BAB 3
ANALISIS DAN PERANCANGAN SISTEM

3.1 Analisis Sistem

Pada bagian analisis sistem ini, akan dilakukan berbagai macam analisis yang berhubungan dengan *game* yang akan dibangun, mulai dari analisis terhadap game sejenis analisis berbagai kebutuhan baik fungsional maupun non-fungsional.

3.1.1 Analisis Masalah

Dalam hal ini diperlukan sebuah analisis yang dilakukan terhadap aplikasi permainan lain yang sejenis untuk menentukan dimana letak kekurangan aplikasi permainan tersebut, sehingga dapat dikembangkan dengan membuat aplikasi permainan baru agar semakin menarik dibandingkan dengan aplikasi permainan yang dianalisis. Aplikasi permainan sejenis yang akan dianalisis adalah:

A. Aplikasi permainan yang berjudul *Garden Maze*.



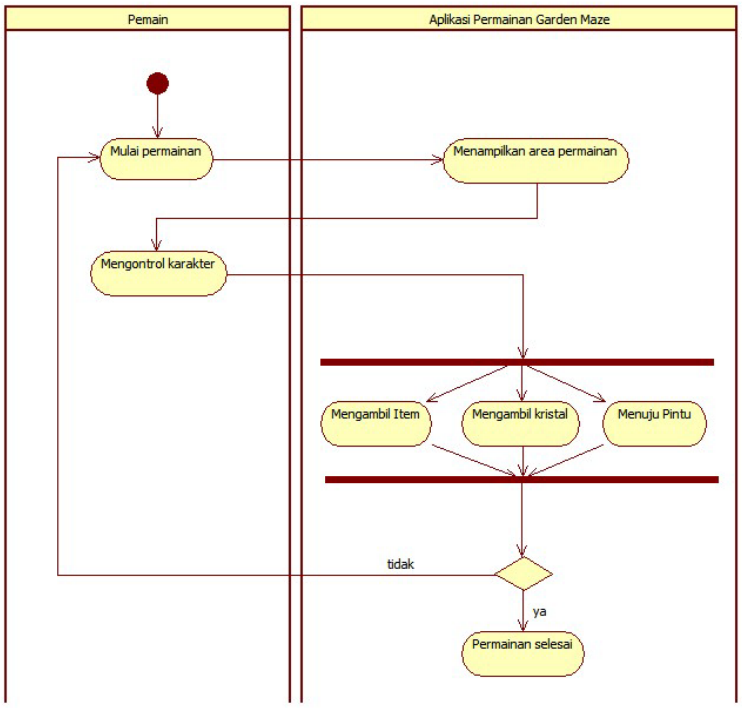
Gambar 3.1 *Screenshoot garden maze* [14]

a. *Gameplay*

Garden Maze menawarkan sebuah permainan *game* ber-*genre maze*. Pada *game* tersebut, pemain diharuskan mengambil sebuah kristal lalu kembali ke rumah untuk bisa melanjutkan ke

Level selanjutnya. Adapun musuh yang hanya terdiri dari 2 atau lebih tetapi hanya mempunyai 1 jenis musuh saja dengan kekuatan yang sama. Musuh bergerak dengan 2 cara yaitu secara horisontal dan vertikal, ketika pemain menyentuh musuh maka pemain akan mengulangi permainan dari *Level* yang sama.

Pada game ini disediakan 4 *Level* yang mempunyai tingkat kesulitan yang semakin meningkat, dimana tingkat kerumitan berasal dari tembok penghalang yang semakin banyak tetapi tidak ada peningkatan kepintaran dalam NPCs.



Gambar 3.2 *Activity Diagram* misi permainan *Garden Maze*

b. *Scoring*

Garden Maze tidak memiliki salah satu *point* penting dalam aplikasi permainan yaitu *score* sehingga pemain tidak bisa mengetahui *scoring* dalam permainannya sendiri.

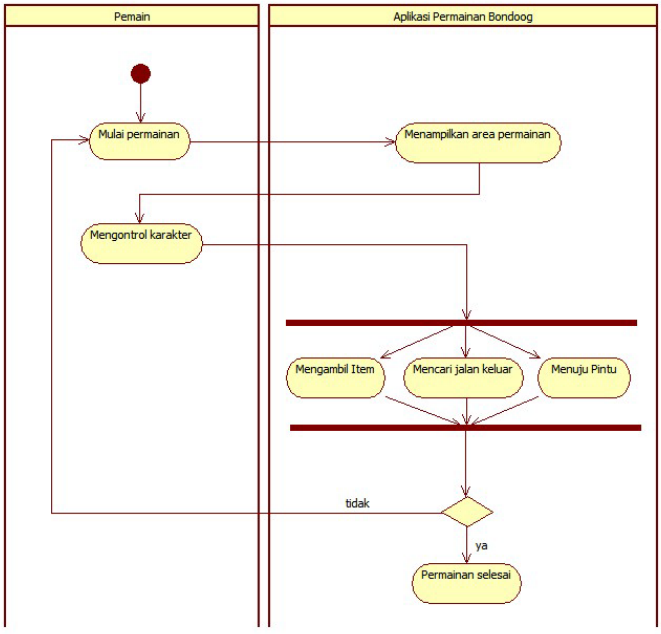
B. Aplikasi permainan yang berjudul *Boondog*.



Gambar 3.3 *Screenshoot Boondog* [15]

a. *Gameplay*

Boondog merupakan permainan seperti labirin, mencari jalan keluar untuk menuju ke pintu dan melanjutkan ke *level* selanjutnya.Pada *game* ini tidak terdapat musuh, hanya mencari jalan keluar dengan melewati rintangan, seperti lubang dan meloncat antara *tile* yang tesedia, sehingga alur permainan ini menjadi monoton dan kurang tertantang.



Gambar 3.4 *Activity diagram* misi permainan *Bondoog*

b. *Scoring*

Boondog mempunyai *Scoring* dalam permainannya, tujuan dari *scoring* permainan Boondog adalah menampilkan *high score* pemain.

Berdasarkan analisis yang dilakukan terhadap *game* sejenis di atas, dapat disimpulkan bahwa dalam *game* tersebut, dalam hal cara bermain masih terlalu monoton, hal ini dibuktikan pada alur permainannya yang masih sederhana, mulai dari tingkatan kesulitan yang hanya dibedakan dalam bentuk penghalang saja.

3.1.2 Analisis Pembangunan Aplikasi Permainan *The Tale of Hunter*

The Tale of Hunter merupakan nama aplikasi permainan ber-*genre maze* yang akan dibuat.Kata *Tale* dalam aplikasi ini berarti kisah, dan kata *Hunter* yang berarti pemburu sehingga *The Tale of Hunter* adalah kisah seorang pemburu. Dalam game ini mengisahkan seorang pemburu yang akan menyelamatkan seorang putri yang ditangkap olah sekelompok bandit. Adapun tujuan dari aplikasi permainan ini yang bukan hanya untuk memberikan hiburan semata saja tetapi meningkatkan ketangkasan pemain. Aplikasi permainan ini dibangun dalam bentuk dua dimensi (2D), adapun perbaikan yang terdapat didalam aplikasi permainan *The Tale of Hunter* yang terdapat pada tabel 3.1.

Tabel 3.1 Perbaikan Aplikasi Permainan

Kelebihan Aplikasi Permainan *The Tale of Hunter*
1. Terdapat musuh yang semakin kuat 2. Penerapan konsep AI pada musuh 3. *Map / area* permainan yang semakin luas. 4. Menggabungkan 2 jenis permainan yang berbeda

3.1.2.1 *Story Line*

Game The Tale of Hunter bercerita tentang sebuah Kerajaan yang di pimpin oleh raja dan ratu. Mereka mempunyai seorang anak perempuan. Suatu hari putri tersebut melakukan perjalanan pulang ke kerajaan, namun dalam perjalanan pulang, para rombongon putri diserang oleh bandit-bandit lalu sang putri pun ditawan oleh bandit-bandit tersebut. Raja dan ratu lalu melakukan sebuah sayembara, barang siapa yang bisa

menyelamatkan putri akan di berikan sebuah hadiah. Ada seseorang pemburu yang bernama Hunter yang mengikuti sayembara tersebut.

Hunterpun melakukan misi penyelamat sang putri ke markas bandit. Untuk menyelamatkan putri, Hunter harus melewati empat *Level*, untuk melewati tiap *level*-nya Hunter harus harus menemukan kunci untuk membuka gerbang ke *Level* selanjutnya, namun harus dalam menemukan kunci tersebut, Hunter harus menghadapi bandit.

Setelah putri diselamatkan, para bandit tidak tinggal diam saja, mereka balik dan menyerang istana karena mereka ingin membalaskan dendam mereka terhadap kerajaan yang mengirimkan Hunter yang mengalahkan banyak bandit.

Alur dan misi dari aplikasi permainan *The Tale of Hunter*, yaitu :

A. Alur Aplikasi Permainan *The Tale of Hunter*

Alur permainan merupakan ketentuan-ketentuan yang terdapat pada sebuah aplikasi permainan agar setiap pemain dapat mengetahui bagaimana cara untuk memainkan permainan ini dan untuk menyelesaikan misi pada aplikasi permainan tersebut.

Alur permainan dari aplikasi permainan *The Tale of Hunter* terdapat pada tabel 3.2.

Tabel 3.2 Alur Aplikasi Permainan

Alur Permainan
1. Pemain bergerak/berjalan mencari kunci untuk melanjutkan *Level*. 2. Pada *Level* ke-4 musuh bergerak dengan cara menentukan rute terpendek. 3. Pemain bisa menembak musuh. 4. Jika pemain menyentuh musuh, maka pemain harus mengulangi di*Level* pertama. 5. Pada level ke- 4, pemain akan membangun tower untuk menghadang musuh

B. Misi Aplikasi Permainan *The Tale of Hunter*

Misi yang harus diselesaikan pada aplikasi permainan *The Tale of Hunter* ini yaitu dengan mencari kunci dan mengalahkan semua bandit untuk melanjutkan ke *level* selanjutnya. Setiap *level* meningkat maka bandit akan bertambah banyak dan semakin kuat. Pada *level* tarakhir pemain harus membangun *tower* untuk mengalahkan musuh yang datang untuk menyerang kerajaan.

3.1.2.2 Analisis Tingkat Kesulitan

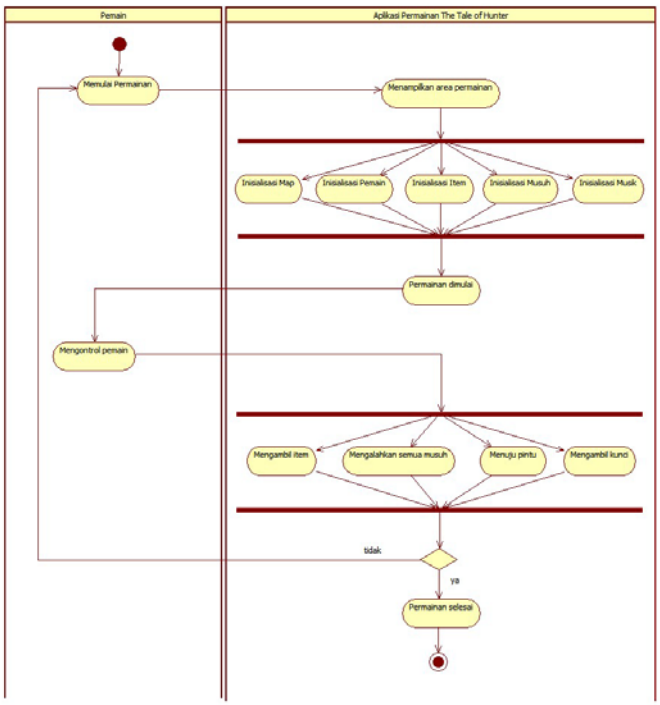
Pada setiap tingkatan level, musuh sudah di terapkan kecerdasan oleh penulis. Pemain akan muncul dari tempat yang sudah ditentukan lalu pemain harus mengambil kunci dan mengalahkan semua musuh yang keluar dari sebuah pintu, ketika kunci sudah diambil dan semua musuh sudah kalah, baru pemain bisa masuk ke sebuah pintu untuk melanjutkan *level*. Tingkatan kesulitan antara *level* berada pada jumlah musuh yang keluar, dan *Lifepoint* musuh yang semakin kuat.

3.1.2.3 *Gameplay*

Pada game ini, untuk menyelamatkan putri, Hunter harus melewati empat *Level*, untuk melewati tiap *level Hunter* harus menemukan kunci untuk membuka gerbang ke *level* selanjutnya, namun harus dalam menemukan kunci tersebut, *Hunter* harus menghadapi bandit-bandit disetiap *level*-nya, setelah kunci di temukan dan semua musuh dikalah baru bisa melanjutkan ke level selanjutnya.

Ketika bandit menyerang ke kerajaan, maka *gameplay* akan berbeda, pemain akan membangun *tower-tower* untuk menghadang bandit yang ingin memasuki daerah istana.

Berikut merupakan gambar *activity diagram gameplay* aplikasi permainan *The Tale of Hunter*.



Gambar 3.5 ***Activity Diagram Gameplay The Tale of Hunter***

3.1.2.4 *Scoring*

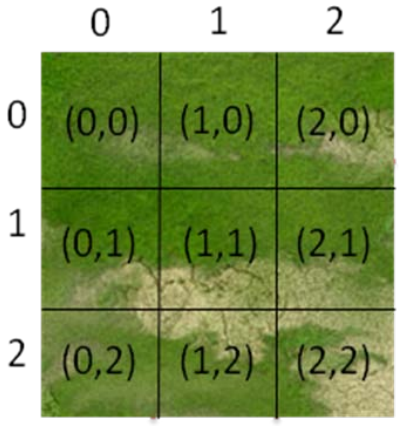
Game The Tale of Hunter mempunyai perhitungan pada setiap kali musuh dikalahkan. Setiap musuh yang kalah akan diberikan *score* 50 *point*, dan berlaku untuk semua musuh. Tujuan di berikan *scoring* juga untuk menambahkan *lifepoint* pemain, ketika *score* menjadi 500, maka *lifetpoint* pemain akan bertambah dan berlaku kelipatan.

3.1.3 Analisis Algoritma

Algoritma A* (A Star) merupakan perbaikan dari metode BFS dengan memodifikasi fungsi heuristiknya. Algoritma ini melakukan pencarian jalur terpendek dengan mencari biaya dari penjumlahan biaya sebenarnya ditambah dengan biaya perkiraan. Dalam notasi matematika dapat dituliskan : $f(n) = g(n) + h(n)$. Pada kondisi yang tepat, A* akan memberikan solusi yang terbaik dalam waktu yang optimal.

1. **Analisis Penggunaan Algoritma A* (A Star) pada *Game The Tale of Hunter***

   Pada aplikasi permainan *The Tale of Hunter* arena permainan berupa kumpulan *tile-tile* yang terdiri dari orde X x Y. Tiap *tile* memiliki koordinat X dan Y atau berbentuk seperti array dua dimensi.



