh perbandingan yang didapatkan

1 kali mengenai target : 5 kali total tembakan = 1/5 (20% atau 0.2)

1 kali mengenai target : 2 kali total tembakan = ½ (50% atau 0.5)

1 kali mengenai target : 3 kali total tembakan = 1/3 (≈33% atau 0.33)

Untuk nilai akurasi yang didapatkan dalam proses *In Game* **akan di ambil nilai rata rata dari hasil penjumlahan nilai akurasi proses *Testing* dan proses *in Game***, misal:

1 kali mengenai target testing : 5 kali total tembakan = 1/5 (20%)

1 kali mengenai target In game : 1 kali total tembakan = 1/1 (100%)

Maka nilai akurasi yang didapatkan adalah (20+100)/2 = 60%

c. Waktu rata-rata (X3)

Waktu rata rata didapatkan melalui proses penghitungan selisih watu antara awal pemain memasuki stage hingga menyelesaikan *stage*, misalnya:

Pemain memasuki *stage* pada detik pertama dan menyelesaikan *stage* pada detik ke 40 maka perhitunganya adalah

40-1 = 39 detik (dalam satuan menit 39/60 = ~0.65

Untuk nilai rata-rata yang didapatkan dalam proses *in game*, selanjutnya akan di ambil **nilai rata-rata antara nilai waktu rata-rata *in game* dengan nilai rata-rata yang diperoleh pada saat *testing*** misal:

Waktu rata-rata yang diperoleh selama proses *testing*

40-1 = 39 detik (dalam satuan menit 39/60 = ~0.65)

Waktu rata rata yang diperoleh dalam proses *in game*

10-1 = 9 detik (dalam satuan menit 9/60 = 0.15)

Maka waktu rata rata yang akan diperoleh adalah

(0.65 + 0.15)/2 = 0.4

d. *Health point* (X4)

Health point didapatkan melalui proses penghitungan *health point* yang tersisa dari *player* dengan pembagi 100, berikut adalah contoh perbandingan yang dilakukan:

Apabila *health point* yang tersisa 45 maka *health point* = 45/100 (0.45)

Apabila health *point* yang tersisa 65 maka *health point* = 65/100 (0.65)

Dst..

Sedangkan untuk nilai dari *Health Point* yang diperoleh selama proses *In Game*, akan diproses dengan pengambilan **nilai rata-rata antara nilai *Health Point* proses *testing* dan nilai *Health Point* proses *In Game***, misal:

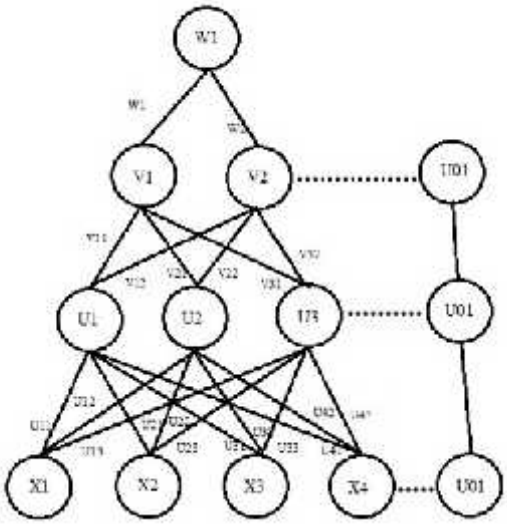
Health point proses *training* tersisa 65 maka *health point* = 65/100 (0.65)

Health point In Game tersisa 85 maka *health point* = 85/100 (0.85)

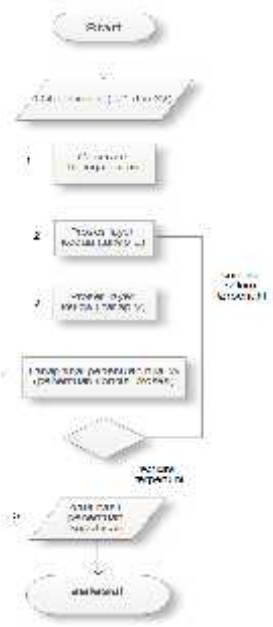
Maka *Health point* yang akan diperoleh adalah (0.65 + 0.85) /2 = 0.75

3.4 Analisis Metode

Pada bagian analisis metode implementasi algoritma JST pada AI BOT parameter yang akan digunakan adalah akurasi, waktu respon dan waktu menyelesaikan *stage*, dimana semua nilai yang didapatkan akan diproses untuk mendapatkan pengambilan keputusan yang optimal, adapun arsitektur yang ada didalamnya adalah sebagai berikut:



Gambar 3.14 Neural network feed forward dengan 4 layer



Gambar 3.15 gambaran dari tahapan proses yang dilakukan

Dengan 4 *layer*, dan 4 *hidden layer* memungkinkan hasil yang optimal dengan proses yang cukup singkat yang diharapkan dapat memberikan respon yang lebih baik.

1.proses *generate* bilangan random untuk bobot neuron dan bias

- Bias

  U_{0j} = 0, 92943

- Bobot

U11 = 0,52493	U21= 0,44544	W11 = 0,20073
U12 = -0,22733	U22 = -0,13356	W21 = 0,05074
U13 = 0,73582	U23 = 0,29151	
U31 = 0,38764	U41 = 0,90443	
U32 = 0,45899	U42 = 0,38949	
U33 = 0,84993	U43 = 0,68898	
V11 = -0,69586	V21 = 0,23457	V31 = 0,10605
V12 = 0,74686	V22 = -0,19096	V32 = 0,69586

Untuk data *learning* yaitu :

X1 = 0,75 **X2 = 0,5** **X3 = 0,65** **X4 = 0,5**

Nilai eror 0,3

2. pencarian nilai nilai untuk layer kedua, dimisalkan dengan U1,U2 dan U3 yaitu:

$$U_in = U_{0j} + \sum_{i=1}^{n} X_i \, U_{ij}$$

$$U = 1/1+e^{U_in}$$

U1_in = 0,92943 + {(0,75 x 0,52493) + (0,5 x 0,44544) + (0,65 x 0,38764) + (0,5 x 0,90443) } = 2,25003

U1 = f(U1_in) = 0,90465

U2_in = 0,92943 + {(0,75 x 0,52493) + (0,5 x 0,44544) + (0,65 x 0,45899) + (0,5 x 0,38949) } = 2,03894

U2 = f(U2_in) = 0,88482

U3_in = 0,92943 + {(0,75 x 0,52493) + (0,5 x 0,44544) + (0,65 x 0,84993) +
(0,5 x 0,68898) } = 2,2428

U3 = f(U3_in) = 0,90403

3. **Kemudian pencarian nilai nilai untuk layer ketiga, dimisalkan dengan V1 dan V2 yaitu:**

$$V_in = V_{0j} + \sum_{i=1}^{n} X_i V_{ij}$$

$$V = 1/1+e^{-V_in}$$

V1_in = 0,92943+{(0,09534 x -0,69586) + (0,11518 x 0,23457) +
(0,07997 x 0,10605)} = 0,89859

V1 = f(V1_in) = 0,71065

V2_in = 0,92943+((0,17568 x 0,74686) + (0,33355 x -0,19096) +
(0,16423 x 0,69586)) = 1,11122

V2 = f(V2_in) = 0,75235

4. **Terakhir, pencarian nilai nilai untuk layer keempat, dimisalkan dengan W1 yaitu:**

$$W_in = W_{0j} + \sum_{i=1}^{n} X_i W_{ij}$$

$$W = 1/1+e^{-W_in}$$

W1_in = 0,92943 + ((0,28934 x 0,20073)+(0,24764 x 0,05074))
= 1,00007

W= f(W1_in) = 0,73107

MSE = 0,5 (Σ ($T_{kn} - W_{kn}$)^2) = 0,26733

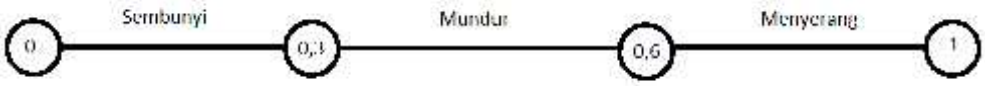
Proses iterasi terhenti karena nilai MSE telah memenuhi syarat < 0,3

Epoch = 1

5. **Data hasil *epoch* tersebut akan di absolutkan dan selanjutnya akan diproses untuk pengambilan keputusan**, berikut adalah ketentuan dalam pengambilan keputusan yang dilakukan:

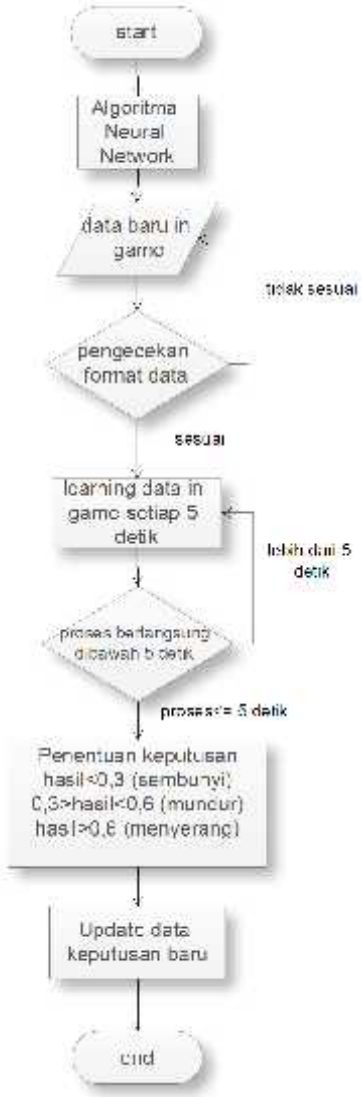
- Apabila nilai yang didapat < 0.3 maka bot akan bersembunyi
- Apabila nilai yang didapat > 0.3 dan < 0.6 maka bot akan mundur
- Apabila nilai yang didapat > 0.6 maka bot akan menyerang

Penentuan batasan diatas dilakukan dengan menggunakan sistem perbandingan antara range nilai dan jumlah keputusan yang diambil, dengan *range* antara 0 hingga 1 dengan 3 keputusan maka *range* terbagi menjadi 3 bagian, berikut ilustrasinya:



Maka berdasarkan hasil perhitungan, keputusan yang diambil adalah menyerang, karena nilai yang didapat bernilai 0,99 , karena berada di wilayah 0,6 sampai dengan 1.

Dalam proses updating data learning AI BOT selama permainan berlangsung, update dilakukan secara berkala setiap 5 detik sekali setelah proses learning berakhir di setiap periodenya, dengan cara re-write (menimpa) data yang sudah ada sebelumnya. Dari sisi AI bot, pada setiap akhir periode dilakukan proses refresh data learning sehingga data yang digunakan selalu merupakan data terbaru yang dihasilkan oleh algoritma neural network, berikut adalah flow chart proses updating data keputusan AI BOT:



Gambar 3.16 Gambaran proses updating data keputusan dengan menggunakan JST *Multi level feed forward*

3.5 Analisis dan 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 4
3. *3ds Max* 2012
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*

3.6 Analisis dan 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 pengembang untuk membangun *game* ini adalah sebagai berikut :

1. *Processor Core 2 Duo* 2,13 Ghz
2. RAM 512 MB
3. VGA 512 MB
4. *Harddisk* 10 GB
5. *Mouse* dan *Keyboard*

Perangkat keras minimum bagi pengguna untuk memainkan *game* ini adalah sebagai berikut :

1. *Dual Core* 1,80 Ghz
2. RAM 256 MB
3. VGA 512 MB
4. *Harddisk* 1 GB
5. *Mouse* dan *Keyboard*

3.7 Analisis Pengguna

Pada analisis pengguna ini akan mencakup analisis beberapa parameter terhadap calon pemain dari aplikasi. *Game* yang akan dibangun ini dapat dimainkan oleh berbagai kalangan.