Gambar 3.6 Arena permainan pada *Game The Tale of Hunter*

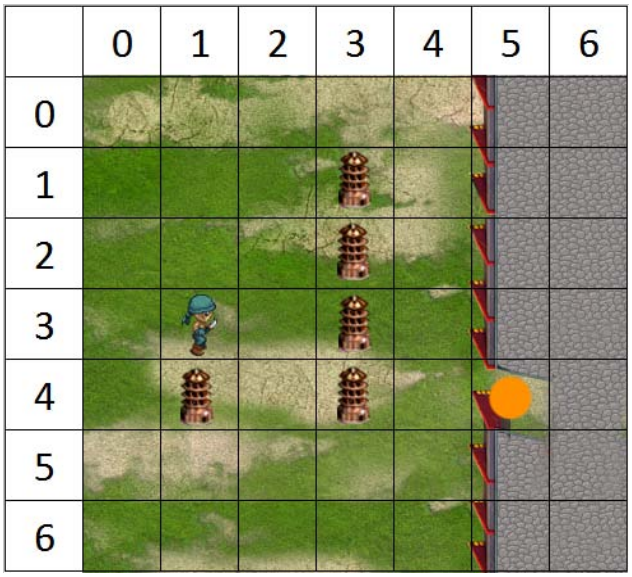
Algortima A* akan diterapkan pada objek musuh dalam melakukan pencarian karakter pemain di arena permainan. Namun terdapat beberapa hal yang perlu didefinisikan terlebih dahulu. Adapun istilah-istilah yang akan dibahas yaitu *open list, closed list*, nilai f, g dan n.

1. *OPEN LIST* adalah *list* yang menyimpan kemungkinan *path* yang akan diperiksa. *OPEN LIST* dibuat terurut berdasarkan nilai f. *OPEN LIST* digunakan untuk menentukan secara selektif (berdasarkan nilai f) jalan yang dikira lebih dekat menuju pada path tujuan. *OPEN* berisi simpul-simpul yang masih memiliki peluang untuk terpilih sebagai simpul terbaik (*best node*).
2. CLOSED adalah senarai (*list*) untuk menyimpan simpul-simpul yang sudah pernah dibangkitkan dan sudah pernah terpilih sebagai simpul terbaik (*best node*) atau senarai yang menyimpan jalan yang sudah diperiksa dari *open list*. Artinya, *CLOSED* berisi simpul-simpul yang tidak mungkin terpilih sebagai simpul terbaik (peluang untuk terpilih sudah tertutup). Kedua list (*OPEN LIST* dan *CLOSED LIST*) ini bertujuan juga untuk menghindari penelusuran berkali-kali jalan (rute) yang memang sudah diidentifikasi agar tidak masuk kembali ke dalam *OPEN LIST*.
3. Nilai F adalah *cost* perkiraan suatu *path* yang teridentifikasi. Nilai F merupakan hasil dari f(n).
4. Nilai G hasil dari fungsi g(n), adalah banyaknya langkah yang diperlukan untuk menuju ke *path* sekarang.
5. Setiap simpul (*node*) harus memiliki informasi nilai h(n), yaitu estimasi harga simpul tersebut dihitung dari simpul tujuan yang hasilnya menjadi nilai H.

Fungsi f sebagai estimasi fungsi evaluasi terhadap *node* n, dapat dituliskan :

2. Ilustrasi Perhitungan Algoritma A*

Arena permainan pada *game The Tale of Hunter* terdiri dari orde X x Y . dalam analisis ini akan dijelaskan dengan ordo minimal yaitu 7x7 dikarenakan perhitungan pada ordo berapa pun akan sama. Pada kondisi ini simpul awal atau posisi musuh dicontohkan berada pada koordinat (1,3), sedangkan simpul tujuan atau posisi karakter pemain berada pada koordinat (5,4) di arena permainan.



Gambar 3.7 Contoh kondisi awal

Langkah selanjutnya akan dilakukan pemeriksaan terhadap simpul-simpul yang bertetangga yang memiliki nilai *cost* / biaya terkecil sebagai *Best Node* sebagai rute yang akan dipilih musuh, dengan melakukan langkah perhitungan sebagai berikut:

Langkah Pertama

a. Koordinat n(2,3)

Nilai nx=2 Nilai ny=3

Simpul Tujuan (5,4) sehingga goal.x=5, goal.y=4

g(2,3)=1

h(2,3) = (abs(nx-goal.x)+abs(ny-goal.y))

h(2,3)=(abs(2-5)+abs(3-4))

h(2,3)=abs(-3)+abs(-1)

h(2,3)=3+1

h(2,3)=4

f(2,3)=g(2,3)+h(2,3)

f(2,3)=1+4

f(2,3)=5

b. Koordinat (1,2)

Nilai nx=1 Nilai ny=2

Simpul Tujuan (5,4) sehingga goal.x=5, goal.y=4

g(1,2) = 1

h(1,2) = (abs(nx-goal.x)+abs(ny-goal.y))

h(1,2) = (abs(1-5)+abs(2-4))

h(1,2) = abs(-4)+abs(-2)

h(1,2) = 4+2

h(1,2) = 6

f(1,2) = g(1,2)+h(1,2)

f(1,2) = 1+6

f(1,2) = 7

c. Koordinat(0,3)

Nilai nx=0 Nilai ny=3

Simpul Tujuan (5,4) sehingga goal.x=5, goal.y=4

g(0,3)=1

h(0,3) = (abs(nx-goal.x)+abs(ny-goal.y))

h(0,3)=(abs(0-5)+abs(3-4))

h(0,3)=abs(-5)+abs(-1)

h(0,3)=5+1

h(0,3)=6

f(0,3)=g(0,3)+h(0,3)

f(0,3)=1+6

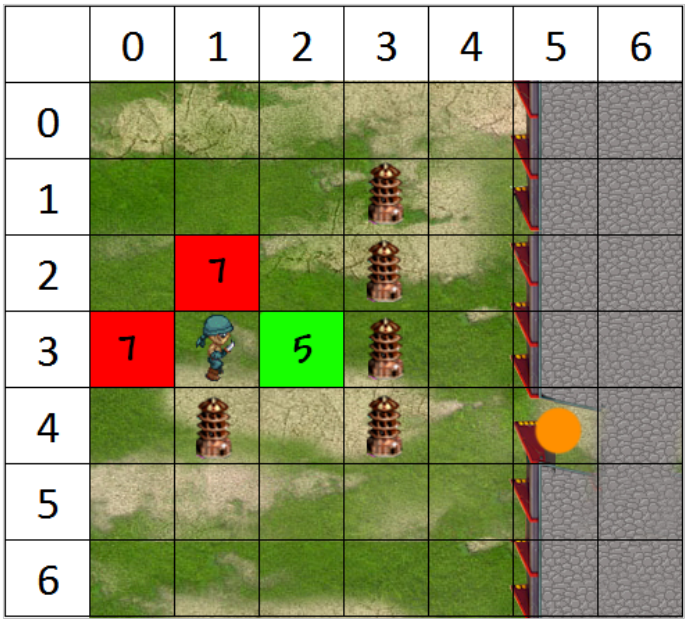
f(0,3)=

Keterangan:

f(n) : berada di dalam simpul di tepi kiri atas

g(n) : berada didalam simpul di tepi kiri bawah

h(n) : berada di dalam simpul di tepi kanan bawah



Gambar 3.8 Langkah pertama pencarian best node

Dari hasil perhitungan diatas seperti terlihat pada gambar 3. Tiga simpul yang mungkin menjadi bestnode. Dari ketiga simpul tersebut yang mungkin menjadi bestnode maka dipilihlah simpul (2,3) sebagai bestnode karena memiliki biaya terkecil yaitu 5.

1. Langkah kedua

a. Koordinat (2,2)

Nilai nx=2 Nilai ny=2

Simpul Tujuan (5,4) sehingga goal.x=5, goal.y=4

g(2,2)=2

h(2,2) = (abs(nx-goal.x)+abs(ny-goal.y))

h(2,2)=(abs(2-5)+abs(2-4))

h(2,2)=abs(-3)+abs(-2)

h(2,2)=3+2

h(2,2)=5

f(2,2)=g(2,2)+h(2,2)

f(2,2)=1+6

f(2,2)=7

b. Koordinat (2,4)

Nilai nx=2 Nilai ny=4

Simpul Tujuan (5,4) sehingga goal.x=5, goal.y=4

g(2,4)=2

h(2,4) = (abs(nx-goal.x)+abs(ny-goal.y))

h(2,4)=(abs(2-5)+abs(4-4))

h(2,4)=abs(-3)+abs(0)

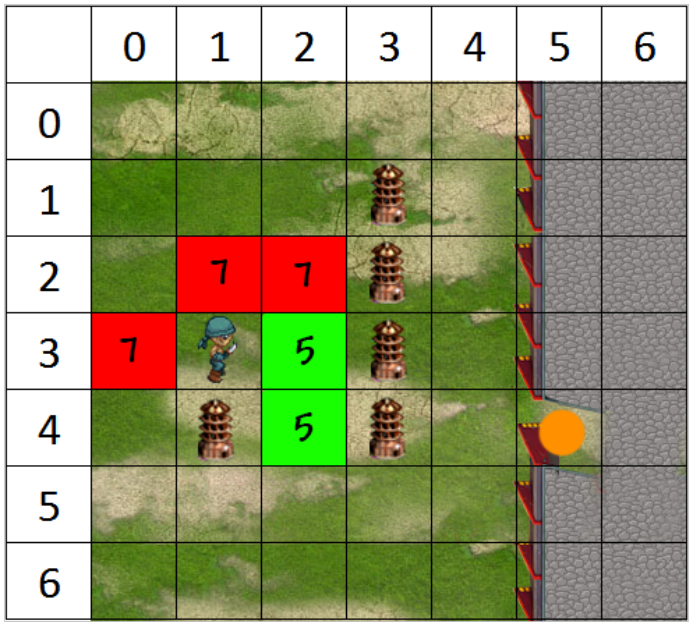
h(2,4)=3+0

h(2,4)=3

f(2,4)=g(2,4)+h(2,4)

f(2,4)=2+3

f(2,4)=5



Gambar 3.9 Langkah kedua pencarian best node

Setelah mendapatkan bestNode dilangkah pertama, dilakukan penelusuran selanjutnya untuk mendapatkan bestnode selanjutnya dengan melakukan penelusuran simpul-simpul yang memiliki nilai cost terkecil. Simpul yang dihitung selanjutnya adalah koordinat (2,2) dan (2,4), koordinat (3,3) tidak dihitung atau diabaikan karena terdapat penghalang yang tidak bisa dilewati musuh. Hasilnya adalah koordinat (2,4) mempunyai cost terkecil yaitu 5, maka koordinat (2,4) dipilih sebagai bestnode selanjutnya.

2. Langkah ketiga

a.Koordinat (2,5)

Nilai nx=2 Nilai ny=5

Simpul Tujuan (5,4) sehingga goal.x=5, goal.y=4

g(2,5)=3

h(2,5) = (abs(nx-goal.x)+abs(ny-goal.y))

h(2,5)=(abs(2-5)+abs(5-4))

h(2,5)=abs(-3)+abs(1)

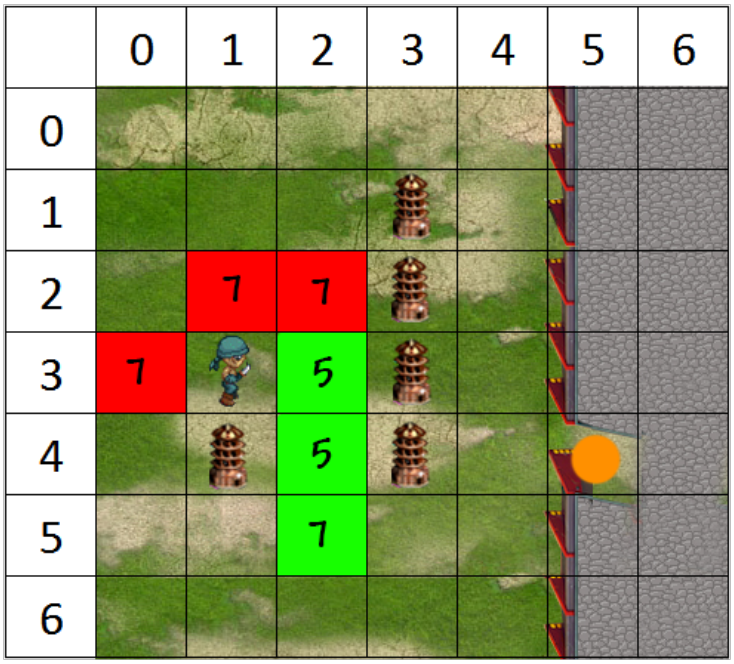
h(2,5)=3+1

h(2,5)=4

f(2,5)=g(2,5)+h(2,5)

f(2,5)=3+4

f(2,5)=7



Gambar 3.10 Langkah ketiga pencarian best node

Setelah mendapatkan bestNode dilangkah kedua, dilakukan penelusuran selanjutnya untuk mendapatkan bestnode selanjutnya dengan melakukan penelusuran simpul-simpul yang memiliki nilai cost terkecil. Simpul yang dihitung selanjutnya adalah koordinat (2,5), koordinat (1,4) dan (3,4) tidak dihitung atau diabaikan karena terdapat penghalang yang tidak bisa dilewati musuh. Hasilnya adalah koordinat (2,5) mempunyai nilai cost 7, maka koordinat (2,5) dipilih sebagai bestnode selanjutnya.

3. Langkah keempat.

a.Koordinat (3,5)

Nilai nx=3 Nilai ny=5

Simpul Tujuan (5,4) sehingga goal.x=5, goal.y=4

g(3,5)=4

h(3,5) = (abs(nx-goal.x)+abs(ny-goal.y))

h(3,5)=(abs(3-5)+abs(5-4))

h(3,5)=abs(-2)+abs(1)

h(3,5)=2+1

h(3,5)=3

f(3,5)=g(2,5)+h(2,5)

f(3,5)=4+3

f(3,5)=7

b. Koordinat (1,5)

Nilai nx=1 Nilai ny=5

Simpul Tujuan (5,4) sehingga goal.x=5, goal.y=4

g(1,5)=4

h(1,5) = (abs(nx-goal.x)+abs(ny-goal.y))

h(1,5)=(abs(1-5)+abs(5-4))

h(1,5)=abs(-4)+abs(1)

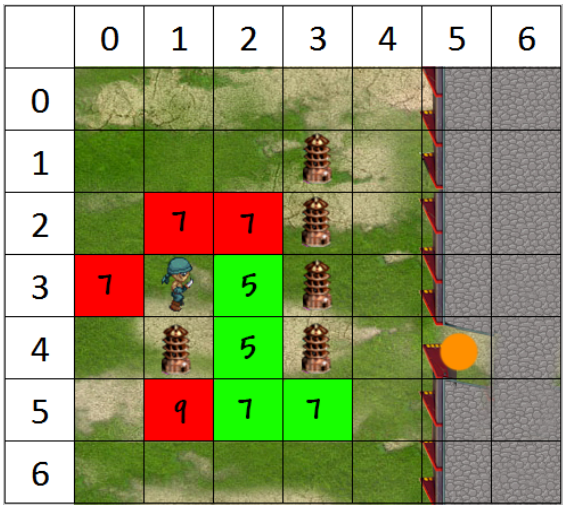
h(1,5)=4+1

h(1,5)=5

f(1,5)=g(2,5)+h(2,5)

f(1,5)=4+5

f(1,5)=9



Gambar 3.11 Langkah keempat pencarian best node

Setelah mendapatkan bestNode dilangkah ketiga, dilakukan penelusuran selanjutnya untuk mendapatkan bestnode selanjutnya dengan melakukan penelusuran simpul-simpul yang memiliki nilai cost terkecil. Simpul yang dihitung selanjutnya adalah koordinat (3,5) dan koordinat (1,5). Hasilnya adalah koordinat (3,5) mempunyai cost terkecil yaitu 7, maka koordinat (3,5) dipilih sebagai bestnode selanjutnya.