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

1. *User Knowledge and Experience*

Berikut ini klasifikasi *knowledge adn experience* dari pengguna aplikasi.

Tabel 3.1 Analisis klasifikasi *knowledge and experience*

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

2. *Users Physical Characteristic*

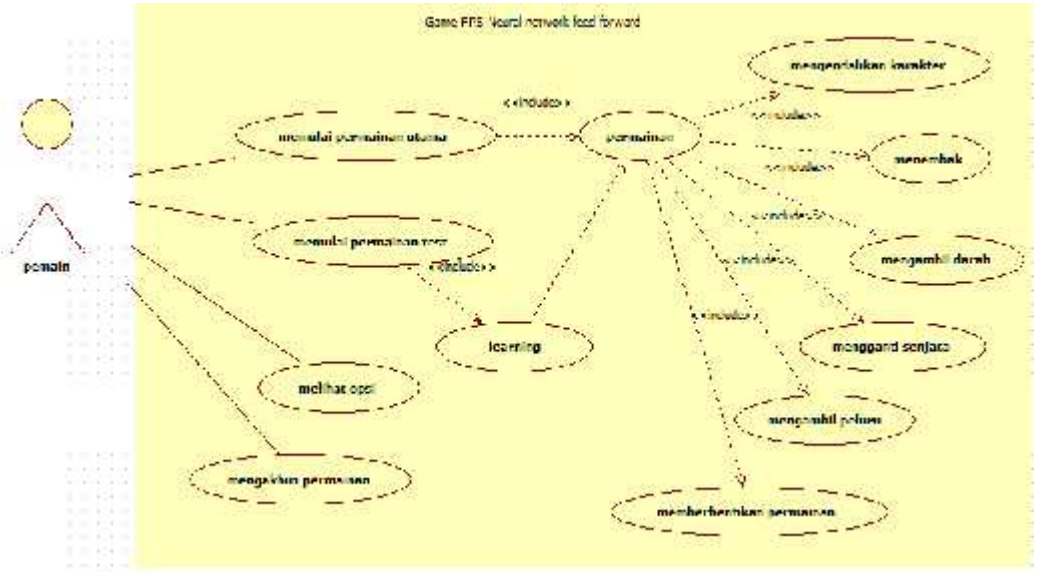
Keadaan fisik seseorang mungkin akan berpengaruh pada penggunaan aplikasi *game* ini. Ada hal-hal yang harus diperhatikan juga terhadap *user* dari karakteristik fisiknya untuk dapat menggunakan aplikasi ini yaitu, *Color Blind*, *Handednes*, dan *Gender*.

Tabel 3.2 Analisis *Users Physical Characteristic*

Age	Semua umur
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.

3.8 Use Case Diagram

Use case atau diagram *use case* merupakan pemodelan untuk prilaku *(behavior)* sistem informasi yang akan dibuat. *Use case* mendeskripsikan sebuah interaksi antara satu atau lebih aktor dengan sistem informasi yang akan dibuat.Di bawah ini adalah perancangan *use case* yang terdapat pada *game first person shooter* Neural network feed forward.



Gambar 3.17 *Use Case Diagram*

3.9 Definisi Use Case

Definisi *use case* berfungsi untuk menjelaskan proses yang terdapat pada setiap *use case*. Berikut di bawah ini tabel dari definisi *use case* yaitu

Tabel 3.3 Definisi *Use Case*

No	*Use Case*	Deskripsi
1	Memulai Permainan	Proses untuk memulai permainan
2	Melihat opsi	Proses untuk meilhat petunjuk permainan dan control permainan
3	Mengakhiri Permainan	Proses untuk keluar dari permainan atau mengakhiri permainan
4	Permainan	Proses ketika pemain telah masuk ke dalam permainan.
5	Berjalan	Proses untuk menggerakan pemaian melalui interaksi dengan *keyboard*
6	Meloncat	Proses untuk membuat pemain melakukan tindakan meloncat melalui interaksi dengan *keyboard*
7	Menembak	Proses untuk menembakan peluru yang terdapat pada senjata
8	Mengambil Darah	Proses untuk menambah darah / *healt* ketika darah tersisa sedikit lagi
9	Mengganti Senjata	Proses untuk menukar senjata
10	Mengambil Peluru	Proses untuk menambah peluru senjata
11	Memberhentikan Permainan	Proses untuk menghentikan sementara permainan

3.10 Skenario Use Case

Skenario *use case* merupakan bagian pada *use case* yang menunjukkan proses apa saja yang terjadi pada setiap bagian di dalam *use case*, 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*. Berikut ini beberapa skenario *use case* berdasarkan *use case* yang ada yaitu :

Tabel 3.4 Skenario *Use Case* Memulai Permainan

Identifikasi	
Nomor	1
Nama	Memulai Permainan
Tujuan	Melihat *scene* loading dan masuk ke dalam permainan
Deskripsi	Proses untuk melihat *scene* cerita permainan dan masuk ke dalam
Aktor	Pemain
Skenario Utama	
Kondisi awal	Pemain berada di menu utama
Aksi Aktor	Reaksi Sistem
Memilih menu mainkan	
	Menampilkan *scene* loading tentang latar belakang permainan
	Menampilkan permainan
Kondisi Akhir	Sistem menampilkan permainan

Tabel 3.5 Skenario *Use Case* Melihat opsi

Identifikasi	
Nomor	2
Nama	Melihat Informasi opsi
Tujuan	Menampilkan informasi opsi
Deskripsi	Proses untuk menampilkan informasi opsi
Aktor	Pemain
Skenario Utama	
Kondisi awal	Pemain berada di menu utama
Aksi Aktor	Reaksi Sistem
Memilih menu petunjuk	
	Menampilkan informasi tentang opsi *keyboard* dan *mouse*.
Kondisi Akhir	Sistem akan menampilkan informasi opsi

Tabel 3.6 Skenario *Use Case* Mengakhri Permainan

Identifikasi	
Nomor	3
Nama	Mengakhiri Permainan
Tujuan	Keluar dari permainan
Deskripsi	Proses untuk mengakhiri permainan
Aktor	Pemain
Skenario Utama	
Kondisi awal	Pemain berada di menu utama
Aksi Aktor	Reaksi Sistem
Memilih menu keluar	
	Keluar dari permainan atau mengakhiri permainan
Kondisi Akhir	Sistem akan keluar dari permainan

Tabel 3.7 Skenario *Use Case* Permainan

Identifikasi	
Nomor	4
Nama	Permainan
Tujuan	permainan

Deskripsi	Proses ketika pemain telah masuk ke dalam permainan	
Aktor	Pemain	
Skenario Utama		
Kondisi awal	Sistem menampilkan permainan	
Aksi Aktor		Reaksi Sistem
		Sistem menampilkan permainan
Pemain mulai memainkan permainan		
Pemain berhasil menyelesaikan misi		
		Sistem merespon misi yang telah selesai
Pemain gagal menyelesaikan misi yang ada di dalam permainan		
		Sistem kembali ke awal permainan
Kondisi Akhir	Sistem akan menampilkan tampilan misi selesai jika pemain berhasil menyelesaikan permainan dan sistem akan kembali ke awal permainan jika pemain gagal menyelesaikan misi permainan	

Tabel 3.8 Skenario *Use Case* Berjalan

Identifikasi	
Nomor	5
Nama	Berjalan
Tujuan	Menggerakan karakter atau tokoh utama
Deskripsi	Proses untuk menggerakan karakter atau tokoh utama yang dimainkan dalam permainan menggunakan

	keyboard.	
Aktor	Pemain	
Skenario Utama		
Kondisi awal	Pemain menekan tombol pada *keyboard*	
Aksi Aktor		Reaksi Sistem
Pemain menekan tombol W		
		Sistem menggerakan karakter ke depan
Pemain menekan tombol D		
		Sistem menggerakan karakter ke kanan
Pemain menekan tombol A		
		Sistem menggerakan karakter ke kiri
Pemain menekan tombol S		
		Sistem menggerakan karakter ke belakang
Pemain menekan tombol W dan A secara bersamaan		
		Sistem menggerakan karakter maju serong ke kiri
Pemain menekan tombol W dan D secara bersamaan		
		Sistem menggerakan karakter maju serong ke kanan
Pemain menekan tombol S dan D secara bersamaan		
		Sistem menggerakan karakter mundur serong ke kanan
Pemain menekan tombol S dan A secara bersamaan		
		Sistem menggerakan karakter mundur serong ke kiri
Kondisi Akhir		Sistem akan menampilkan karakter atau tokoh utama melakukan pergerakan berupa berjalan.

Tabel 3.9 Skenario *Use Case* Meloncat

Identifikasi	
Nomor	6
Nama	Meloncat
Tujuan	Menggerakan karakter atau tokoh utama
Deskripsi	Proses untuk menggerakan karakter atau tokoh utama yang dimainkan dalam permainan menggunakan tombol *keyboard*.
Aktor	Pemain
Skenario Utama	
Kondisi awal	Pemain menekan tombol spasi pada *keyboard*
Aksi Aktor	Reaksi Sistem
Pemain menekan tombol spasi pada *keyboard*	
	Karakter atau tokoh utama dalam permainan akan meloncat sesuai dengan inputan dari pemain
Kondisi Akhir	Sistem akan menampilkan karakter atau tokoh utama melakukan loncatan

Tabel 3.10 Skenario *Use Case* Menembak

Identifikasi	
Nomor	7
Nama	Menembak
Tujuan	Melakukan tembakan

Deskripsi	Proses untuk melakukan tembakan	
Aktor	Pemain	
Skenario Utama		
Kondisi awal	Pemain menekan tombol kiri *mouse*	
Aksi Aktor		Reaksi Sistem
Pemain menekan tombol pada kiri *mouse*		
		Sistem mengeluarkan peluru dari senjata yang digunakan
Kondisi Akhir	Pemain menembak dengan menggunakan tombol kiri *mouse*	

Tabel 3.11 Skenario *Use Case* Mengambil Darah

Identifikasi		
Nomor	8	
Nama	Mengambil Darah	
Tujuan	Menambah darah ketika darah berkurang	
Deskripsi	Proses menambahkan darah ketika darah pemain berkurang	
Aktor	Pemain	
Skenario Utama		
Kondisi awal	Pemain mengambil darah	
Aksi Aktor		Reaksi Sistem
Pemain mengambil darah		
		Jumlah health poin pemain bertambah
Kondisi Akhir		Sistem menampilkan darah yang bertambah dan objek health pack menghilang

Tabel 3.12 Skenario *Use Case* Mengganti Senjata

Identifikasi		
Nomor	9	
Nama	Mengganti Senjata	
Tujuan	Mengganti senjata dengan senjata laras panjang atau senjata laras pendek	
Deskripsi	Proses mengganti senjata dengan pilihan senjata laras panjang atau senjata laras pendek	
Aktor	Pemain	
Skenario Utama		
Kondisi awal	Sistem menampilkan permainan	
Aksi Aktor		Reaksi Sistem
		Sistem menampilkan permainan
Pemain memainkan permainan		
Pemain menggunakan senjata		
Pemain menekan tombol 1 jika ingin menggunakan senjata laras pendek		
		Sistem menampilkan karakter dengan senjata laras pendek
Pemain menekan tombol 2 jika ingin menggunakan senjata laras panjang		
		Sistem menampilkan karakter dengan senjata laras panjang
Kondisi Akhir		Sistem menampilkan senjata yang dipilih *player*

Tabel 3.13 Skenario *Use Case* Mengambil Peluru laras pendek

Identifikasi	
Nomor	10
Nama	Mengambil Peluru laras pendek
Tujuan	Menambah peluru senjata laras pendek
Deskripsi	Proses dimana pemain menambahkan peluru laras pendek
Aktor	Pemain
Skenario Utama	
Kondisi awal	Pemain mengambil darah
Aksi Aktor	Reaksi Sistem
Pemain mengambil health pack	
	Sistem menampilkan *icon* health pack
	Sistem melakukan pengecekan
	Apabila darah pemain penuh, sistem tidak mengalami perubahan apapun
	Apabila darah pemain tidak penuh, sistem menampilkan darah pemain yang bertambah
	Apabila darah pemain tidak penuh, sistem menghilangkan *icon* health pack yang telah diambil oleh pemain
Kondisi Akhir	Sistem menampilkan *darah* yang bertambah atau tidak dan menampilkan *icon* health pack yang menghilang atau tidak menghilang.