4. Langkah kelima

a. Koordinat (4,5)

Nilai nx=4 Nilai ny=5

Simpul Tujuan (5,4) sehingga goal.x=5, goal.y=4

g(4,5)=5

h(4,5) = (abs(nx-goal.x)+abs(ny-goal.y))

h(4,5)=(abs(4-5)+abs(5-4))

h(4,5)=abs(-1)+abs(1)

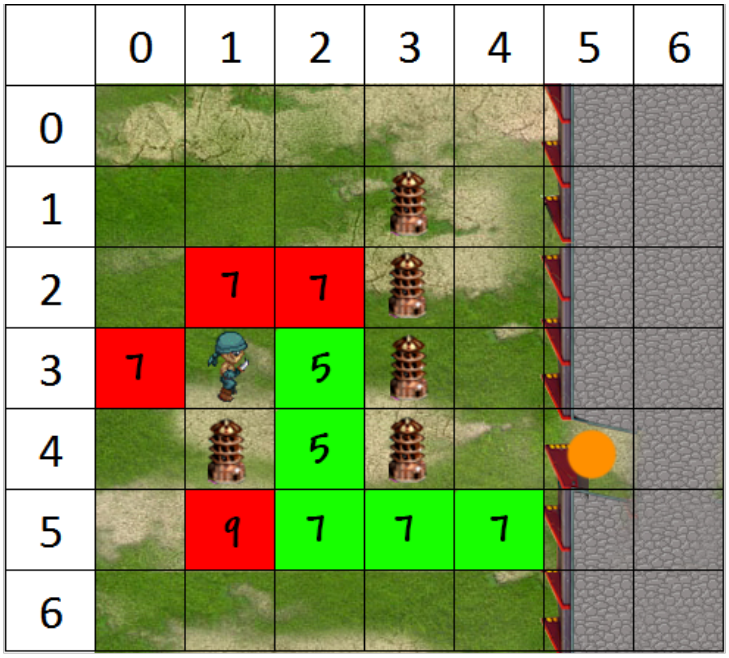
h(4,5)=1+1

h(4,5)=2

f(4,5)=g(4,5)+h(4,5)

f(4,5)=5+2

f(4,5)=7



Gambar 3.12 Langkah kelima pencarian best node

Setelah mendapatkan bestNode dilangkah keempat, dilakukan penelusuran selanjutnya untuk mendapatkan bestnode selanjutnya dengan melakukan penelusuran simpul-simpul yang memiliki nilai cost terkecil. Simpul yang dihitung selanjutnya adalah koordinat (4,5), koordinat (3,4) tidak dihitung atau diabaikan karena terdapat penghalang

yang tidak bisa dilewati musuh. Hasilnya adalah koordinat (4,5) mempunyai nilai cost 7, maka koordinat (4,5) dipilih sebagai bestnode selanjutnya.

5. Langkah keenam

a.Koordinat (5,5)

Nilai nx=5 Nilai ny=5

Simpul Tujuan (5,4) sehingga goal.x=5, goal.y=4

g(5,5)=6

h(5,5) = (abs(nx-goal.x)+abs(ny-goal.y))

h(5,5)=(abs(5-5)+abs(5-4))

h(5,5)=abs(0)+abs(1)

h(5,5)=0+1

h(5,5)=1

f(5,5)=g(5,5)+h(5,5)

f(5,5)=6+1

f(5,5)=7

b. Koordinat (4,4)

Nilai nx=4 Nilai ny=4

Simpul Tujuan (5,4) sehingga goal.x=5, goal.y=4

g(4,4)=6

h(4,4) = (abs(nx-goal.x)+abs(ny-goal.y))

h(4,4)=(abs(4-5)+abs(4-4))

h(4,4)=abs(-1)+abs(0)

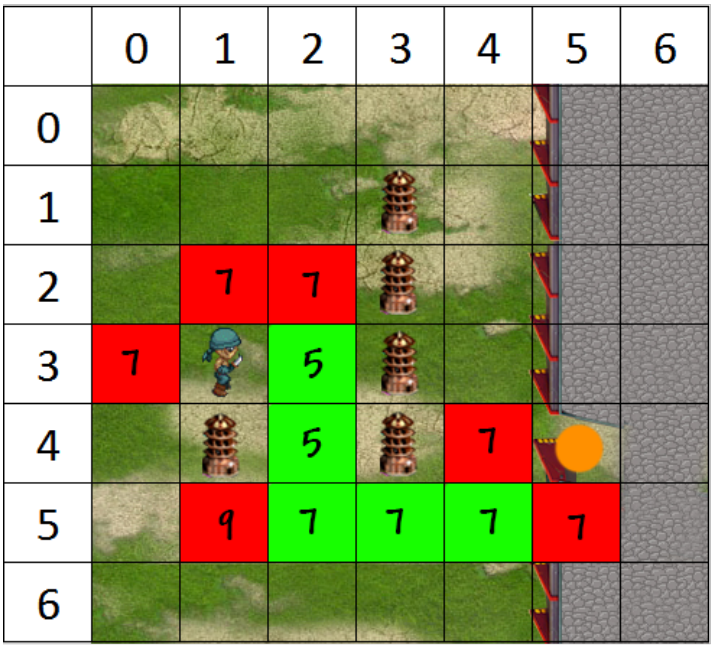
h(4,4)=1+0

h(4,4)=1

f(4,4)=g(4,4)+h(4,4)

f(4,4)=6+1

f(4,4)=7



Gambar 3.13 Langkah keenam pencarian best node

Setelah mendapatkan bestNode dilangkah kelima, dilakukan penelusuran selanjutnya untuk mendapatkan bestnode selanjutnya dengan melakukan penelusuran simpul-simpul yang memiliki nilai cost terkecil. Simpul yang dihitung selanjutnya adalah koordinat (5,5) dan (4,4). Hasilnya adalah koordinat (5,5) dan (4,4) mempunyai nilai cost yang sama yaitu 7, maka akan dihitung untuk langkah selanjutnya dengan menghitung koordinat (5,4) dan (6,5).

6. Langkah ketujuh

a. Koordinat (5,4)

Nilai nx=5 Nilai ny=4

Simpul Tujuan (5,4) sehingga goal.x=5, goal.y=4

g(5,4)=7

h(5,4) = (abs(nx-goal.x)+abs(ny-goal.y))

h(5,4)=(abs(5-5)+abs(4-4))

h(5,4)=abs(0)+abs(0)

h(5,4)=0+0

h(5,4)=0

f(5,4)=g(5,4)+h(5,4)

f(5,4)=7+0

f(5,4)=7

b. Koordinat (6,5)

Nilai nx=6 Nilai ny=5

Simpul Tujuan (5,4) sehingga goal.x=5, goal.y=4

g(6,5)=7

h(6,5) = (abs(nx-goal.x)+abs(ny-goal.y))

h(6,5)=(abs(6-5)+abs(5-4))

h(6,5)=abs(1)+abs(1)

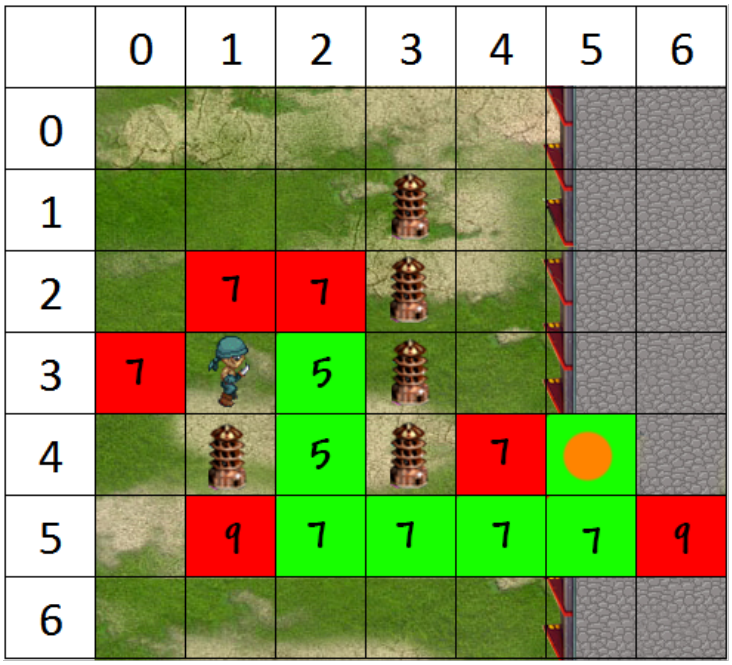
h(6,5)=1+1

h(6,5)=2

f(6,5)=g(6,5)+h(6,5)

f(6,5)=7+2

f(6,5)=9

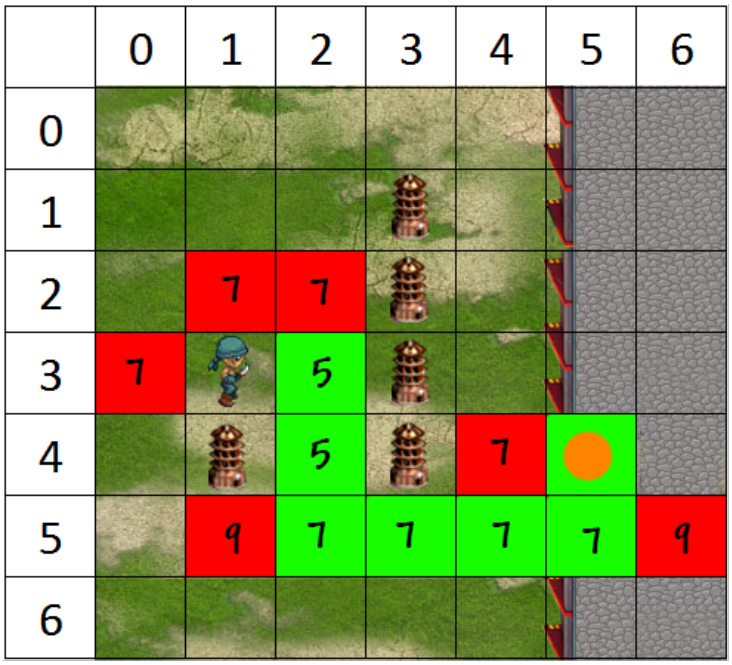


Gambar 3.14 Langkah ketujuh pencarian best node

Setelah mendapatkan bestNode dilangkah keenam, dilakukan penelusuran selanjutnya untuk mendapatkan bestnode selanjutnya dengan melakukan penelusuran simpul-simpul yang memiliki nilai cost terkecil. Simpul yang dihitung selanjutnya adalah

koordinat (6,5) dan (5,4). Hasilnya adalah koordinat (5,4) mempunyai cost paling kecil yaitu 7, maka koordinat (5,4) dipilih sebagai bestnode selanjutnya.

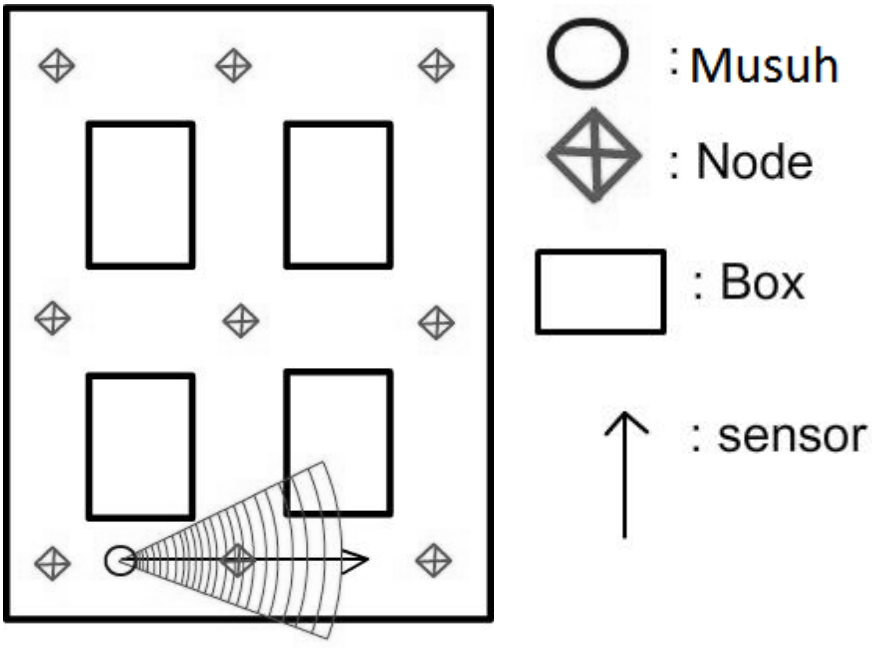
Dari perhitunganyang telah dilakukan dipilihlah biaya / cost terkecil pada setiap langkahnya sehingga akan menghasilkan jalur terpendek yang akan dilalui musuh untuk melakukan pencarian posisi pemain yang terlihat pada gambar 3.9.



Gambar 3.15 Hasil pencarian jalan menggunakan algoritma A*

2. Analisis Waypoint

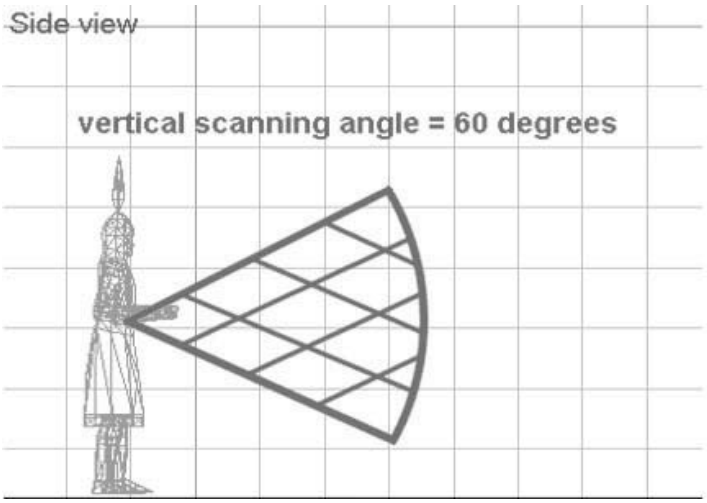
pada metode ini kami menggunakan dua algoritma untuk dapat berpatroli dengan random. Menggunkan funsi scan yang berguna untuk memprediksi apakah ada objek di sekitar daerah tesebut, dan berikutnya menggunakan waypoint untuk menentukan jalur.



Gambar 3.16 *Bot* men-*scan node*

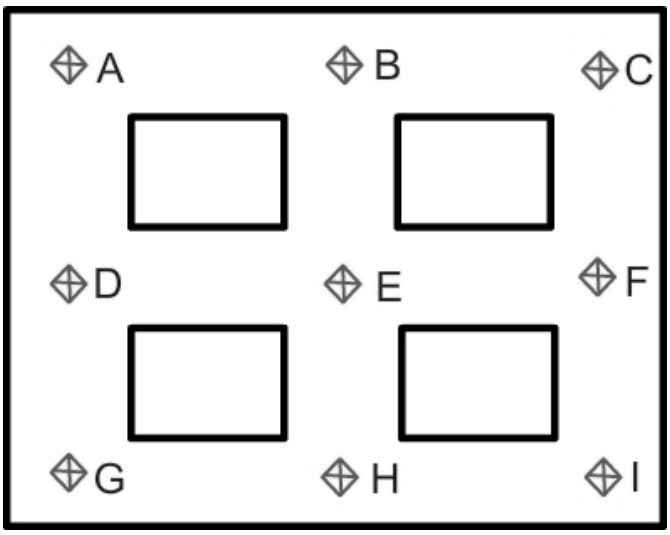
Penjabaran alogritma berdasarkan gambar di atas, yaitu sebagai berikut :

```
1   function scan(distance,radius,tilt){
2     var temp;
3     temp.pan = radius;
4     temp.tilt = tilt;
5     temp.z = distance;
6     scan_entity (musuh, temp);
7   }
8   if( scan(300,120,60) > 0 && sensor_1 > 100 ){
9     c_move(musuh,vector(5*time,0,0));
10  }
11  else{
12    musuh.rotation += 3;
13  }
```

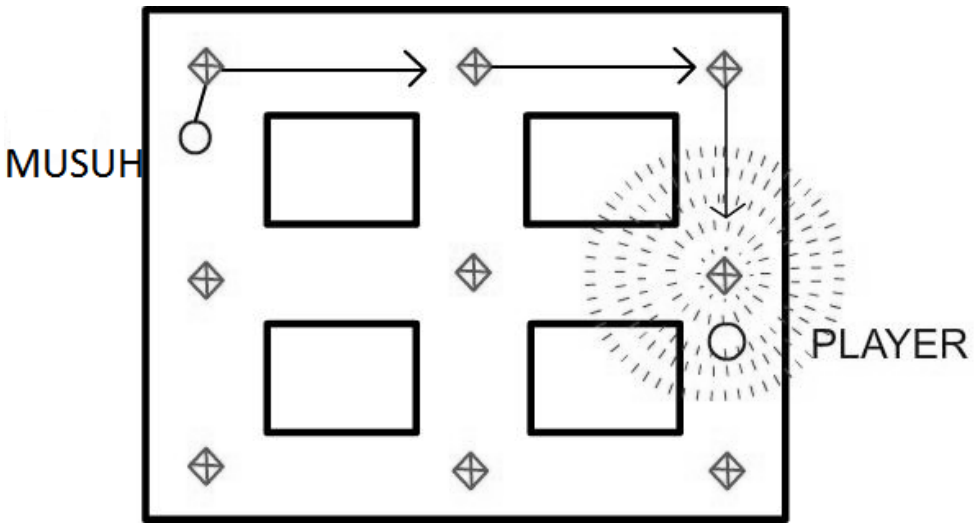


Gambar 3.17 Proses Scanning

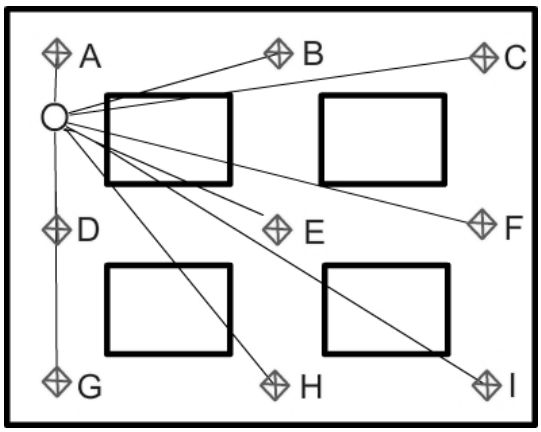
Pancarian yang kami gunakan menggunakan algoritma *data-based search*. Dengan metode komputer tidak perlu lagi menghitung jarak terpendek untuk menuju *node* yang dituju. Pada gambar dibawah merupakan contoh dimana musuh mencari posisi player berada.



Gambar 3.18 Label Pada Node



Gambar 3.19 Musuh Mencari Posisi Player



Gambar 3.20 Musuh Mencari Rute Posisi Player

Tabel 3.3 – *Complit Tabel Node*

	A	B	C	D	E	F	G	H	I
A	-	B	B	D	D	B	D	B	I
B	A	-	C	C	E	C	A	E	C
C	B	B	-	F	F	F	F	B	F
D	A	A	A	-	E	E	G	G	G
E	B	B	B	D	-	F	H	H	H
F	C	C	C	E	E	-	I	I	I
G	D	D	D	D	D	D	-	H	H
H	E	E	E	E	E	E	G	-	I
I	F	F	F	F	F	F	H	H	-

Pertama memberikan label pada tiap node pada gambar 3.20, lalu tiap node diberikan fungsi scan untuk mendeteksi beberadaan player dan mendeteksi keberadaan musuh, musuh akan mendekati posisi node yang terdekat dengan player. Pada gambar 3.19, musuh bergerak mendekati player.

Lalu bagaimana bot dapat bergerak ke node A pada awal-awalnya kenapa tidak ke node D padahal lebih dekat? (gambar 3.18 dan 3.19). Sebelum masuk pada tabel, node tersebut men-*scan* musuh seperti pada gambar 3.20, lalu dicari mana nilai jarak yang terpendek dari semua kemungkinan yang ada.

3.1.4 Analisis Kebutuhan Non Fungsional

3.1.4.1 Analisis Kebutuhan Perangkat Keras

Analisis perangkat keras digunakan untuk mengetahui spesifikasi perangkat keras dalam membangun dan menjalankan aplikasi permainan *the tale of hunter***,** dalam kebutuhan kebutuhan perangkas keras di bagi menjadi dua yaitu spesifikasi perangkat keras developer dan spesifikasi perangkat keras pengguna.

Berikut merupakan spesifikasi perangkat keras untuk membangun palikasi permainan *The Tale of Hunter*.

Tabel 3.4 Spesifikasi Perangkat Keras *Developer*

Nama Perangkat	Spesifikasi
Prosesor	2.20 GH
RAM	2 GB
Harddisk	40 GB
Display	Resolusi 1366 x 768
Keyboard	Standar
Mouse	Standar

Berikut merupakan spesifikasi perangkat keras untuk membangun aplikasi permainan *The Tale of Hunter*.

Tabel 3.5 Spesifikasi Perangkat Keras Pengguna

Nama Perangkat	Spesifikasi
Prosesor	1.0 GH
RAM	256 MB
Harddisk	20 GB
Display	Resolusi 800 x 600
Keyboard + Mouse	Standar

3.1.4.2 Analisis Kebutuhan Perangkat Lunak

Analisis perangkat lunak digunakan untuk mengetahui spesifikasi perangkat lunak dalam membangun dan menjalankan aplikasi permainan *the tale of hunter***.**

Berikut merupakan spesifikasi perangkat keras untuk membangun aplikasi permainan *The Tale of Hunter*.

Tabel 3.6 Spesifikasi Perangkat Lunak Developer

Nama Perangkat Lunak	Spesifikasi
Sistem Operasi	Microsoft Windows XP
Tools	JDK 6 dan Greenfoot
	Adobe Illustrator
	Adobe Photoshop

Tabel 3.7 Spesifikasi Perangkat Lunak Pengguna

Nama Perangkat Lunak	Spesifikasi
Sistem Operasi	Microsoft Windows XP
Tools	JDK 6

3.1.4.3 Analisis Kebutuhan Perangkat Pikir

Analisis kebutuhan perangkat pikir digunakan untuk mengetahui spesifikasi pengguna untuk memainkan aplikasi *game The Tale of Hunter*, seperti pada tabel 3.7

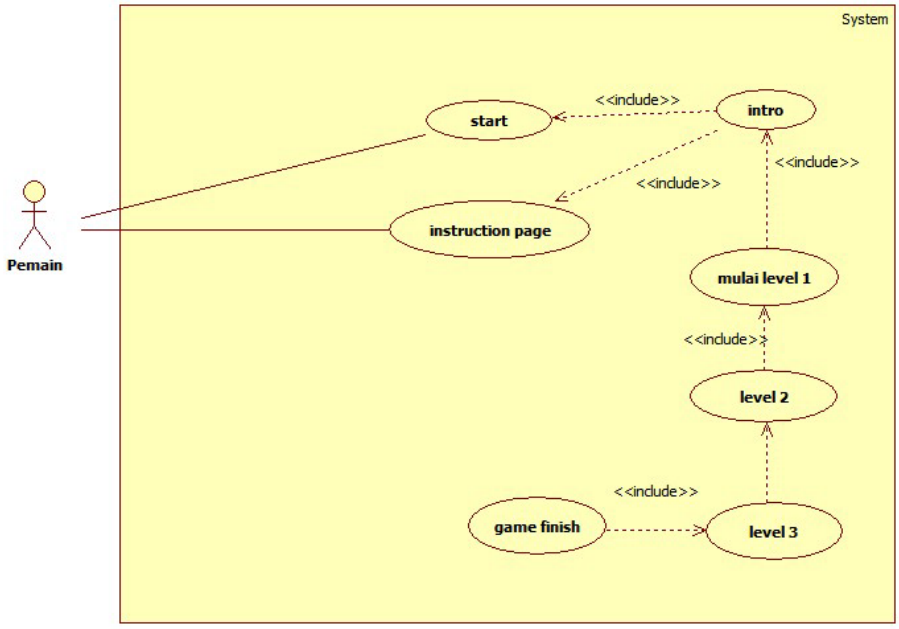
Tabel 3.8 Kebutuhan Perangkat Pikir

Peran	Pemain
Kondisi fisik	Tidak cacat fisik pada tangan kanan dan kiri, serta tidak buta warna
Usia	12 tahun keatas
Tingkat Ketrampilan	dapat menggunakan komputer
Pengalaman	Pernah memainkan aplikasi *game* lainnya seperti garden maze

3.1.5 Analisis Kebutuhan Kebutuhan Fungsional

3.1.5.1 Diagram *Use Case*

Secara umum, interaksi antara penggunakan yaitu pemain dengan aplikasi permainan *the tale of hunter* di gambarkan pada diagram *Use Case*, seperti pada gambar 3.19



Gambar 3.21 Diagram *Use Case The Tale of Hunter*

3.1.5.2 Skenario *Use Case*

Skenario pada diagram *use case* berisi tentang kejadian *'flow if event' use case* utama. Di dalamnya menjelaskan urutan interaksi antara aktor dengan *use case* tersebut dari awal sampai akhir. Adapun scenario *use case* yang terdapat pada aplikasi permainan *the tale of hunter* ini adalah sebagai berikut :

1. **Skenario *Use Case* Tampil *Main***

   Skenario *use case* ini menjelaskan interaksi antara aktor, yaitu pemain, dengan *use case main* yang di jelaskan pada tabel 3.9

   Nama Use Case : Main

   Skenario :

Tabel 3.9 Skenario *Use Case Main*

Aksi Aktor	Reaksi Sistem
Skenario Normal	
	a. Menampilkan menu *main*
b. Pilih Start	
	c. Menampilkan Intro
Skenario Alternatif	
	a. Menampilkan menu *main*
b. Pilih *Instructions*	
	c.Menampilkan menu *instructions*

2. Skenario *Use Case* Tampil *Instruction*

Skenario *use case* ini menjelaskan interaksi antara aktor, yaitu pemain, dengan *use case main* yang di jelaskan pada tabel 3.10

Nama Use Case : Instruction

Skenario :

Tabel 3.10 Skenario *Use Case* Tampil *Main*

Aksi Aktor	Reaksi Sistem
Skenario Normal	
	a. Menampilkan menu *intruction*
b. pilih *next*	
	c. Menampilkan intro
Skenario Alternatif	
	a. Menampilkan menu *intruction*
b. pilih *back*	
	c.Menampilkan menu *main*

3. Skenario Use Case Mulai Permainan *Level* 1

Skenario *use case* ini menjelaskan interaksi antara aktor, yaitu pemain, dengan *use case* mulai permainan *Level* 1 yang dijelaskan pada tabel 3.11

Nama *Use Case* : Mulai Permainan *Level* 1

Skenario :

Tabel 3.11 Skenario *Use Case* Mulai Permainan *Level* 1

Aksi Aktor	Reaksi Sistem
Skenario Normal	
	a. Tampilkan layar permainan pada level 1
	b. Inisialisasi posisi penghalang
	c. Inisialisasi posisi pemain
	d. Insialisasi posisi item
	e. Insialisasi posisi musuh
f. Menyelesaikan permainan	
	g. Berpindah ke level selanjutnya
Skenario Alternatif	
	a. Tampilkan layar permainan pada level 1
	b. Inisialisasi posisi penghalang
	c. Inisialisasi posisi pemain
	d. Insialisasi posisi item
	e. Insialisasi posisi musuh
f. Misi permainan tidak terselesaikan	
	g. Kembali ke level 1
	h. Tampilkan layar permainan pada level 1
	i. Inisialisasi posisi penghalang
	j. Inisialisasi posisi pemain

	k. Insialisasi posisi item
	l. Insialisasi posisi musuh
m. Menyelesaikan permainan	
	n. Berpindah ke level selanjutnya

4. Skenario Use Case *Level* 2

Skenario *use case* ini menjelaskan interkasi antara aktor, yaitu pemain, dengan *use case Level* 2 yang dijelaskan pada tabel 3.12

Nama *Use Case* : *Level* 2

Skenario:

Tabel 3.12 Skenario *Use Case Level* 2

Aksi Aktor	Reaksi Sistem
Skenario Normal	
	a. Tampilkan layar permainan pada level 2
	b. Inisialisasi posisi penghalang
	c. Inisialisasi posisi pemain
	d. Insialisasi posisi item
	e. Insialisasi posisi musuh
f. Menyelesaikan permainan	
	g. Berpindah ke level selanjutnya
Skenario Alternatif	
	a. Tampilkan layar permainan pada level 2
	b. Inisialisasi posisi penghalang
	c. Inisialisasi posisi pemain
	d. Insialisasi posisi item
	e. Insialisasi posisi musuh

f. Misi permainan tidak terselesaikan	
	g. Kembali ke level 1
	h. Tampilkan layar permainan pada level 1
	i. Inisialisasi posisi penghalang
	j. Inisialisasi posisi pemain
	k. Insialisasi posisi item
	l. Insialisasi posisi musuh
m. Menyelesaikan permainan	
	n. Berpindah ke level selanjutnya

5. Skenario Use Case *Level* 3

Skenario *use case* ini menjelaskan interaksi antara aktor, yaitu pemain, dengan use *case Level* 3 yang di jelaskan pada tabel 3.13

Nama *Use Case* : *Level* 3

Skenario :

Tabel 3.13 Skenario *Use Case Level* 3

Aksi Aktor	Reaksi Sistem
Skenario Normal	
	a. Tampilkan layar permainan pada level 3
	b. Inisialisasi posisi penghalang
	c. Inisialisasi posisi pemain
	d. Insialisasi posisi item
	e. Insialisasi posisi musuh
g. Menyelesaikan permainan	
	h. Berpindah ke level selanjutnya
Skenario Alternatif	

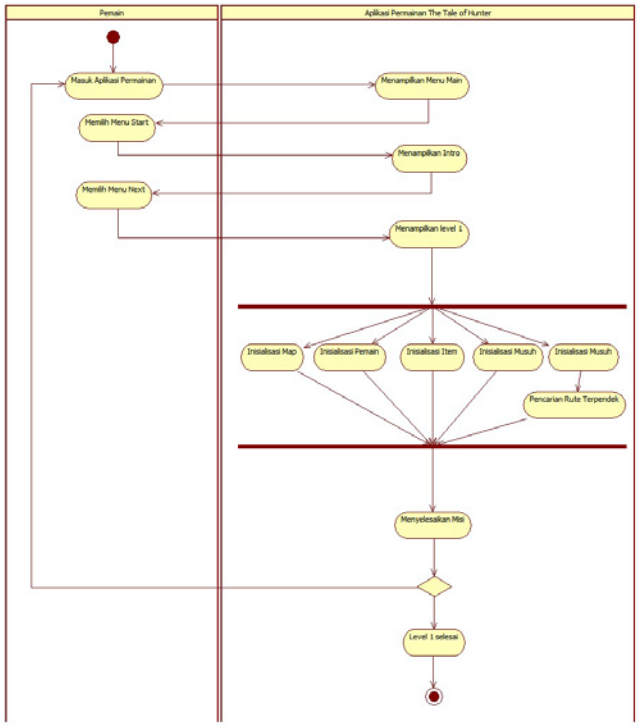
	a. Tampilkan layar permainan pada level 3
	b. Inisialisasi posisi penghalang
	c. Inisialisasi posisi pemain
	d. Insialisasi posisi item
	e. Insialisasi posisi musuh
f. Misi permainan tidak terselesaikan	
	g. Kembali ke level 1
	h. Tampilkan layar permainan pada level 1
	i. Inisialisasi posisi penghalang
	j. Inisialisasi posisi pemain
	k. Insialisasi posisi item
	l. Insialisasi posisi musuh
m. Menyelesaikan permainan	
	n. Berpindah ke level selanjutnya