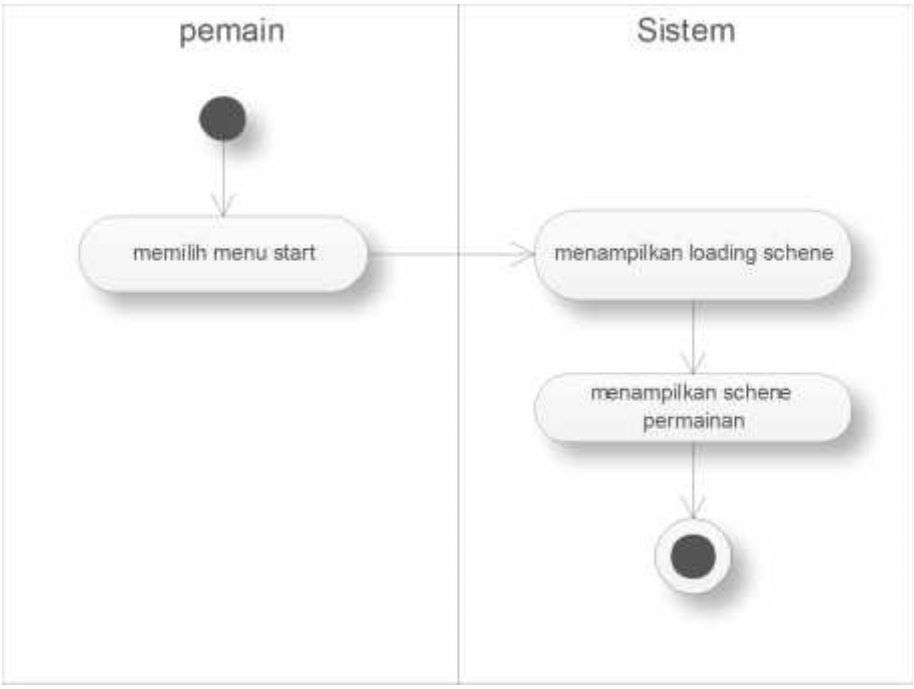
Tabel 3.14 Skenario *Use Case* Memberhentikan Permainan

Identifikasi		
Nomor	11	
Nama	Memberhentikan Permainan	
Tujuan	Melihat petunjuk permainan dan informasi kontrol atau keluar dari permainan	
Deskripsi	Proses ketika pemain ingin melihat petunjuk permainan dan informasi kontrol atau keluar dari	
Aktor	Pemain	
Skenario Utama		
Kondisi awal	Sistem menampilkan permainan	
Aksi Aktor		Reaksi Sistem
Pemain menekan tombol P		
		Sistem menampilkan *menu* jeda
		Sistem menampilkan petunjuk untuk keluar dari permainan dan kembali ke menu utama
Pemain menekan kiri *mouse* di sembarang arena permainan		
		Sistem kembali ke permainan
Pemain menekan tombol P		
		Sistem kembali ke menu utama
Kondisi Akhir	Sistem menampilkan menu jeda	

3.11 Activity diagram

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

A. *Activity Diagram* Proses Memulai Permainan

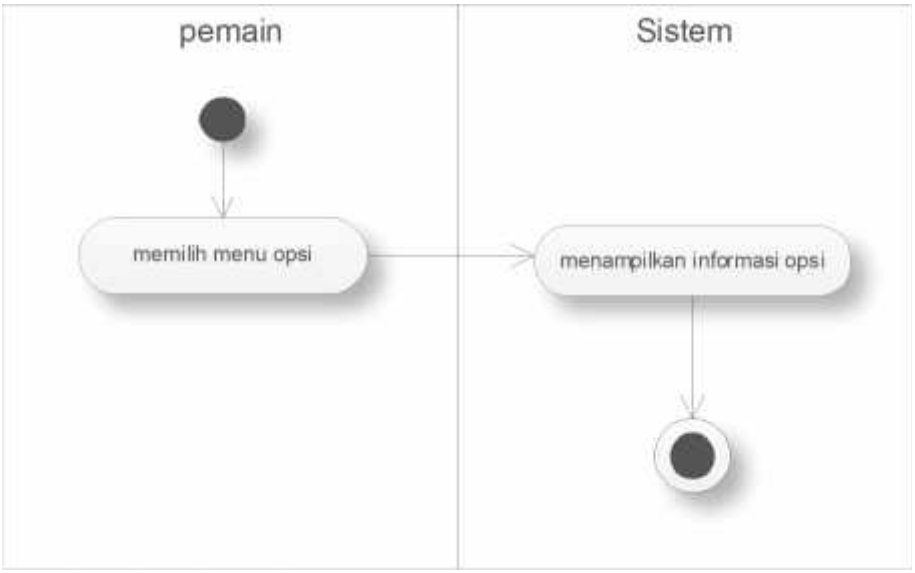


Gambar 3.18 *Activity Diagram* Memulai Permainan

Tabel 3.15 Penjelasan Proses Memulai Permainan

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

B. *Activity Diagram* Proses Melihat Informasi opsi

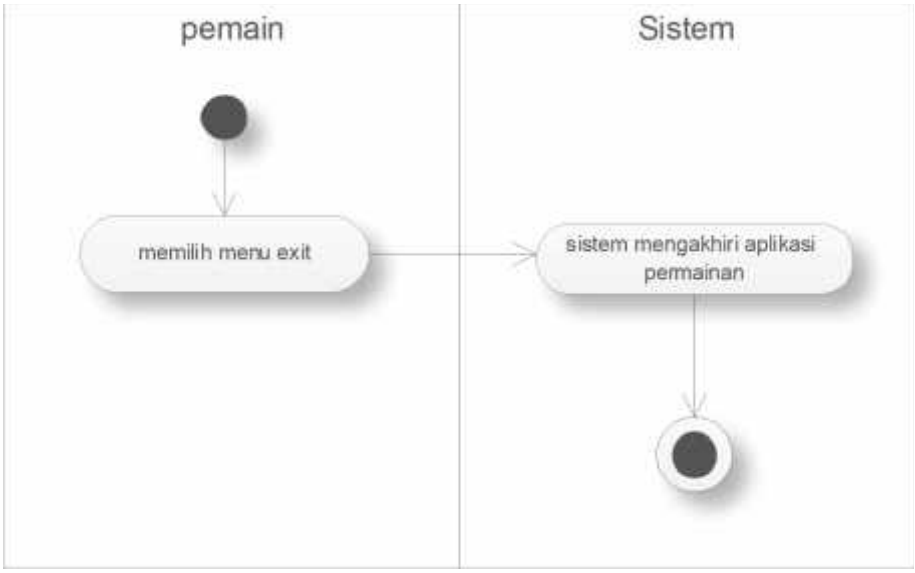


Gambar 3.19 *Activity Diagram* Melihat Informasi opsi

Tabel 3.16 Penjelasan Proses Melihat Informasi opsi

Deskripsi	Menu petunjuk digunakan pemain untuk melihat petunjuk permainan
Normal Flow	1. Pemain memilih Menu petunjuk di menu utama 2. Sistem menampilkan opsi

C. *Activity Diagram* Mengakhiri Permainan

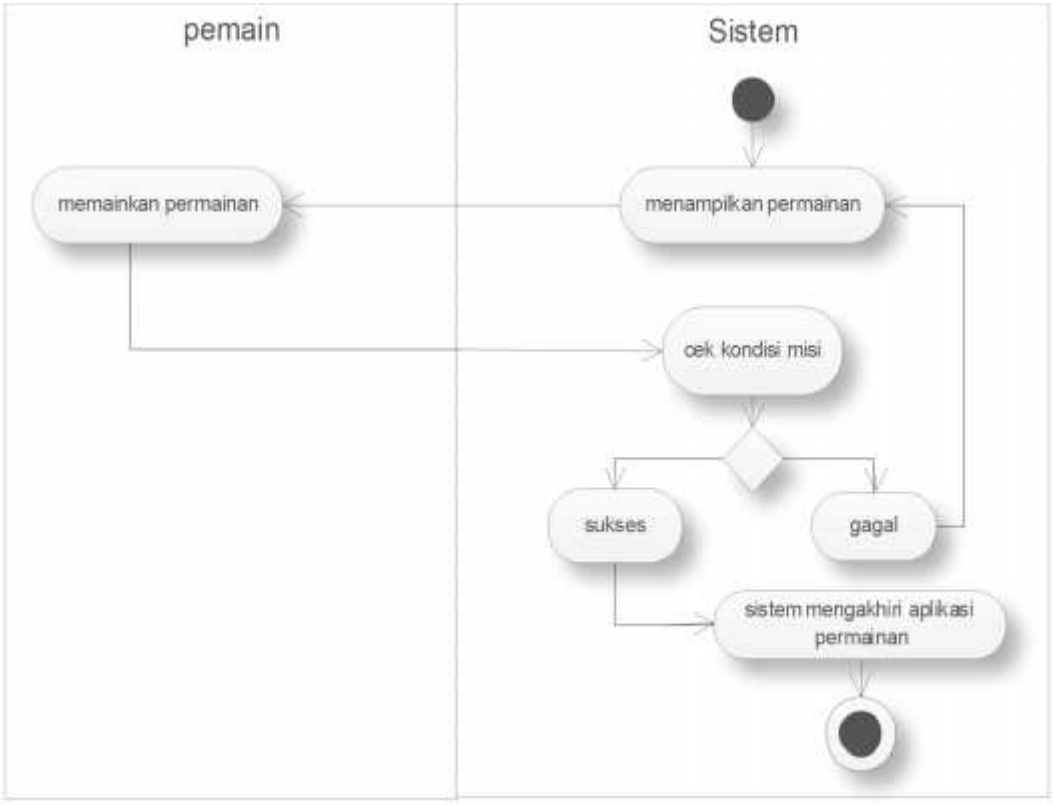


Gambar 3.20 *Activity Diagram* Mengakhiri Permainan

Tabel 3.17 Penjelasan Proses Mengakhiri Permainan

Deskripsi	Menu keluar merupakan menu yang digunakan untuk keluar dari permainan
Normal Flow	1. Pemain memilih menu keluar pada menu utama 2. Sistem akan mengakhiri permainan

1. Activity *Diagram* Permainan

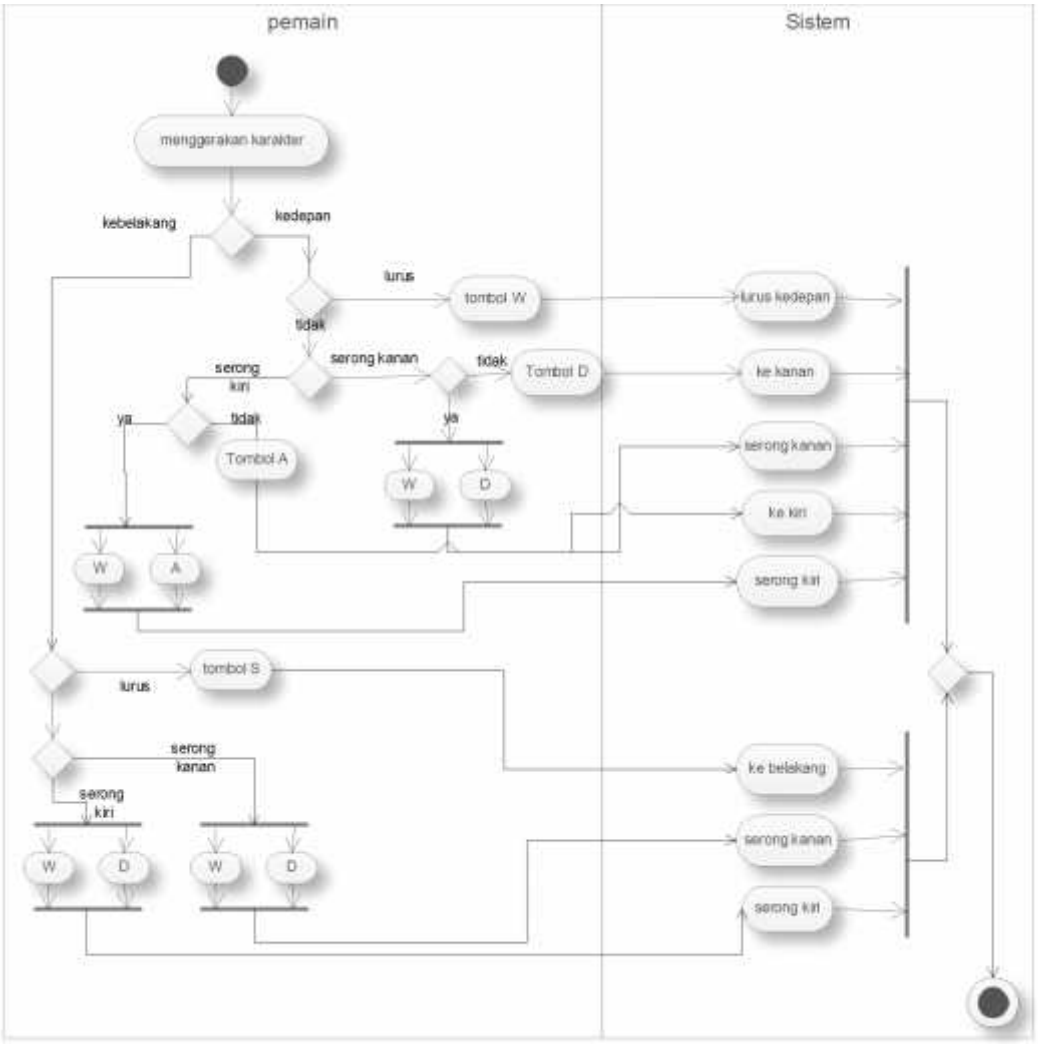


Gambar 3.21 *Activity Diagram* Permainan

Tabel 3.18 Penjelasan Proses Permainan

Deskripsi	Permainan merupakan kondisi pemain ketika telah berada di dalam permainan.
Normal Flow	1. Sistem menampilkan permainan 2. Pemain mulai memainkan permainan 3. Pemain berhasil menyelesaikan misi yang dihadapi 4. Sistem menampilkan tampilan misi selesai 5. Pemain gagal menyelesaikan misi 6. Sistem kembali ke awal permainan

D. *Activity Diagram* Berjalan

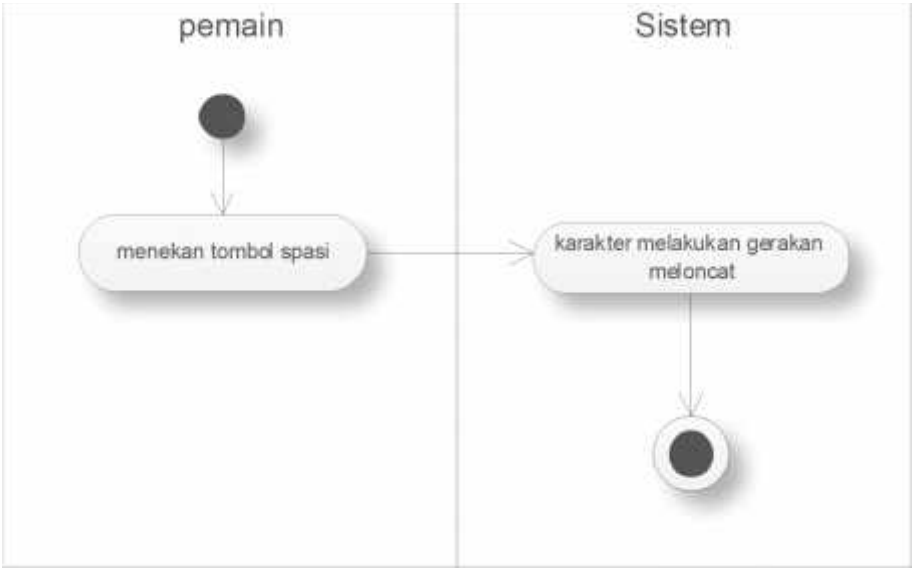


Gambar 3.22 *Activity Diagram* Berjalan

Tabel 3.19 Penjelasan Proses Berjalan

Deskripsi	Proses untuk menggerakan karakter atau tokoh utama yang dimainkan dalam permainan menggunakan tombol *keyboard*.
Normal Flow	1. Pemain menggerakan karakter 2. Pemain menekan tombol W 3. Karakter berjalan ke depan 4. Pemain menekan tombol W dengan A secara bersamaan 5. Karakter berjalan maju ke sebelah kiri 6. Pemain menekan tombol A 7. Karakter berjalan ke kiri 8. Pemain menekan tombol W dengan D secara bersamaan 9. Karakter berjalan maju ke kanan 10. Pemain menekan tombol D 11. Karakter berjalan ke kanan 12. Pemain menekan tombol S 13. Karakter berjalan lurus mundur 14. Pemain menekan tombol S dengan D secara bersamaan 15. Karakter berjalan mundur ke kanan 16. Pemain menekan tombol S dengan A secara bersamaan 17. Karakter berjalan mundur ke kiri

E. *Activity Diagram* Meloncat

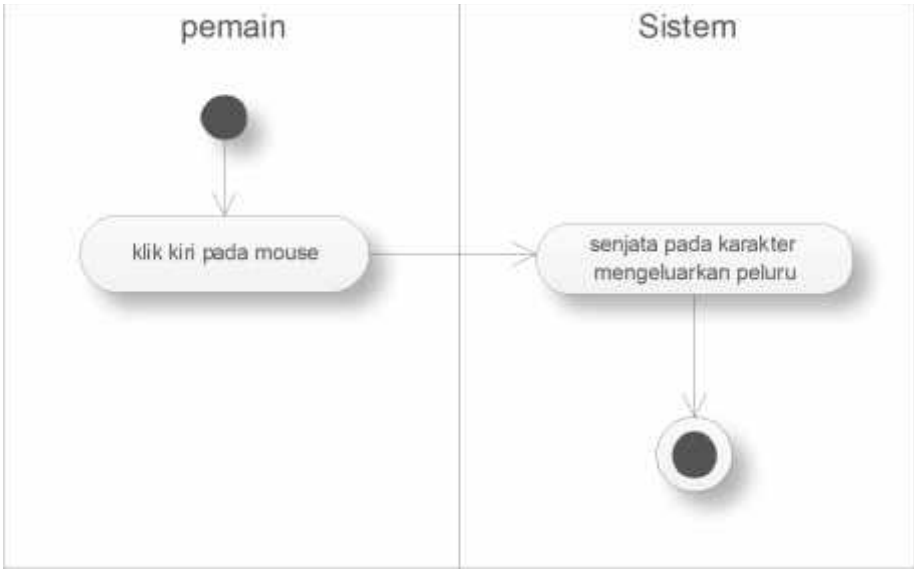


Gambar 3.23 *Activity Diagram* Meloncat

Tabel 3.20 Penjelasan Proses Meloncat

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

G. *Activity Diagram* Menembak

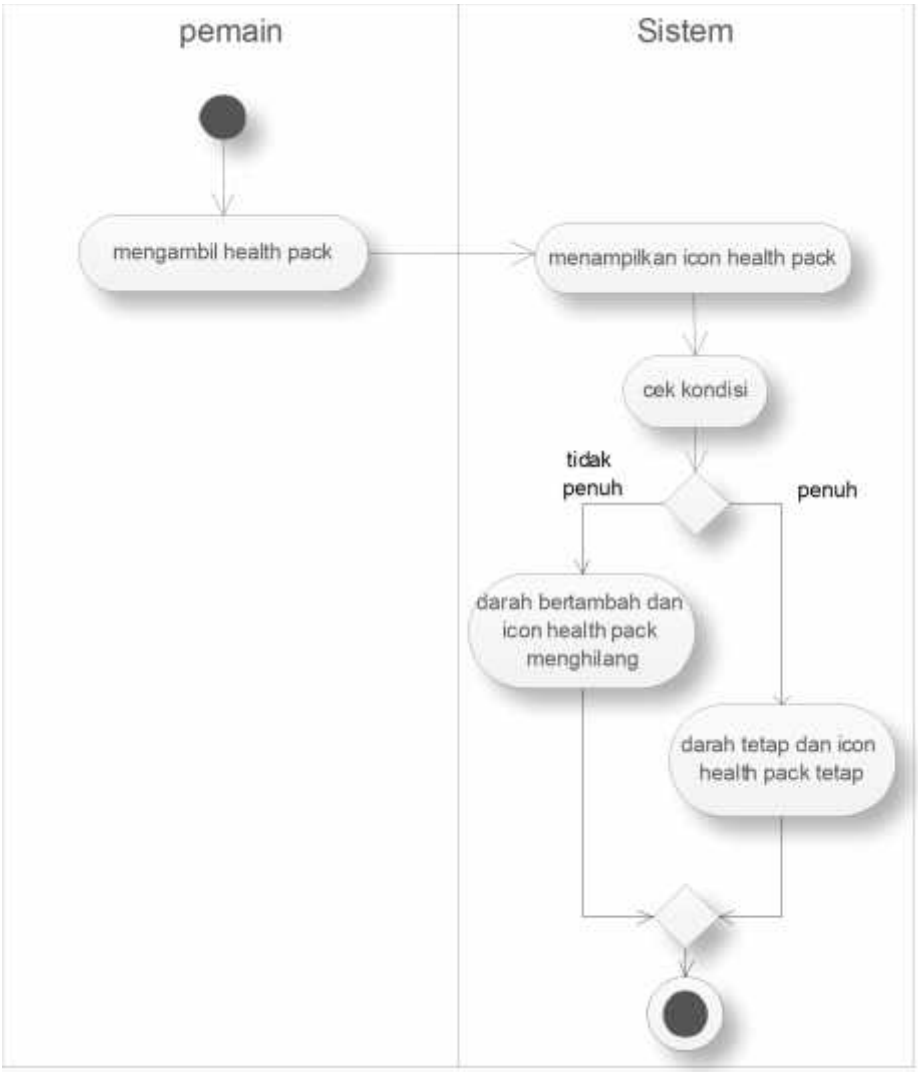


Gambar 3.24 *Activity Diagram* Menembak

Tabel 3.21 Penjelasan Proses Melakukan Tembakan

Deskripsi	Proses untuk melakukan tembakan
Normal Flow	1. Pemain menekan tombol kiri pada *mouse* 2. Sistem menampilkan senjata yang mengeluarkan peluru