3.1.5.3Diagram *Activity*

Diagram ini berisi penjelasan secara lengkap mengenai urutan aktivitas yang terjadi dengan aplikasi *The Tale of Hunter.*

A. Diagram *Activity* Permainan *Level* 1

Proses ini terjadi saat permainan di jalankan. Sistem akan menampilkan arena permainan dimana pemain dapat langsung memainkan level 1 dan menyelesaikan misi permainan pada *level* 1, apabila misi tidak terselesaikan maka pemain kembali ke permainan *level* 1 lagi. Diagram *activity* mulai permainan *level* 1 terdapat pada gambar 3.22



Gambar 3.22 Diagram *Activity* Level 1 The Tale of Hunter

B. Diagram *Activity* Level 2

Proses ini terjadi saat pemain telah menyelesaikan palikasi permainan *The Tale of Hunter* pada *level* 1. Sistem akan menampilkan arena permainan *level* 2 dan pemain harus menyelesaikan misi permainan pada *level* 2, apabila misi permainan tidak terselesaikan maka pemain akan memulai permainan dari *level* 1. Diagram *activity level* 2 terdapat pada gambar 3.23

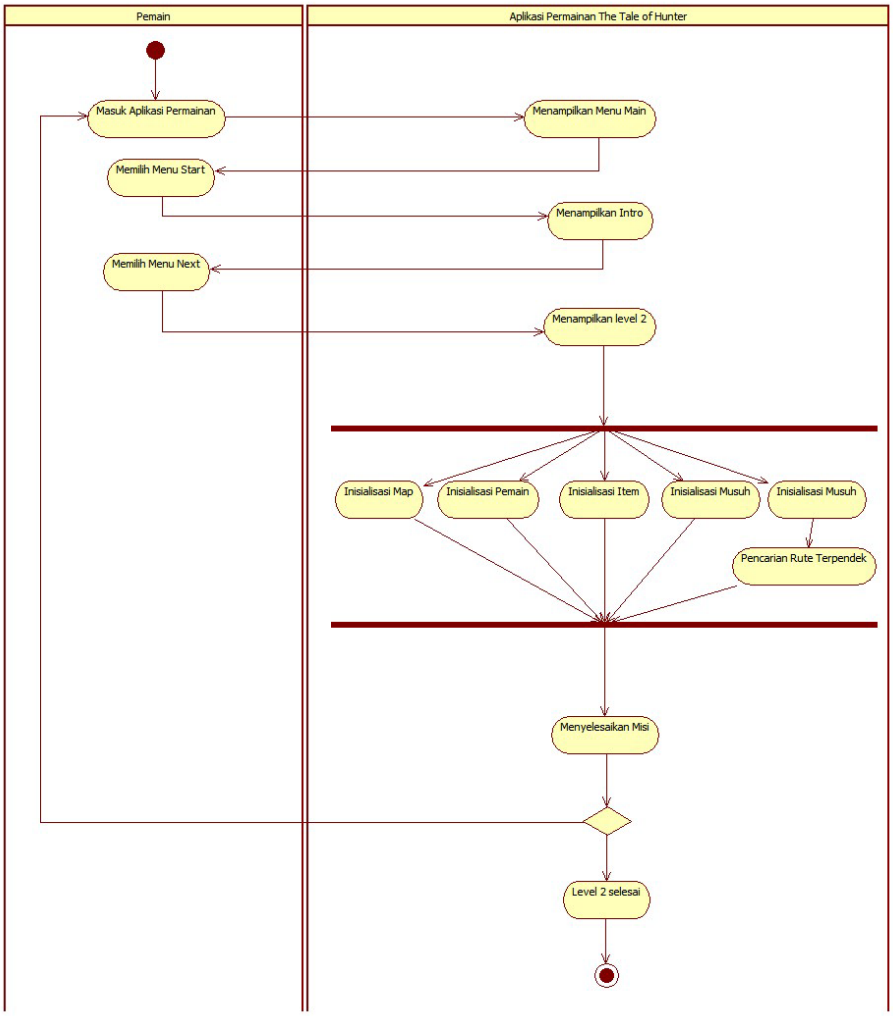
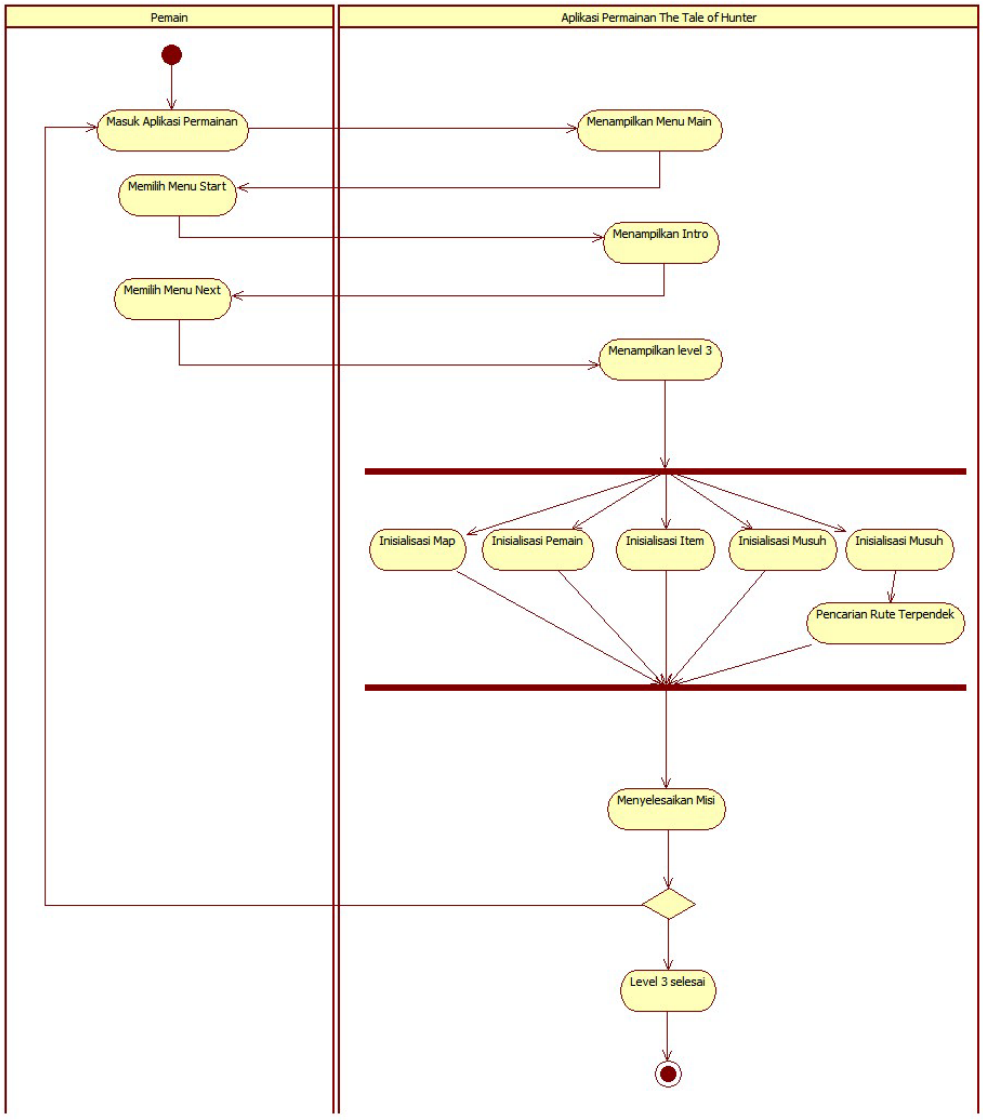


Diagram 3.23 Diagram *Activity Level* 2 *The Tale of Hunter*

C. Diagram *Activity* level 3

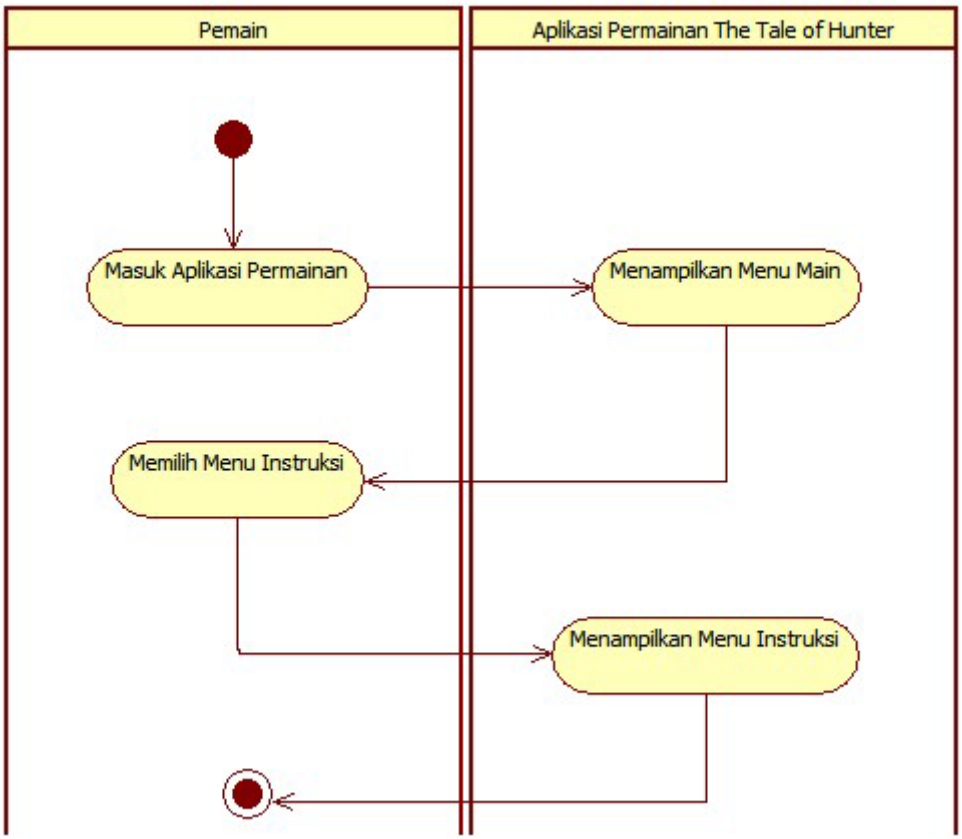
Proses ini terjadi saat pemain telah menyelesaikan palikasi permainan *The Tale of Hunter* pada *level* 3. Sistem akan menampilkan arena permainan *level* 3 dan pemain harus menyelesaikan misi permainan pada *level* 3, apabila misi permainan tidak terselesaikan maka pemain akan memulai permainan dari *level* 1. Diagram *activity level* 3 terdapat pada gambar 3.24



Gambar 3.24 Diagram *Activity Level* 3 *The Tale of Hunter*

D. Diagram *Activity* Tampil Petunjuk

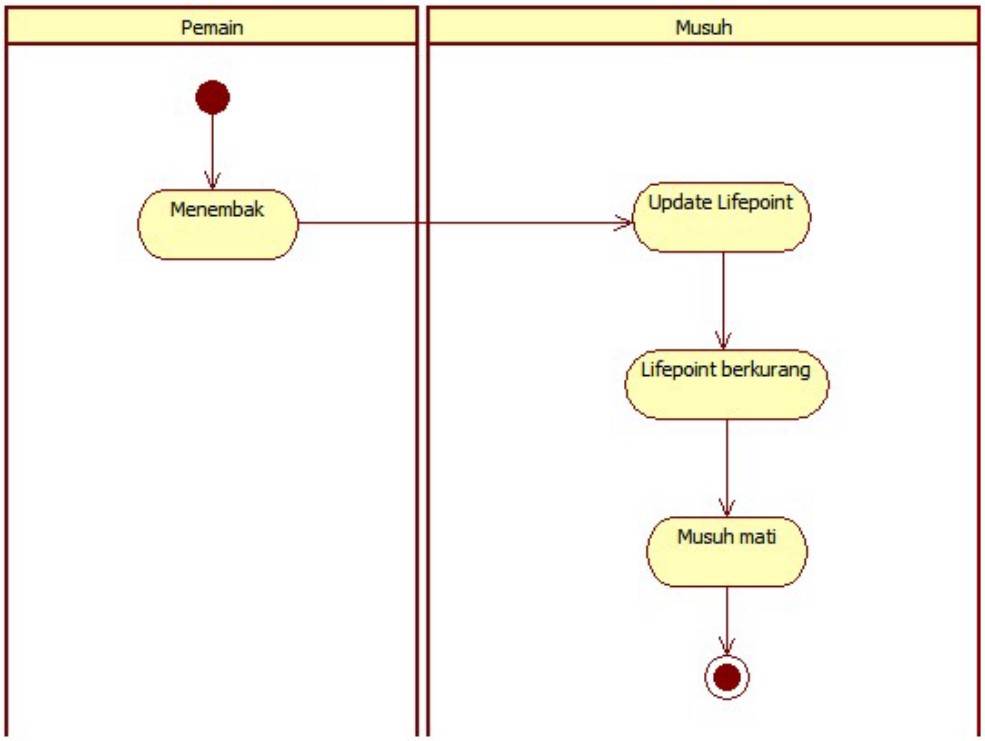
Proses ini terjadi saat pemain memilih menu instruksi pada aplikasi permainan *The Tale of Hunter*. Sistem akan menampilkan petunjuk permainan dari aplikasi permainan *The Tale of Hunter*. Diagram tampil petunjuk terdapat pada gambar 3.25



Gambar 3.25 Diagram *Activity* Tampil Pentunjuk *The Tale of Hunter*

E. Diagram *activity* menembak musuh

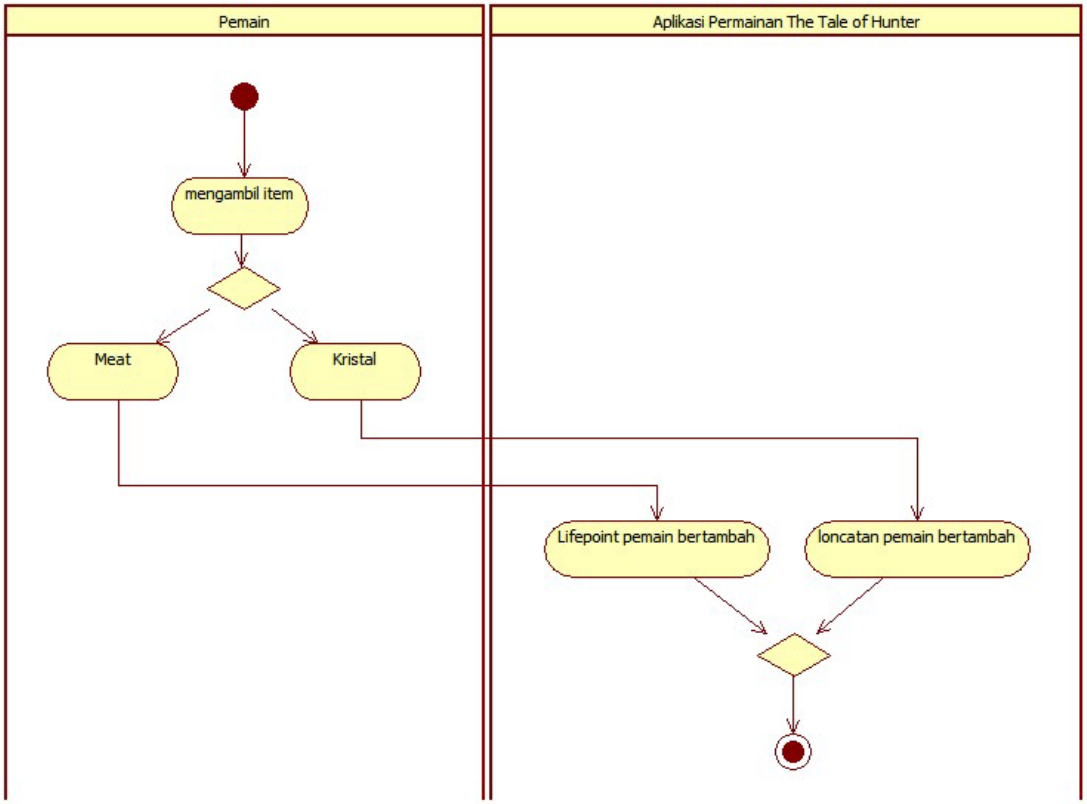
Proses ini terjadi ketika pemain menembak musuh. Diagram activity menembak musuh terdapat pada gambar 3.26