H. *Activity Diagram* Mengambil Darah

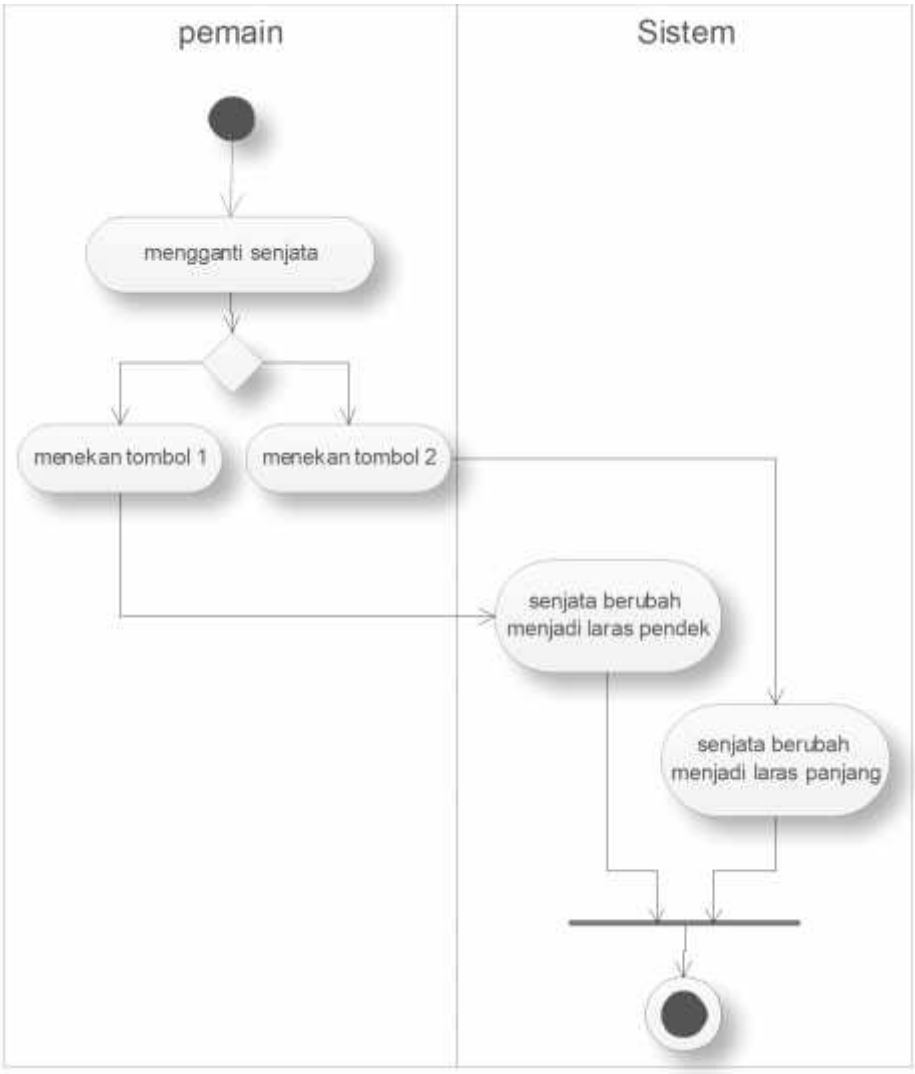


Gambar 3.25 *Activity Diagram* Mengambil Darah

Tabel 3.22 Penjelasan Proses Mengambil Darah

Deskripsi	Proses dimana pemain mengambil darah
Normal Flow	1. Pemain mengambil darah 2. Sistem menampilkan *icon* darah 3. Sistem melakukan pengecekan 4. Apabila darah pemain penuh, sistem menampilkan *bar* darah yang tidak berubah 5. Apabila darah pemain penuh, sistem menampilkan *icon* darah yang tidak menghilang 6. Apabila darah pemain tidak penuh, sistem menampilkan *bar* darah pemain yang bertambah 7. Apabila darah pemain tidak penuh, sistem menghilangkan *icon* darah yang telah diambil oleh pemain

I. *Activity Diagram* Mengganti Senjata



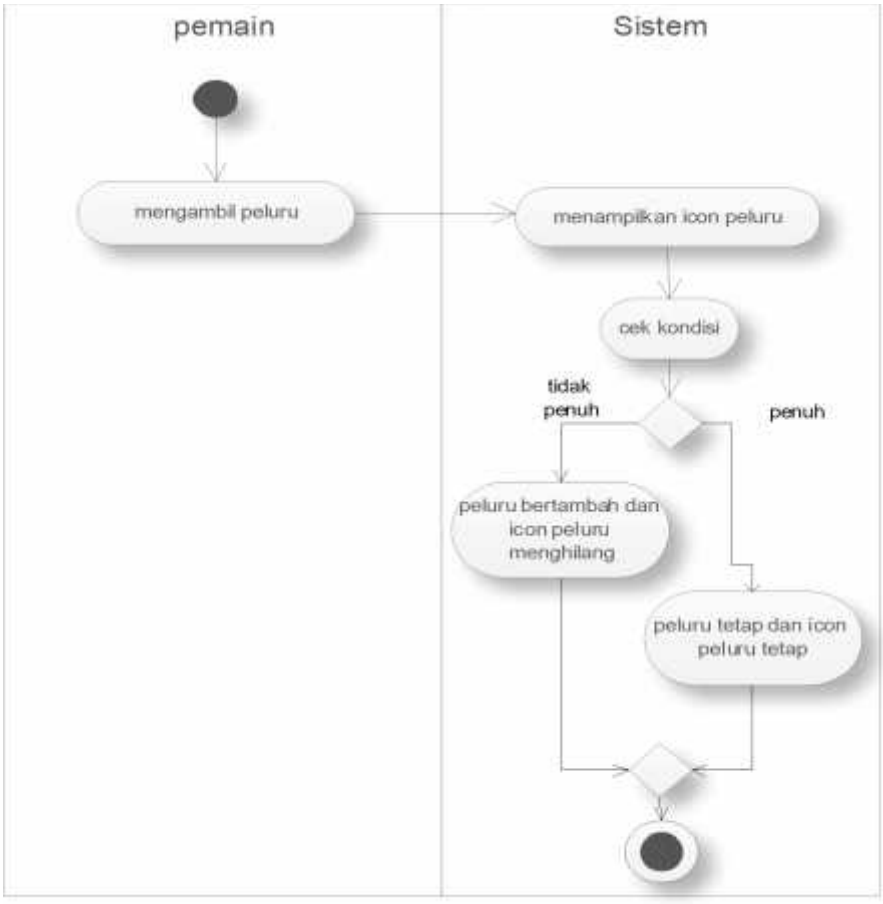
Gambar 3.26 *Activity Diagram* Mengganti Senjata

Tabel 3.23 Penjelasan Proses Mengganti Senjata

Deskripsi	Proses mengganti senjata dengan pilihan senjata laras panjang atau senjata laras pendek
Normal Flow	1. Pemain menggunakan senjata 2. Pemain menekan tombol 1 pada *keyboard*

	3. Sistem menukar senjata 4. Sistem menampilkan pemain menggunakan senjata laras pendek 5. Pemain menekan tombol 2 pada *keyboard* 6. Sistem mengganti senjata 7.Sistem menampilkan pemain menggunakan sejata laras panjang

J. *Activity Diagram* Mengambil Peluru

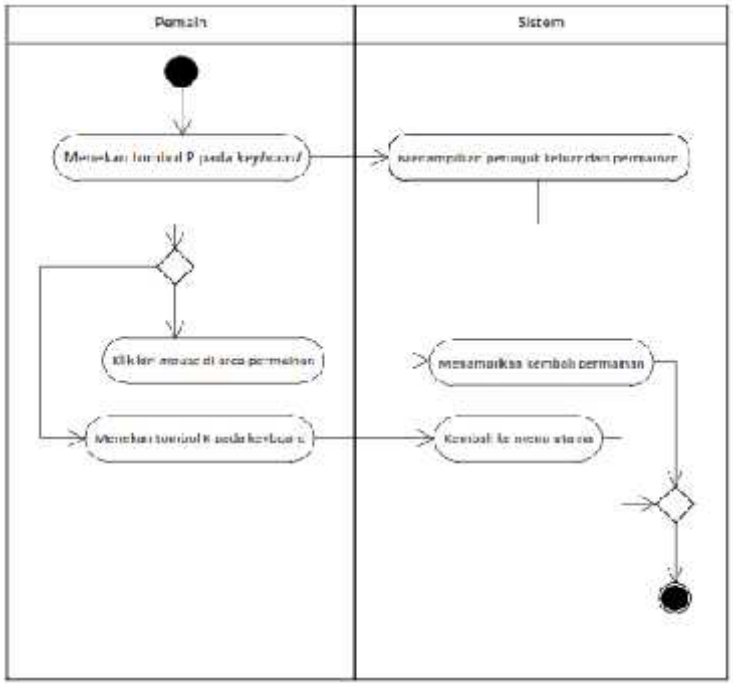


Gambar 3.27 *Activity Diagram* Mengambil Peluru

Tabel 3.24 Penjelasan Proses Mengambil Peluru

Deskripsi	Proses dimana pemain mengambil peluru
Normal *Flow*	1. Pemain mengambil peluru 2. Sistem menampilkan *icon* peluru 3. Sistem melakukan pengecekan 4. Apabila peluru pemain penuh, *bar* peluru tidak bertambah 5. Apabila peluru pemain penuh, *icon* peluru tidak menghilang 6. Apabila peluru pemain tidak penuh, sistem menampilkan bar peluru pemain yang bertambah 7. Apabila peluru pemain tidak penuh, sistem menghilangkan icon peluru yang telah diambil oleh pemain