Gambar 3.26 Diagram *Activity* mengambil item *The Tale of Hunter*

F. Diagram *activity* mengambil item

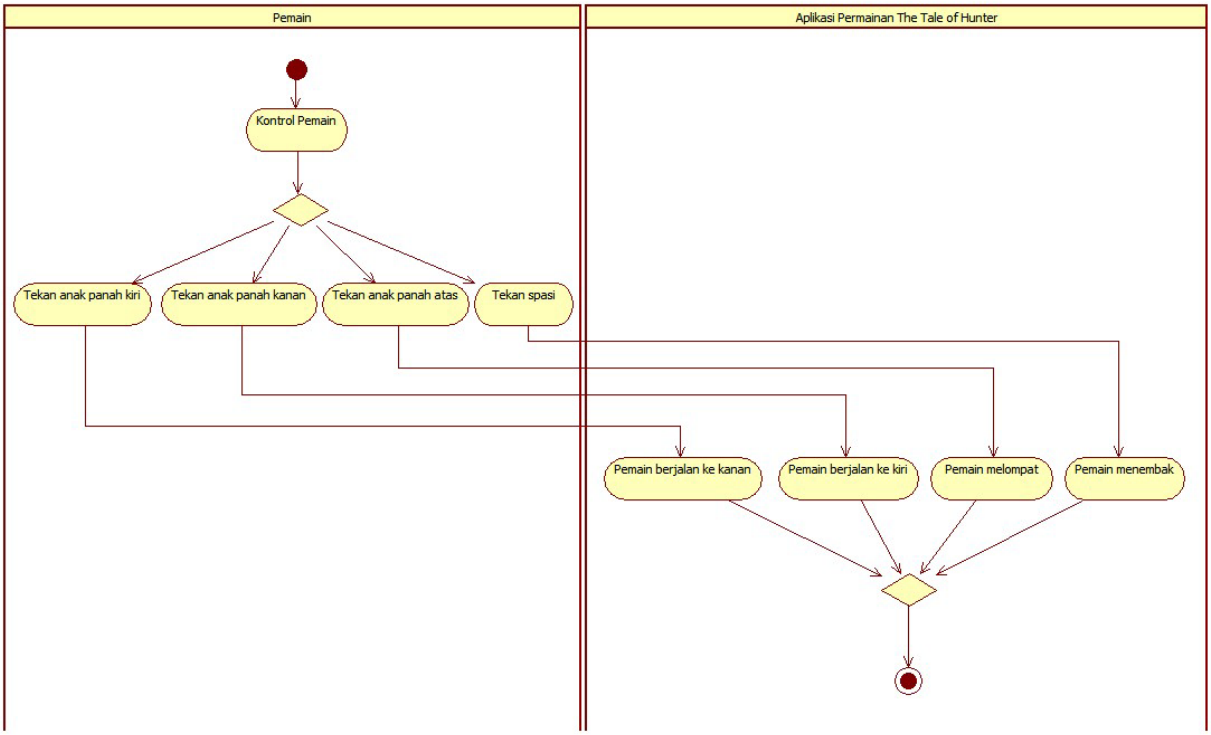
Proses ini terjadi ketika pemain mengambil item. Diagram activity mengambil item terdapat pada gambar 3.27



Gambar 3.27 Diagram *Activity* mengambil *item The Tale of Hunter*

G. Diagram *activity* kontrol pemain

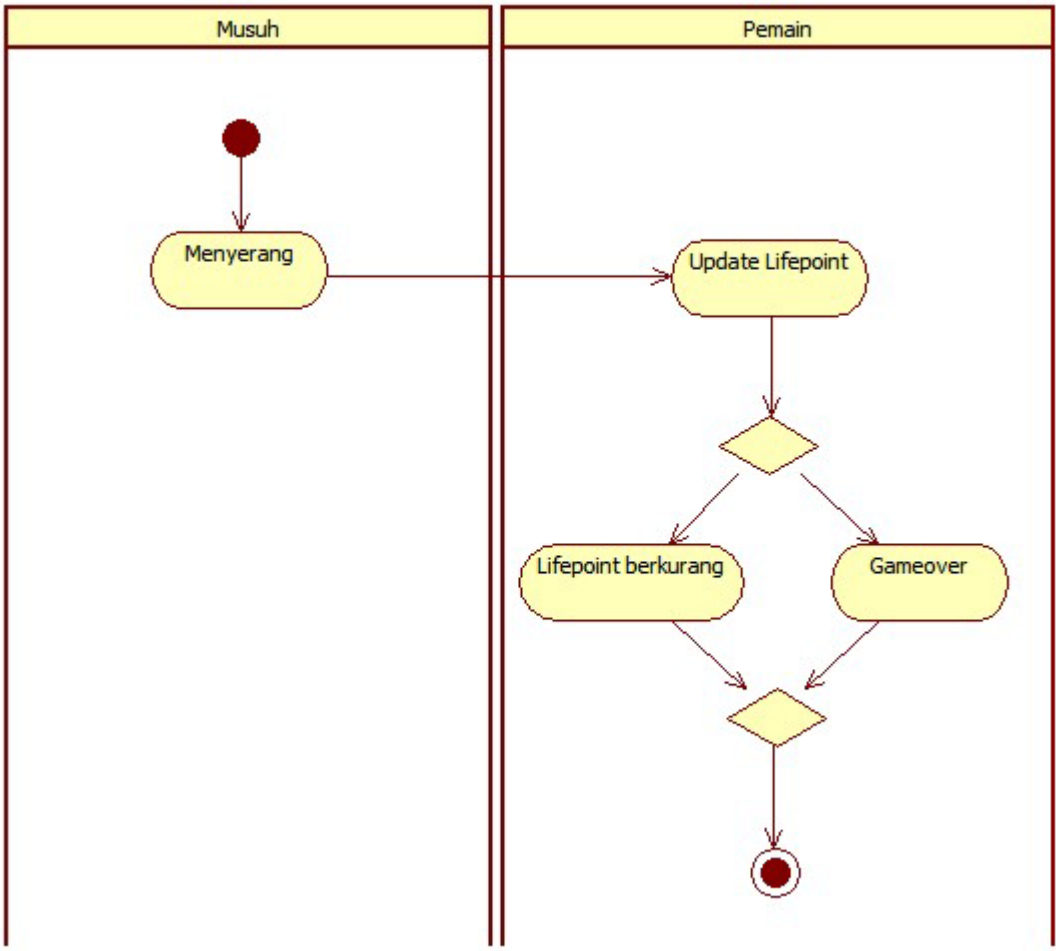
Proses ini terjadi ketika pemain mengontrol katakter utama. Diagram *activity* kontrol pemain terdapat pada gambar 3.28



Gambar 3.28 Diagram *Activity* kontrol pemain *The Tale of Hunter*

H. Diagram activity musuh menyerang

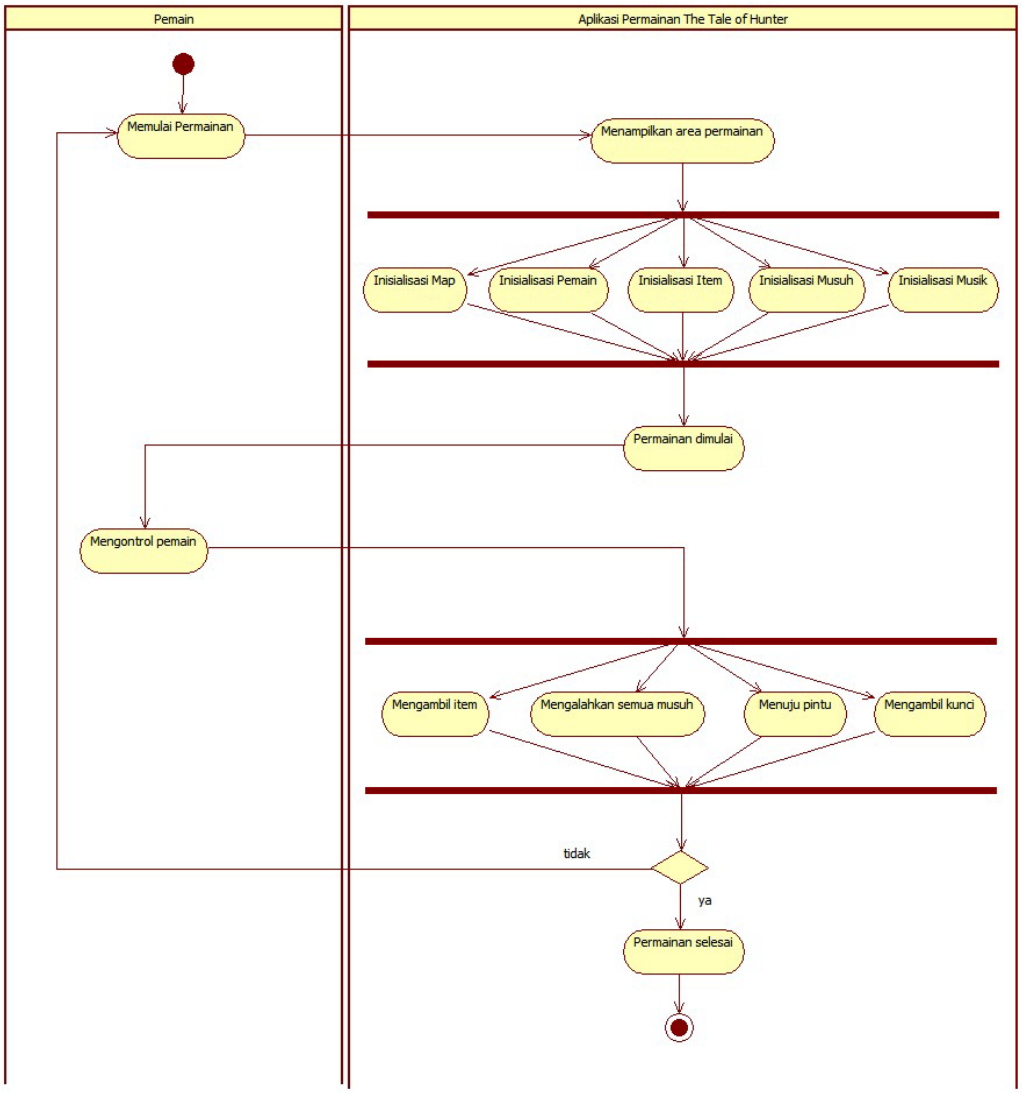
Proses ini terjadi ketika musuh menyerang katakter utama. Diagram *activity* musuh menyerang terdapat pada gambar 3.29



Gambar 3.29 Diagram *Activity* musuh menyerang *The Tale of Hunter*

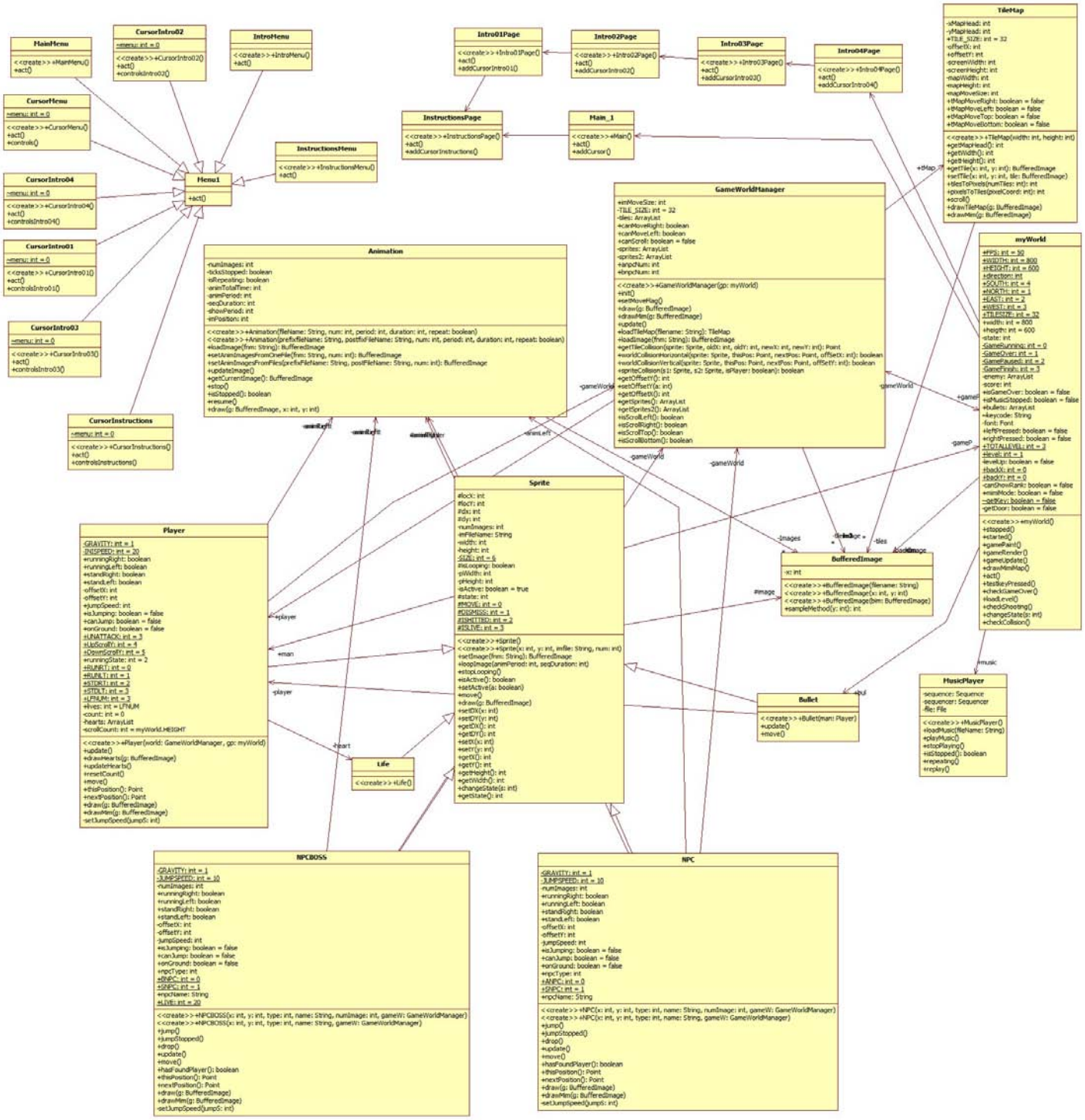
I. Diagram *activity* misi utama

Proses ini adalah proses misi utama yang harus diselesaikan oleh pemain Diagram *activity* misi utama terdapat pada gambar 3.30



Gambar 3.30 Diagram *Activity* misi utama *The Tale of Hunter*

3.1.5.4Diagram *Class*



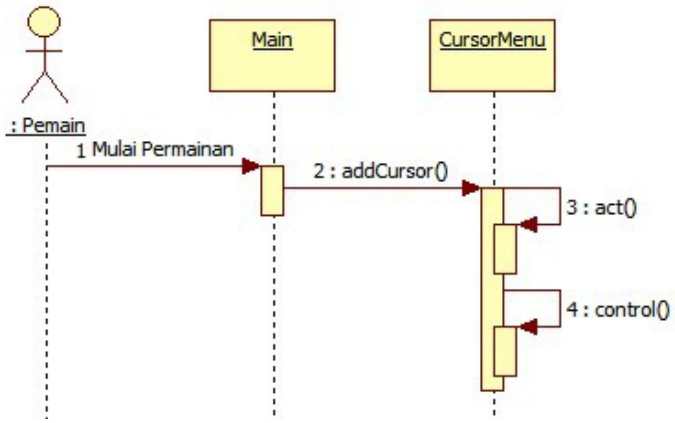
Gambar 3.31 ***Class Diagram The Tale of Hunter***

3.1.5.5Diagram *Sequence*

Berikut merupakan diagram *sequence* yang terdapat pada aplikasi permainan *the tale of hunter* berdasarkan *use case* yang ada.