K. *Activity Diagram* Memberhentikan Permainan



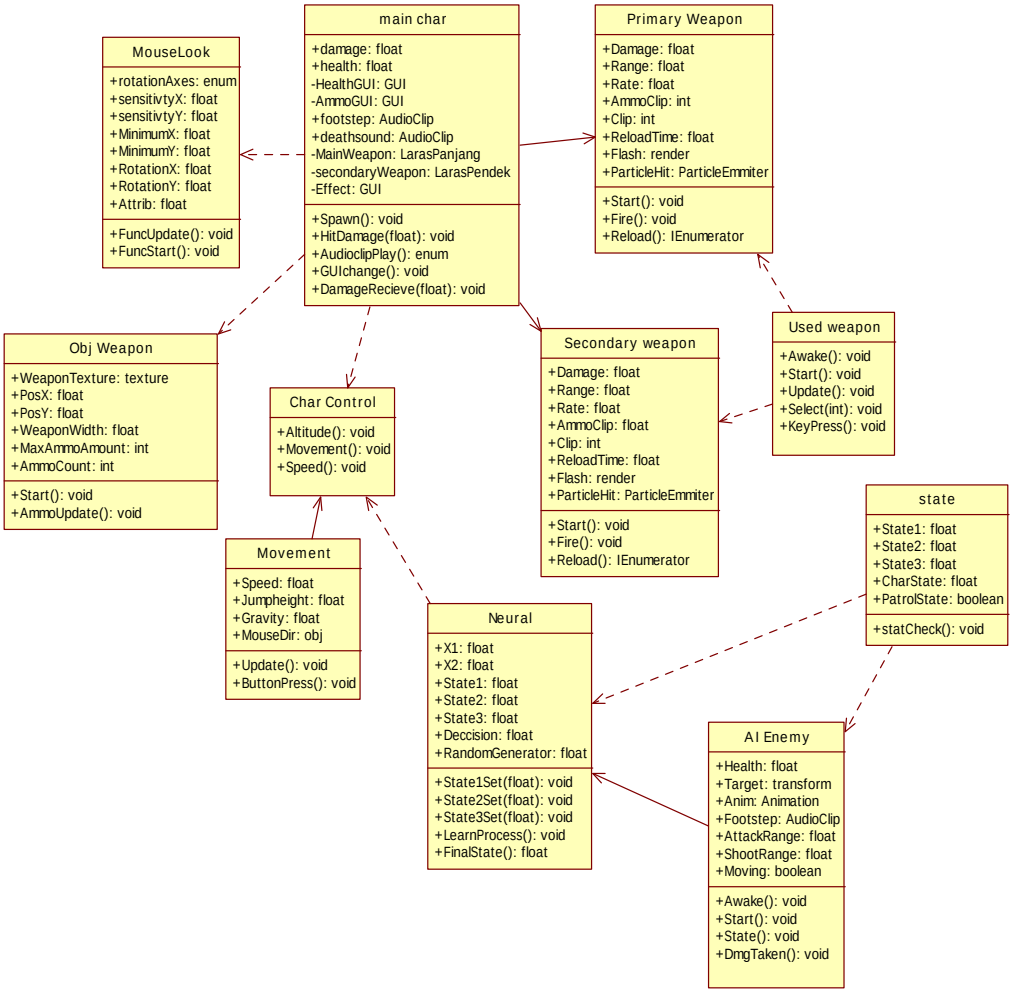
Gambar 3.28 *Activity Diagram* Memberhentikan Permainan

Tabel 3.25 Penjelasan Proses Memberhentikan Permainan

Deskripsi	Proses ketika pemain ingin melihat informasi kontrol atau keluar dari permainan
Normal Flow	1. Pemain menekan tombol P 2. Sistem menampilkan menu jeda berisi petunjuk untuk keluar dari permainan 3. Pemain menekan kiri *mouse* pada arena permainan 4. Sistem akan kembali menampilkan permainan 5. Pemain menenkan tombol p pada *keyboard* 6. Sistem akan keluar permainan dan kembali ke menu utama

3.12 Class diagram

Class Diagram digunakan untuk menggambarkan struktur sistem sebagai pendefinisian class-class yang dibuat untuk membangun sebuah sistem, berikut adalah Class Diagram pada Game yang dibangun:



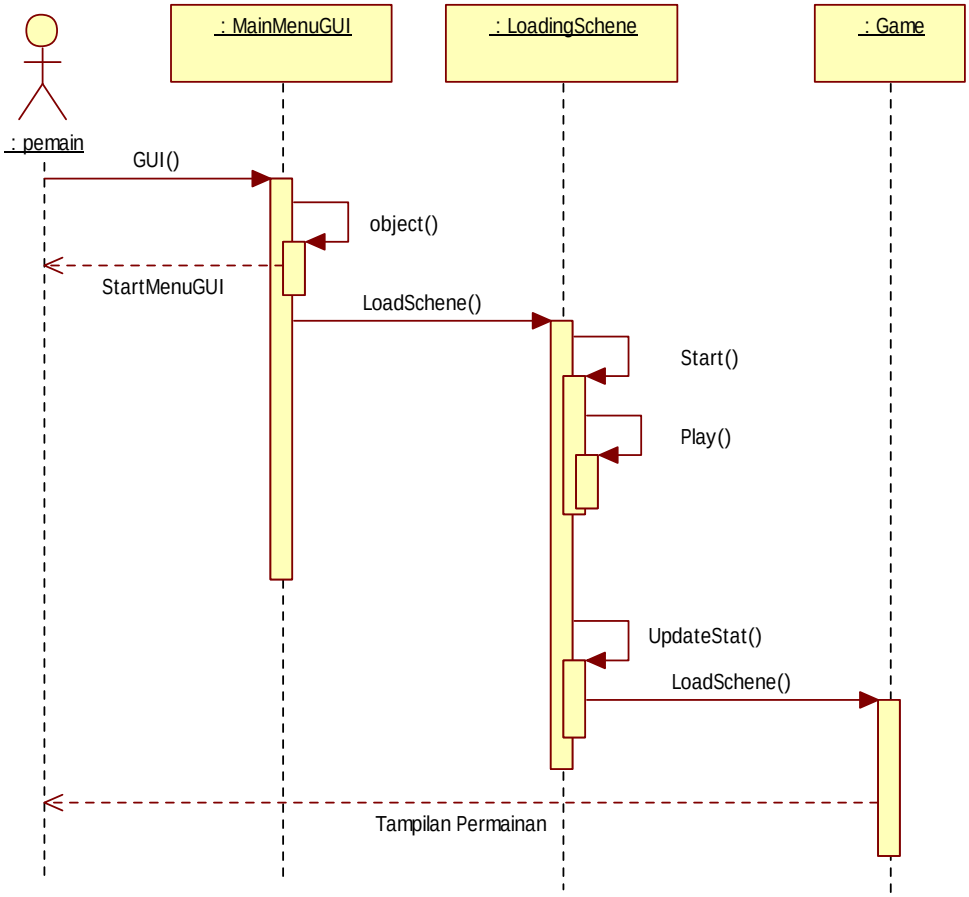
Gambar 3.29 *Class Diagram* Game

3.13 Sequence Diagram

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

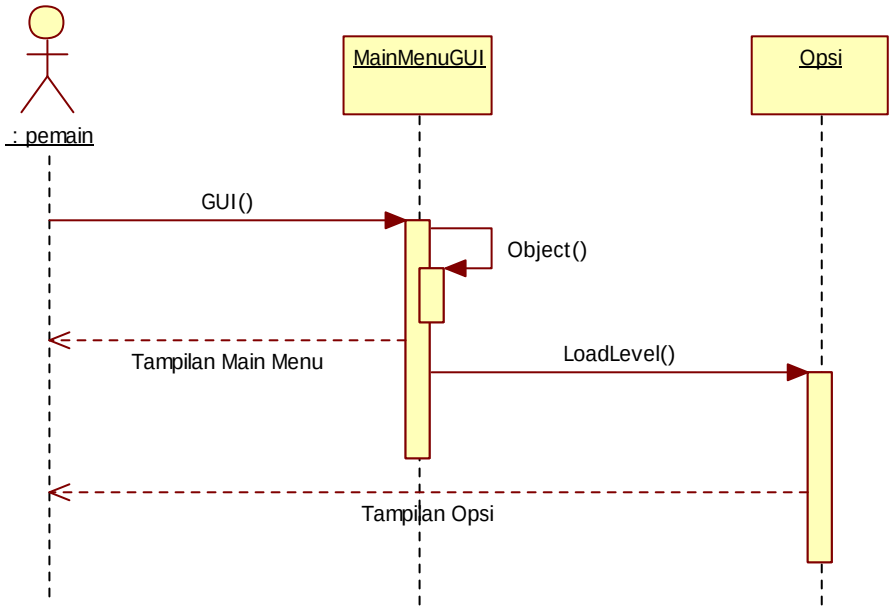
Sequence diagram biasa digunakan untuk menggambarakan skenario atau rangkaian langkah-langkah yang dilakukan sebagai respon dari sebuah event untuk menghasilkan output tertentu. Diawali dari apa yang men-*trigger* aktivitas tersebut, proses dan perubahan apa saja yang terjadi secara internal dan output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

A. *Sequence Diagram* Memulai Permainan



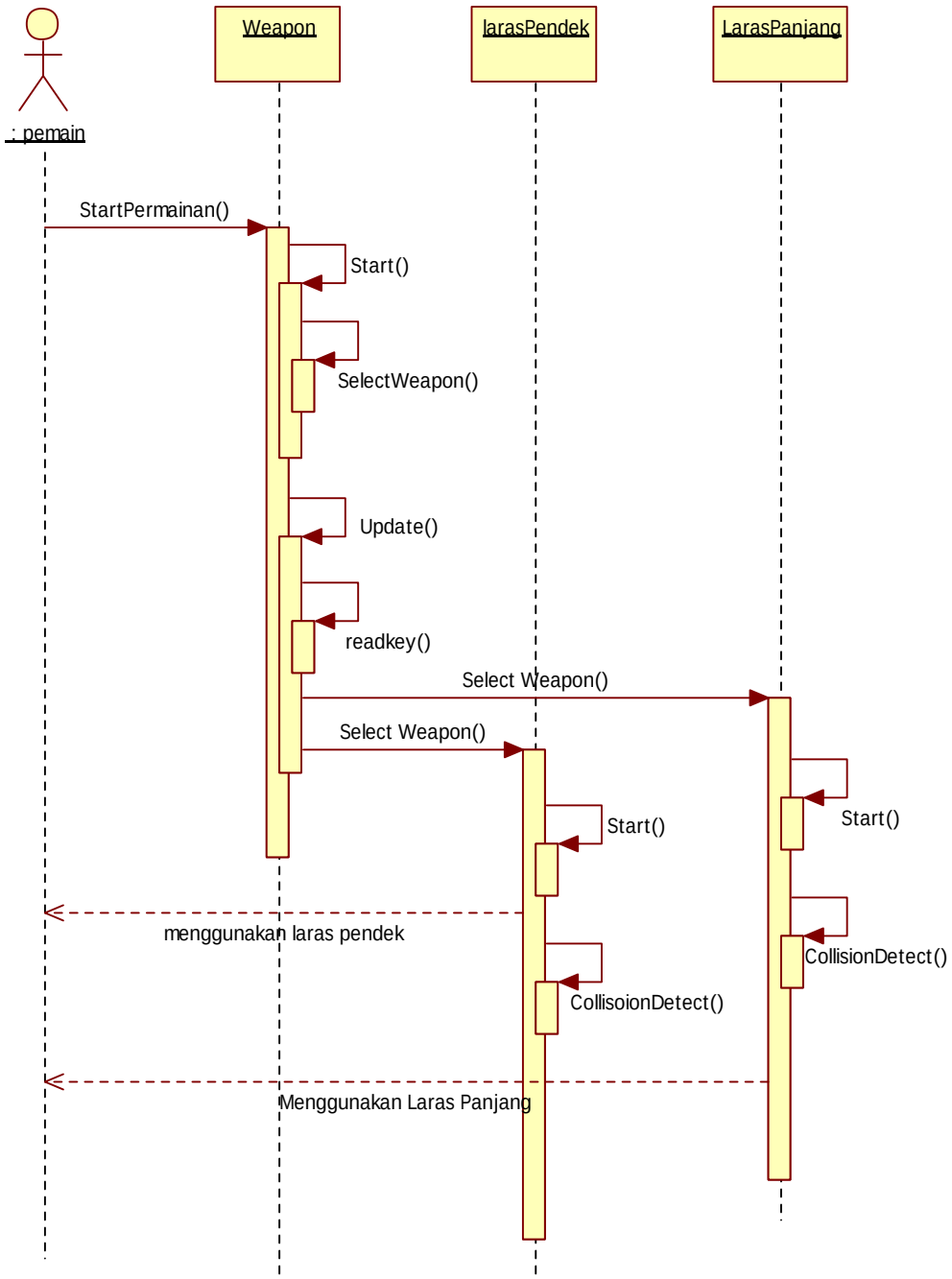
Gambar 3.30 *Sequence Diagram* Memulai Permainan