Diagram *sequence* mulai permainan *level* 1 terdapat pada gambar 3.32.

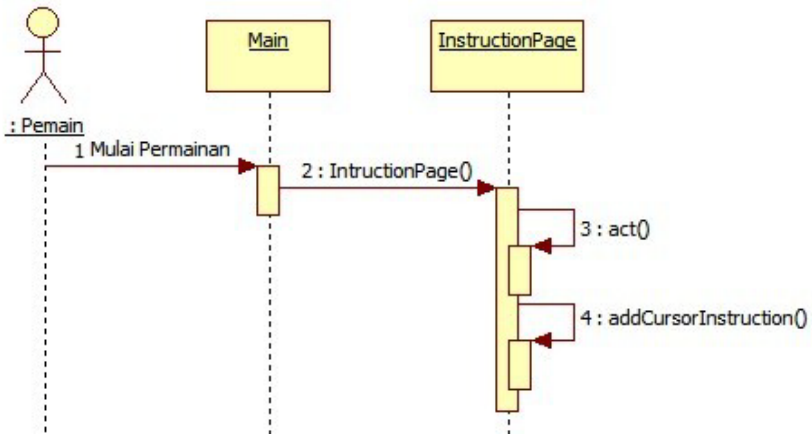
A. Diagram *Sequence* dari *Use Case* : *main*



Gambar 3.32 *Diagram Sequence untuk Use Case main*

B. Diagram Sequence *Use Case : instructions page*

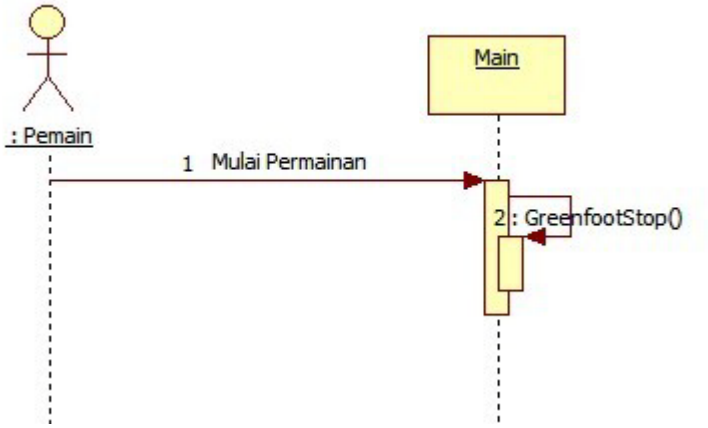
Diagram sequence instructions page terdapat pada gambar 3.33.



Gambar 3.33 *Diagram Sequence untuk Use Case instructions page*

C. Diagram Sequence *Use Case : exit*

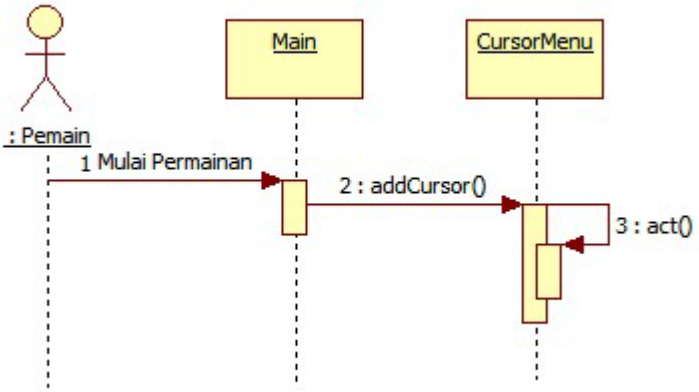
Diagram *sequence Exit* terdapat pada gambar 3.34.



Gambar 3.34 *Diagram Sequence untuk Use Case exit*

D. Diagram *Sequence Use Case* : *start*

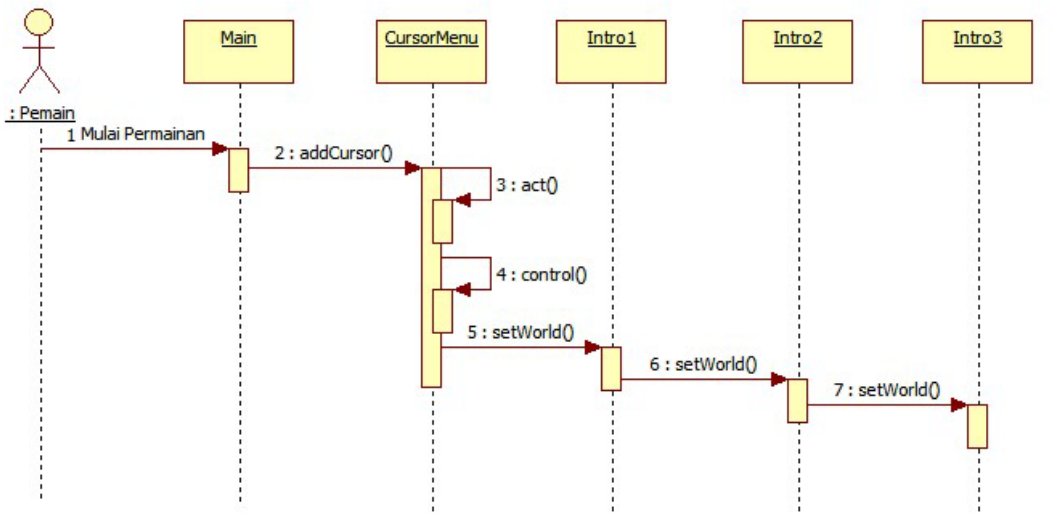
Diagram *sequence Exit* terdapat pada gambar 3.35.



Gambar 3.35 *Diagram Sequence untuk Use Case start*

E. *Diagram Sequence Use Case* : *intro*

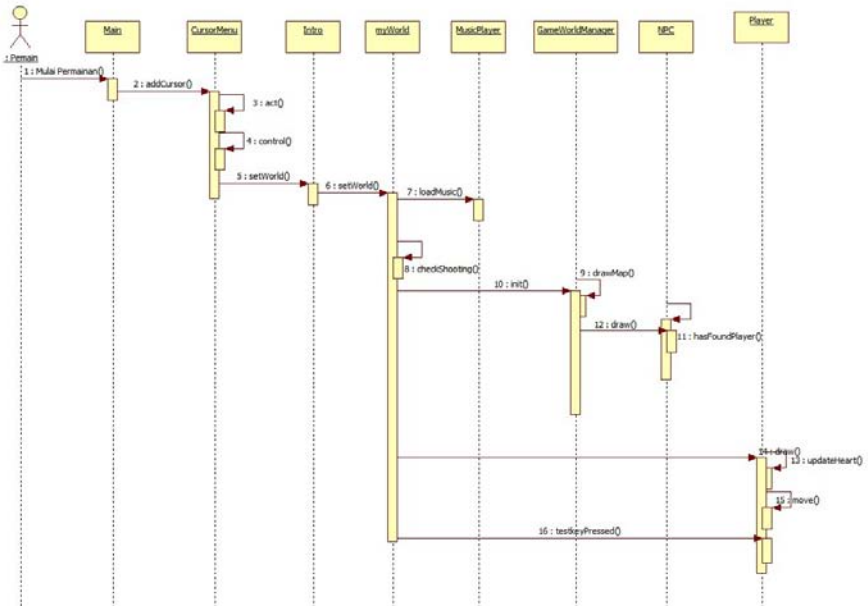
Diagram *sequence intro* terdapat pada gambar 3.36.



Gambar 3.36 *Diagram Sequence* untuk *Use Case intro*

F. Diagram *Sequence Use Case* : *level* 1 dan 2

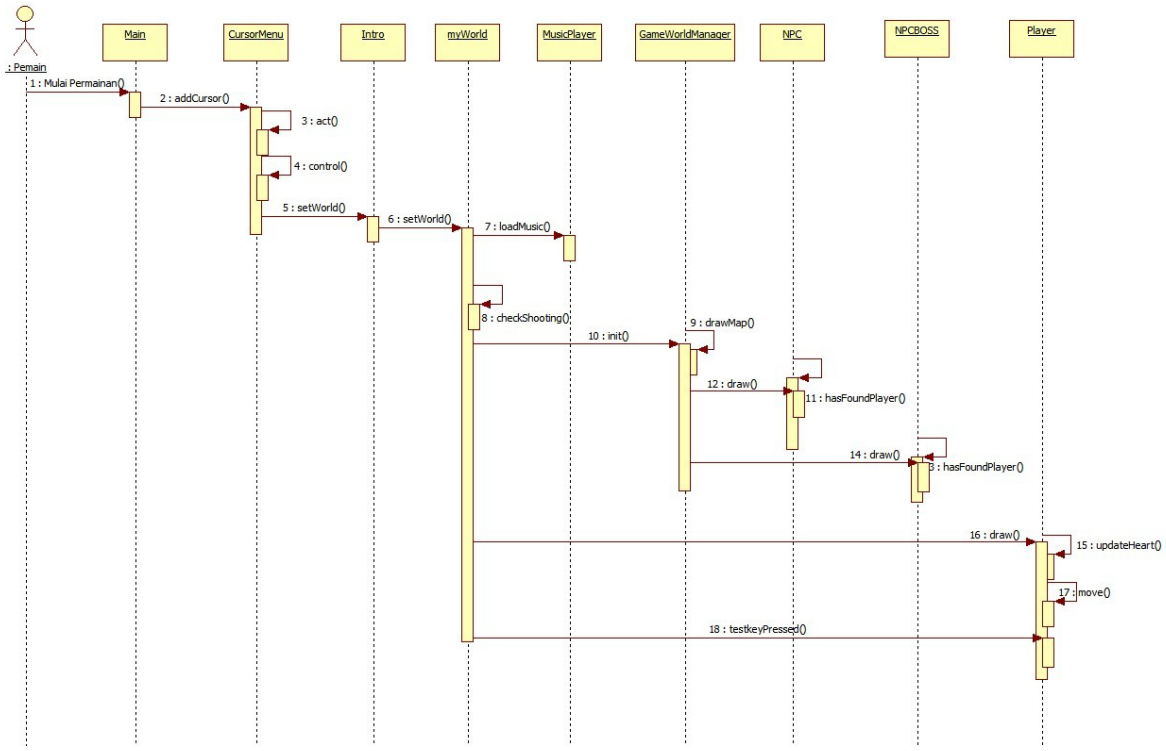
Diagram *sequence level* 1 dan 2 terdapat pada gambar 3.37.



Gambar 3.37 *Diagram Sequence* untuk *Use Case level 1* dan *level 2*

G. Diagram *Sequence Use Case* : *level 3*

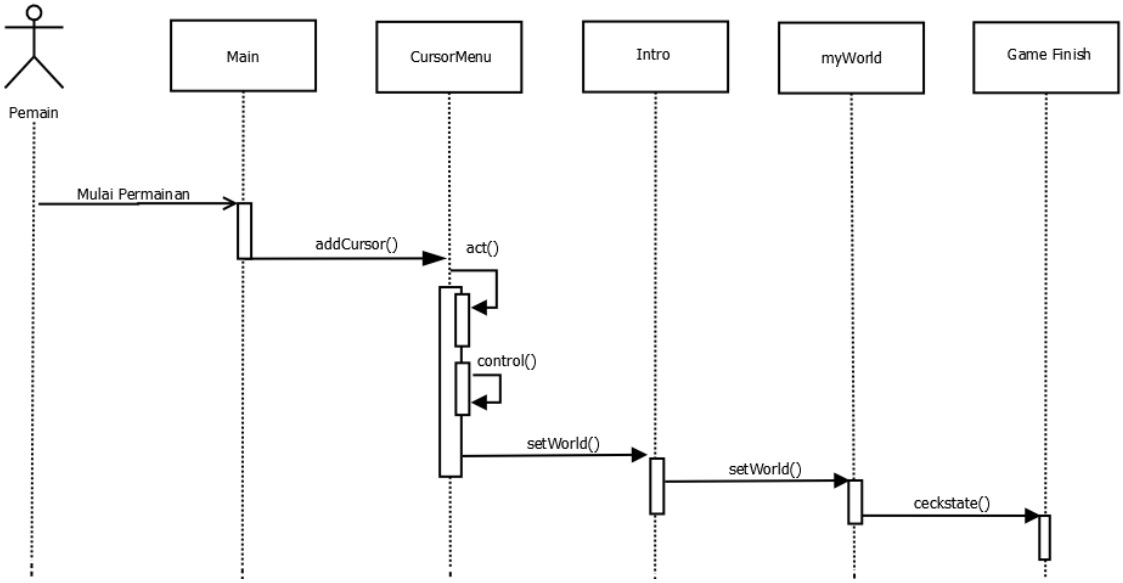
Diagram *sequence level 3* terdapat pada gambar 3.38.



Gambar 3.38 *Diagram Sequence* untuk *Use Case level 3*

H. Diagram *Sequence Use Case* : *game finish*

Diagram *sequence game finish* terdapat pada gambar 3.39



Gambar 3.39 *Diagram Sequence* untuk *Use Case game finish*

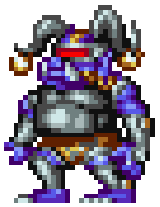
3.2 Perancangan Sistem

3.2.1 Perancangan Komponen Permainan

3.2.1.1 Karakter

Karakter atau entitas dalam *game The Tale of Hunter* terbagi menjadi dua, yaitu karakter pemain dan NPC, adapun terdapat *item* yang digunakan untuk melanjutkan pemain ke *level* selanjutnya. Karakter dan item yang terdapat didalam *game the tale of hunter* ditunjukan pada tabel 3.8

Tabel 3.14 Karakter dan *item*

No	Nama	Gambar	Penjelasan
1	Hunter		Pada karakter pemain ini, merupakan karakter utama yang dimainkan. Karakter ini dapat menembak musuh
2	Bandit		Pada NPC (*Non-Payer Characters*) ini, merupakan karakter musuh pada level, jika pemain menyentuh musuh ini maka akan langsung kalah. Musuh ini mempunyai lifepoint
3	The Owl		Pada NPC (*Non-Payer Characters*) ini, merupakan karakter musuh
4	Bapomet		Pada NPC (*Non-Payer Characters*) ini, merupakan karakter musuh

5	Fox		Pada NPC (*Non-Payer Characters*) ini, merupakan karakter musuh
5	Bullet		Item ini merupakan sejata untuk mengalahkan musuh
6	Meat		Item ini berfungsi untuk menambahkan *lifepoint*
	Kristal		Item ini berfungsi untuk menambahkan *power*
6	Kunci		Item ini merupakan kunci untuk melanjutkan ke level selanjutnya
7.	Pintu		Merupakan pintu untuk melanjutkan level selanjutnya
8.	Tower upgrade level 1		Merupakan tower level 1 untuk menghadang musuh yg menerang istana

9.	Tower upgrade level 2		Merupakan tower level 2 untuk menghadang musuh yg menerang istana

3.2.1.2 *Story Board*

Storyboard aplikasi *game The Tale of Hunter*, terdapat pada gambar 3.39



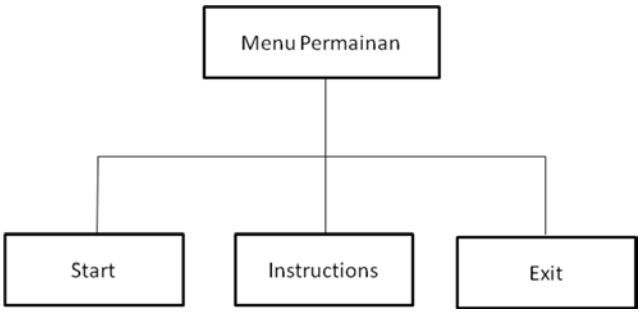
Gambar 3.40 *Storyboard*

Pada Pemain akan mengambil Kunci pada point K, setelah semua musuh kalah dan mendapatkan kunci, pemain baru bisa menuju pintu selanjutnya untuk melanjutkan level dengan menuju point H.