B. *Sequence Diagram* Melihat Opsi



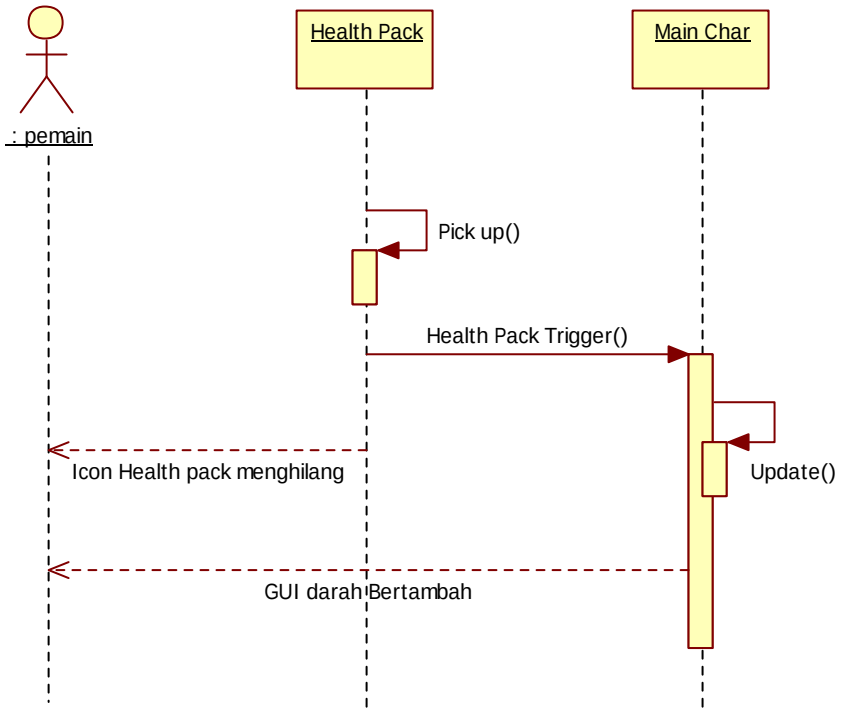
Gambar 3.31 *Sequence Diagram* Melihat Opsi

D. *Sequence Diagram* Mengganti Senjata



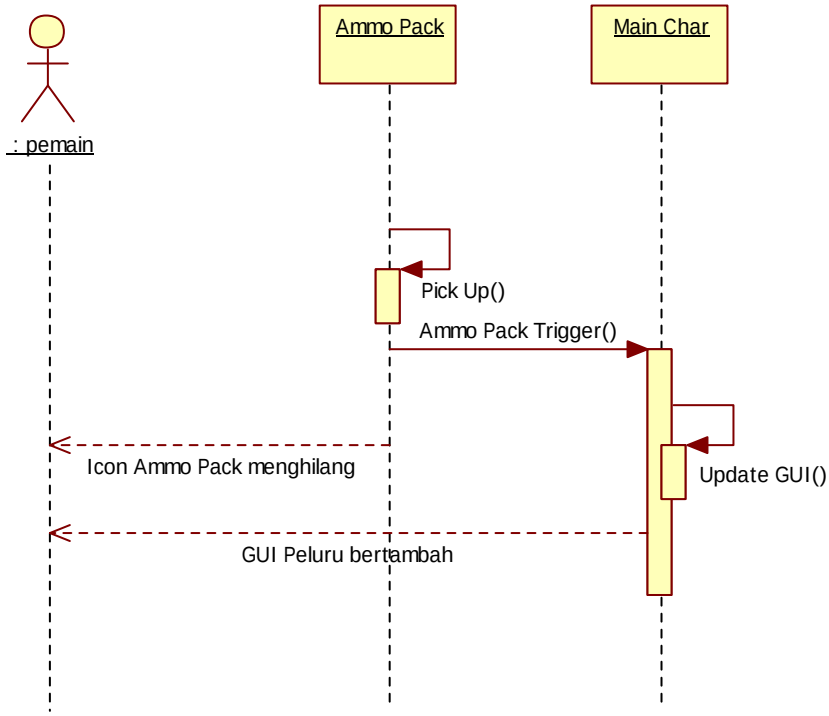
Gambar 3.32 *Sequence Diagram* Mengganti Senjata

E. *Sequence Diagram* Mengambil Darah



Gambar 3.33 *Sequence Diagram* Mengambil Darah

F. *Sequence Diagram* Mengambil Peluru



Gambar 3.34 *Sequence Diagram* Mengambil Peluru

3.14 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.

3.14.1 Perancangan Komponen Permainan

3.14.1.1 Karakter

Karakter yang dimainkan dalam *game* ini digambarkan sebagai seseorang yang terjebak di sebuah hotel dan harus meloloskan diri dari penjaga yang ada di sekitar hotel tersebut, dilengkapi dengan senjata laras panjang dan senjata laras pendek yang dimainkan dengan style FPS (*First person Shooter*).

a. Senjata

   Senjata yang digunakan terbagi kedalam 2 macam, yaitu laras pendek yang akan otomatis digunakan dan senjata laras panjang yang didapatkan di spot respawn supply



Gambar 3.35 Senjata Laras Panjang



Gambar 3.36 Senjata Laras Pendek

b. Musuh

Musuh dalam game ini berupa penjaga yang menggunakan senjata laras panjang sebagai senjatanya, dan juga memiliki kemampuan untuk menentukan gerak/ action yang akan dilakukan saat bertemu dengan pemain

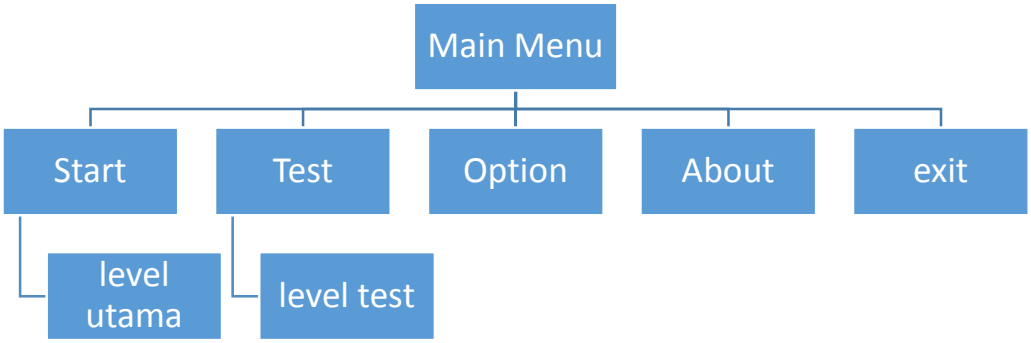


Gambar 3.37 Karakter AI penjaga

3.14.2 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.

3.14.2.1 Perancangan Struktur Menu



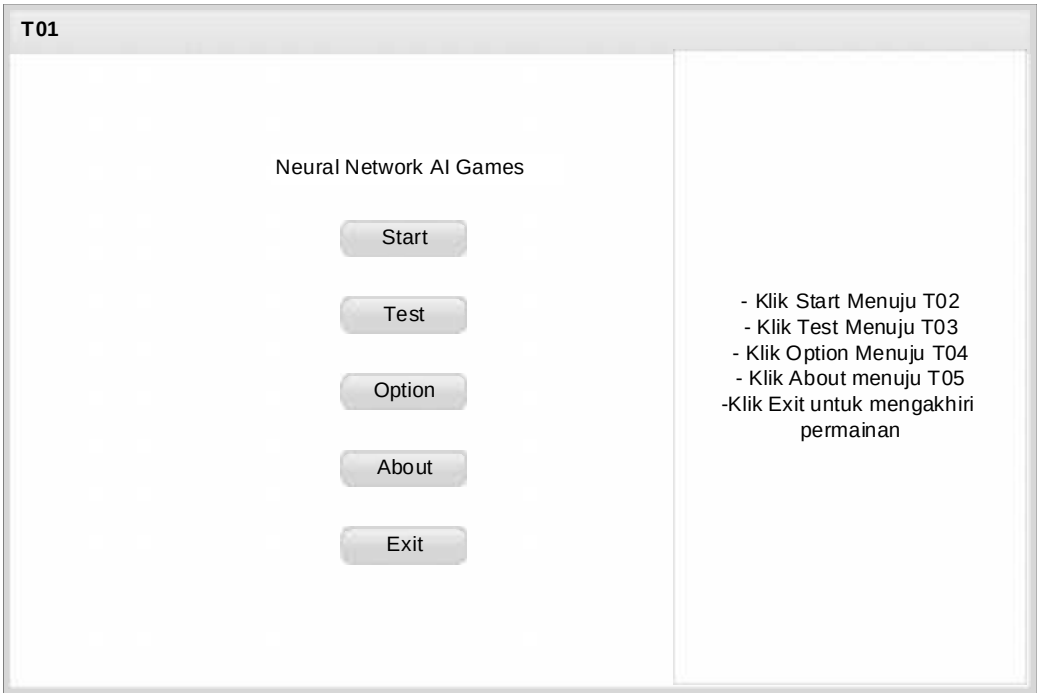
Gambar 3.38 Struktur menu utama

3.14.3 Perancangan Antarmuka

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

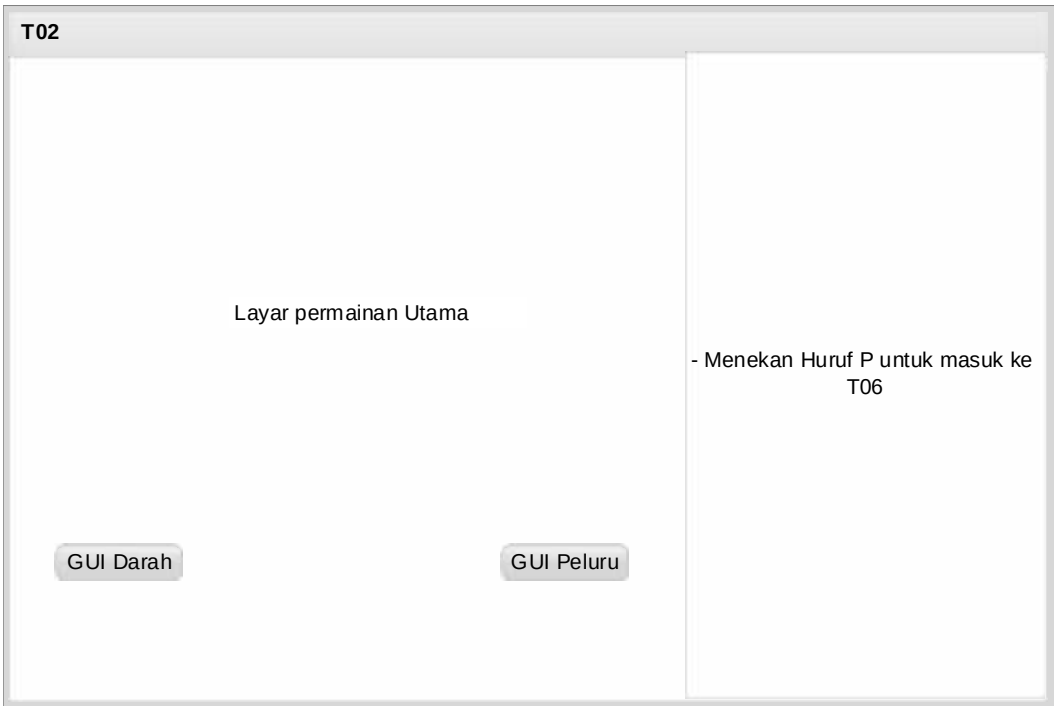
1. Menu Utama

   Berikut dapat dilihat perancangan antarmuka menu utama dari *game*



Gambar 3.39 Tampilan Menu Utama

2. Permainan



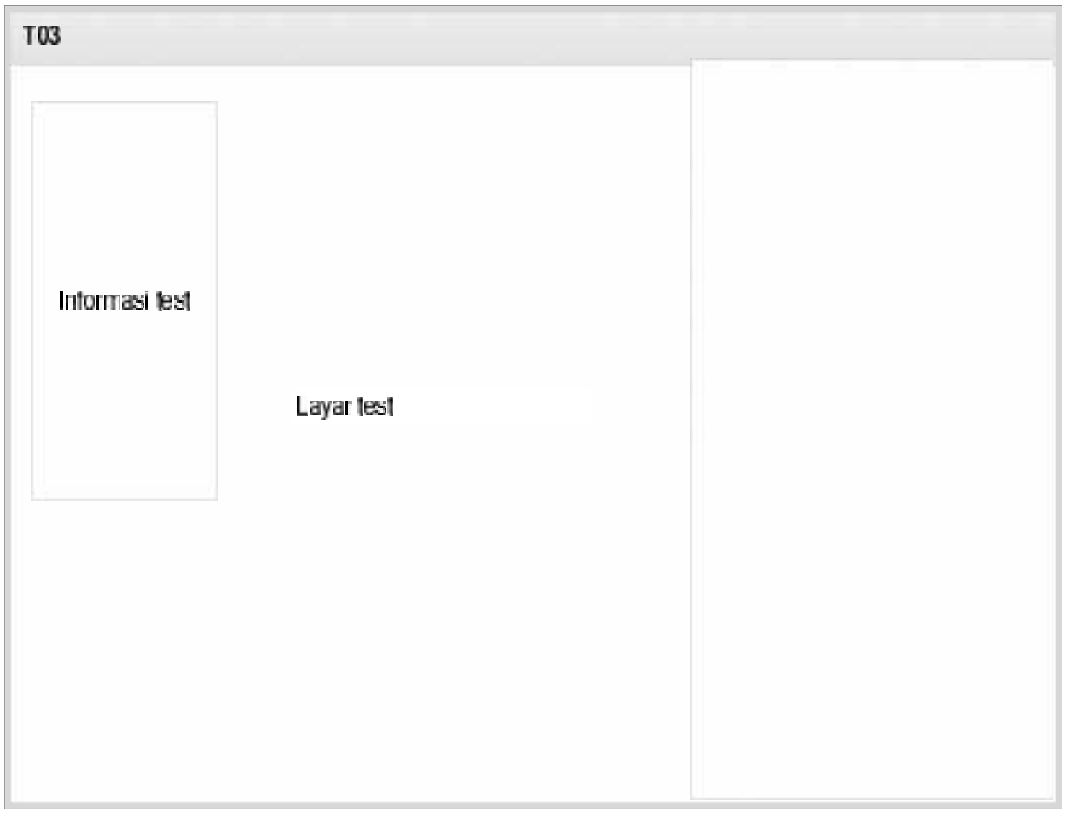
Gambar 3.40 Tampilan Permainan

3. Pause menu



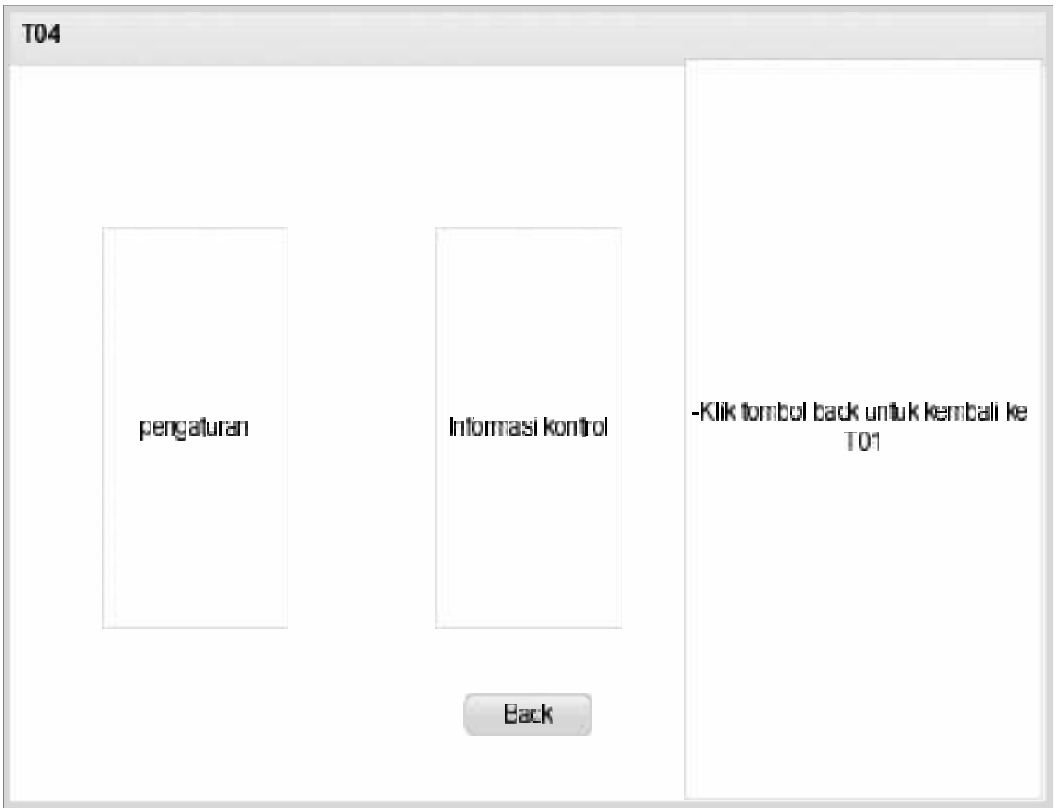
Gambar 3.41 Tampilan Pause Menu

4. Layar test



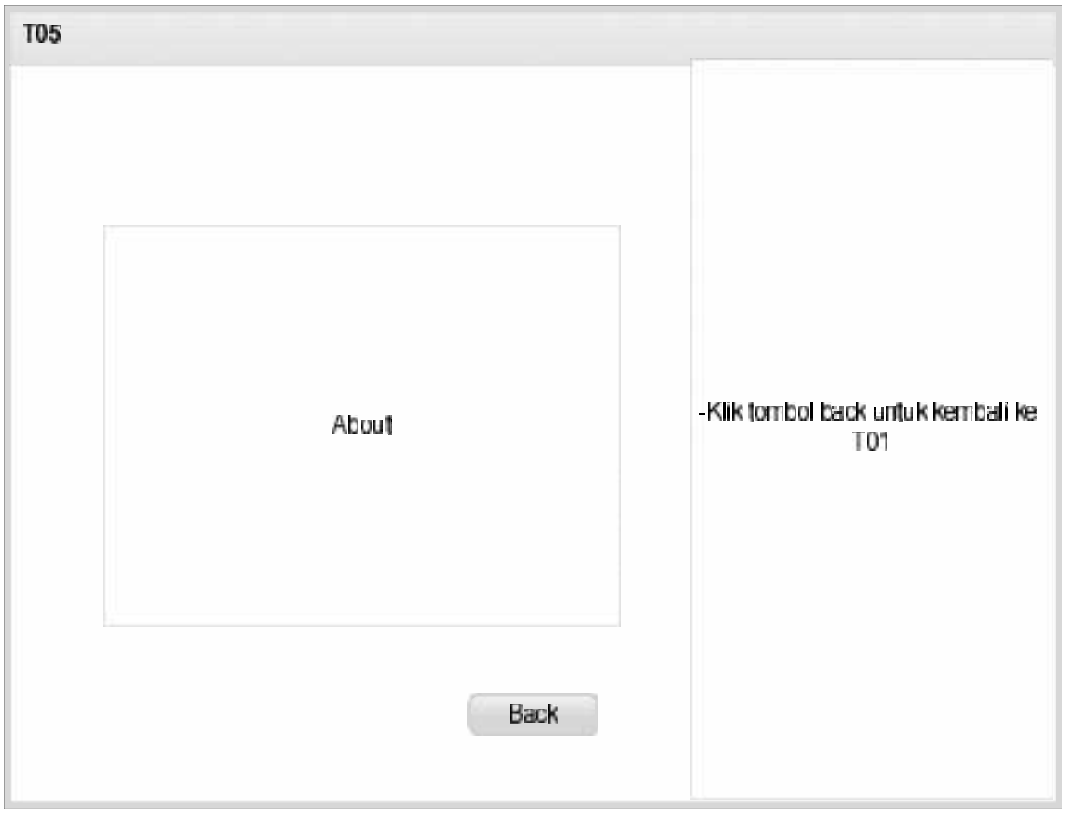
Gambar 3.42 Tampilan Layar test

5. Opsi



Gambar 3.43 Tampilan Opsi

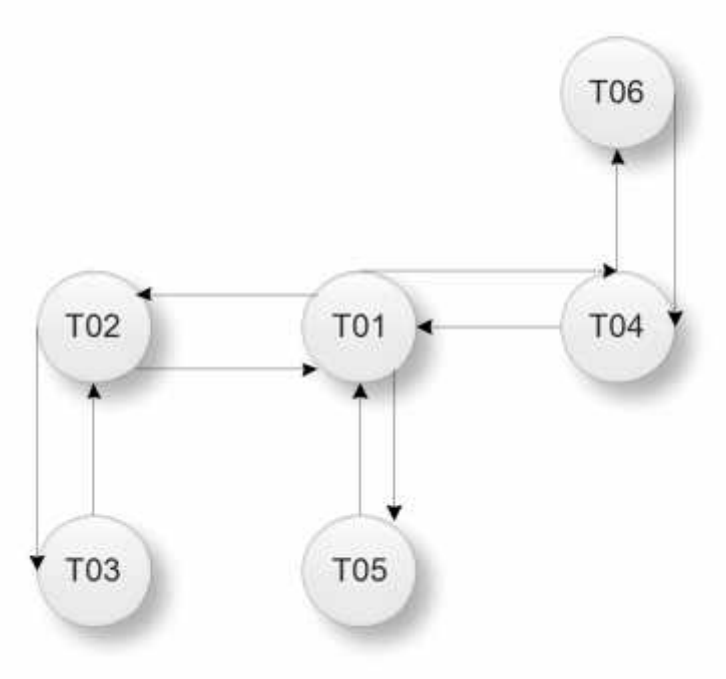
6. About



Gambar 3.44 Tampilan About

3.14.3.1 Jaringan Semantik

Secara struktur game ini tersusun secara modular, dalam kata lain tersusun atas modul modul yang dapat dipisahkan untuk tujuan testing maupun untuk di modifikasi, berikut adalah jaringan semantik dari game ini:



Gambar 3.45 Jaringan Semantik