Musuh juga sudah mempunyai kecerdasan, karena sudah diterapkan algoritma A* (A *star*) untuk mencari rute terpendek dari posisinya menuju ke pemain.

3.2.2 Perancangan Struktur Menu

Untuk mempermudah dalam pembuatan sistem, diperlukan perancangan struktur menu program yang akan dibangun. Perancangan struktur menu program ini membantu dalam merancang bagian-bagian dari sistem yang sebenarnya dan untuk mengetahui bagian mana yang nantinya akan diakses terlebih dahulu setelah program selesai dibuat. Perancangan arsitektur menu pada aplikasi permainan *The Tale of Hunter* dapat dilihat pada gambar 3.30



Gambar 3.41 Arsitektur Menu *The Tale of Hunter*

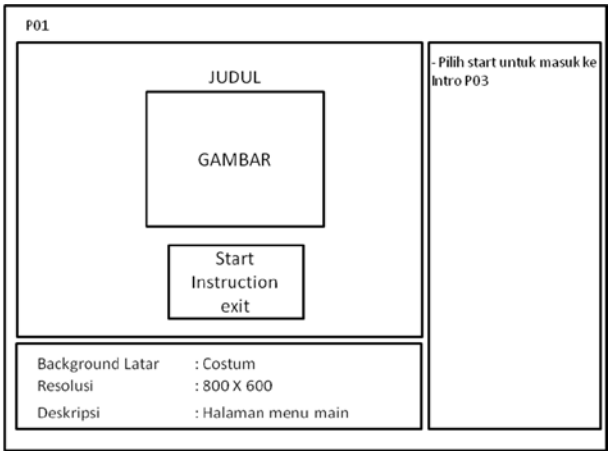
A. Instruksi : merupakan menu untuk menampilkan informasi tentang cara memainkan aplikasi permainan *the tale of hunter*.

B. Mulai permainan : merupakan menu untuk memulai permainan baru.

C. Keluar :merupakan menu untuk keluar dari aplikasi permainan *the tale of hunter*.

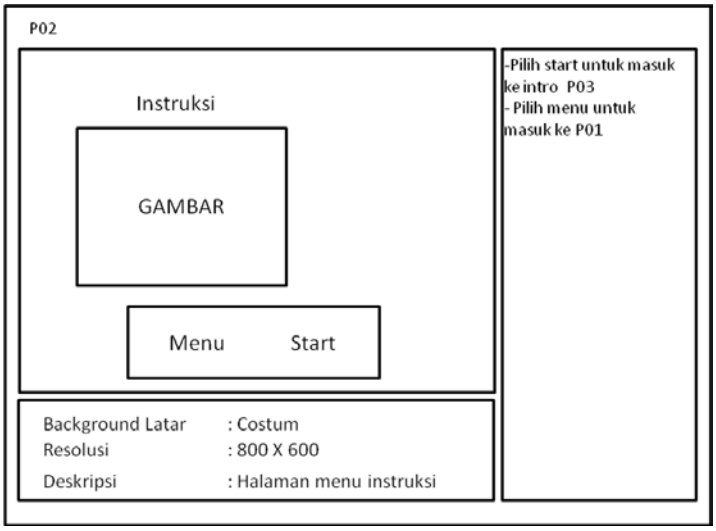
3.2.3 Perancangan Antar Muka

Perancangan antarmuka menu *main* pada aplikasi permainan the tale of hunter dapat dilihat pada gambar 3.42



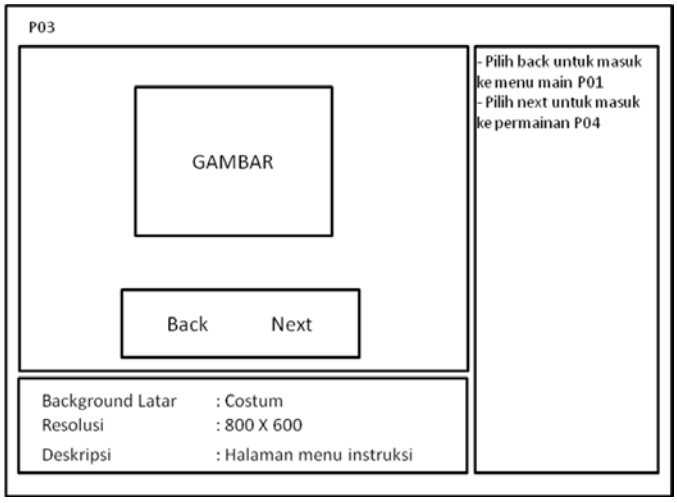
Gambar 3.42 Perancangan Antarmuka Menu *Main The Tale of Hunter*

Perancangan antarmuka instruksi pada aplikasi permainan *the tale of hunter* dapat dilihat pada gambar 3.43.



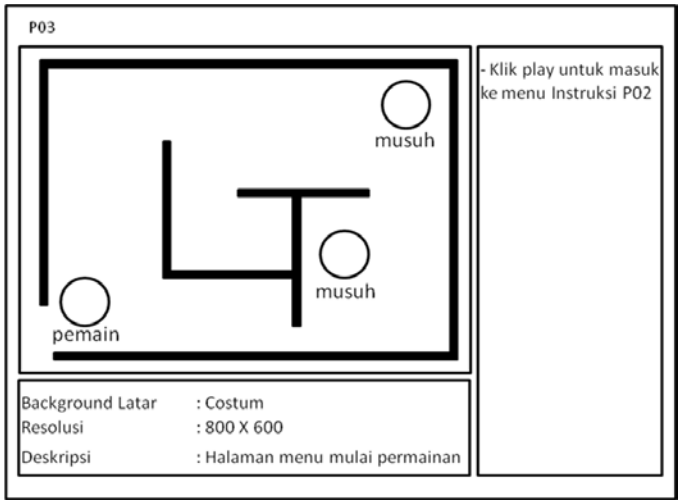
Gambar 3.43 Perancangan Antarmuka Instruksi *The Tale of Hunter*

Perancangan antarmuka intro permainan pada aplikasi permainan *the tale of hunter* dapat dilihat pada gambar 3.44.



Gambar 3.44 Perancangan Antarmuka Intro *The Tale of Hunter*

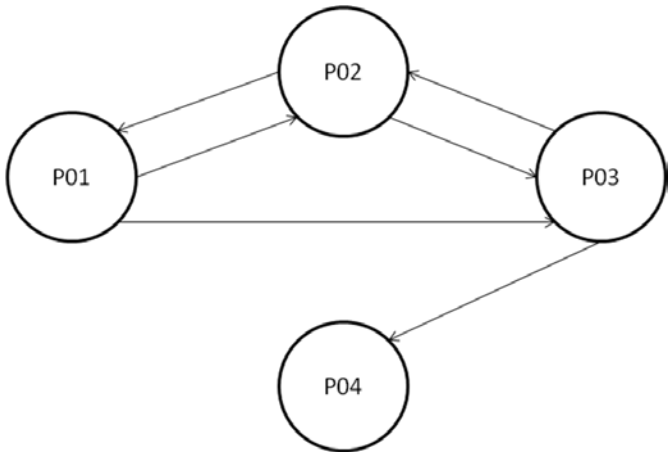
Perancangan antarmuka mulai permainan pada aplikasi permainan *the tale of hunter* dapat dilihat pada gambar 3.45.



Gambar 3.45 Perancangan Antarmuka Memulai Permainan *The Tale of Hunter*

3.2.4 Jaringan Semantik

Jaringan semantik pada aplikasi permainan the tale of hunter dapat dilihat pada gambar 3.36



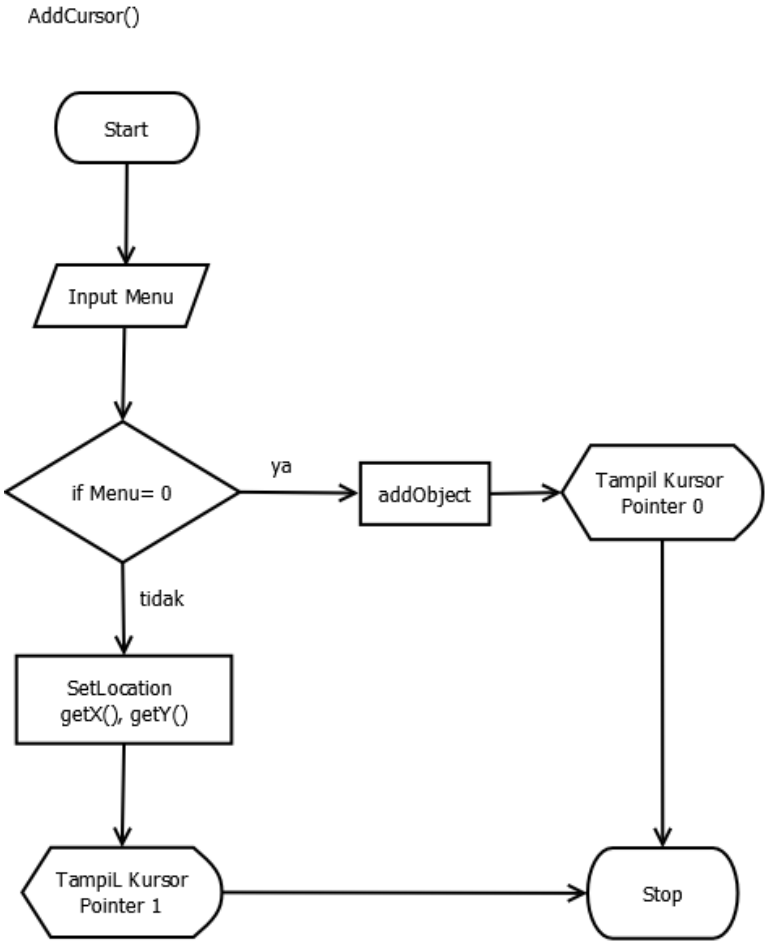
Gambar 3.46 Jaringan Semantik *The Tale of Hunter*

3.2.5 Perancangan Method

Perancangan komponen method merupakan perancangan yang dibuat setelah perancangan arsitektur menu, antarmuka, pesan, dan jaringan semantik. Perancangan ini berfungsi untuk mendeskripsikan method-method yang berada di dalam aplikasi permainan *the tale of hunter*. Adapun method-method yang terdapat dalam aplikasi permainan *The Tale of Hunter* yang akan dibangun adalah sebagai berikut :

A. Method addcursor()

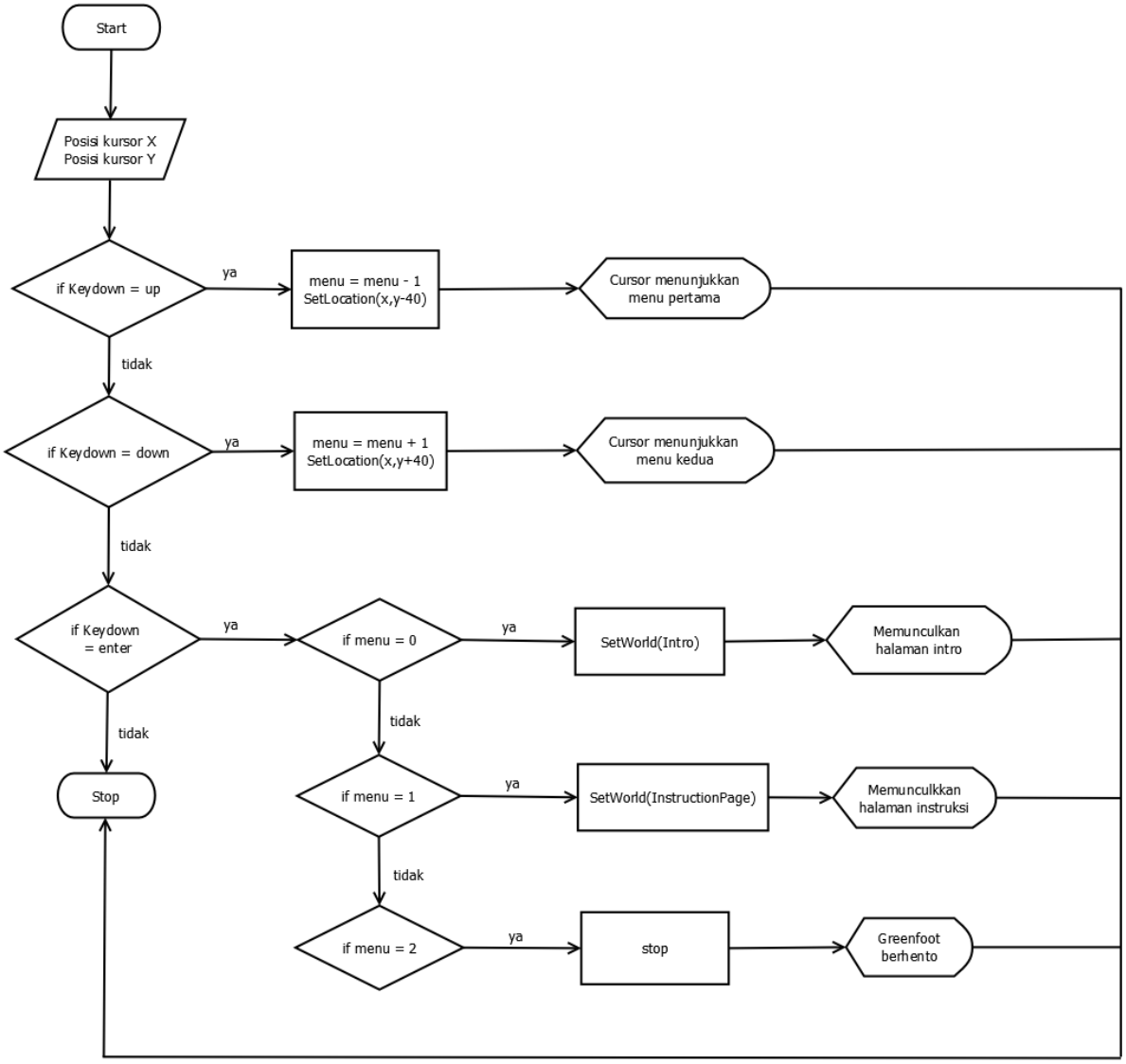
Method addcursor() akan dipanggil ketika memulai permainan



Gambar 3.47 Flowchart Method addcursor()

B. Method Control()

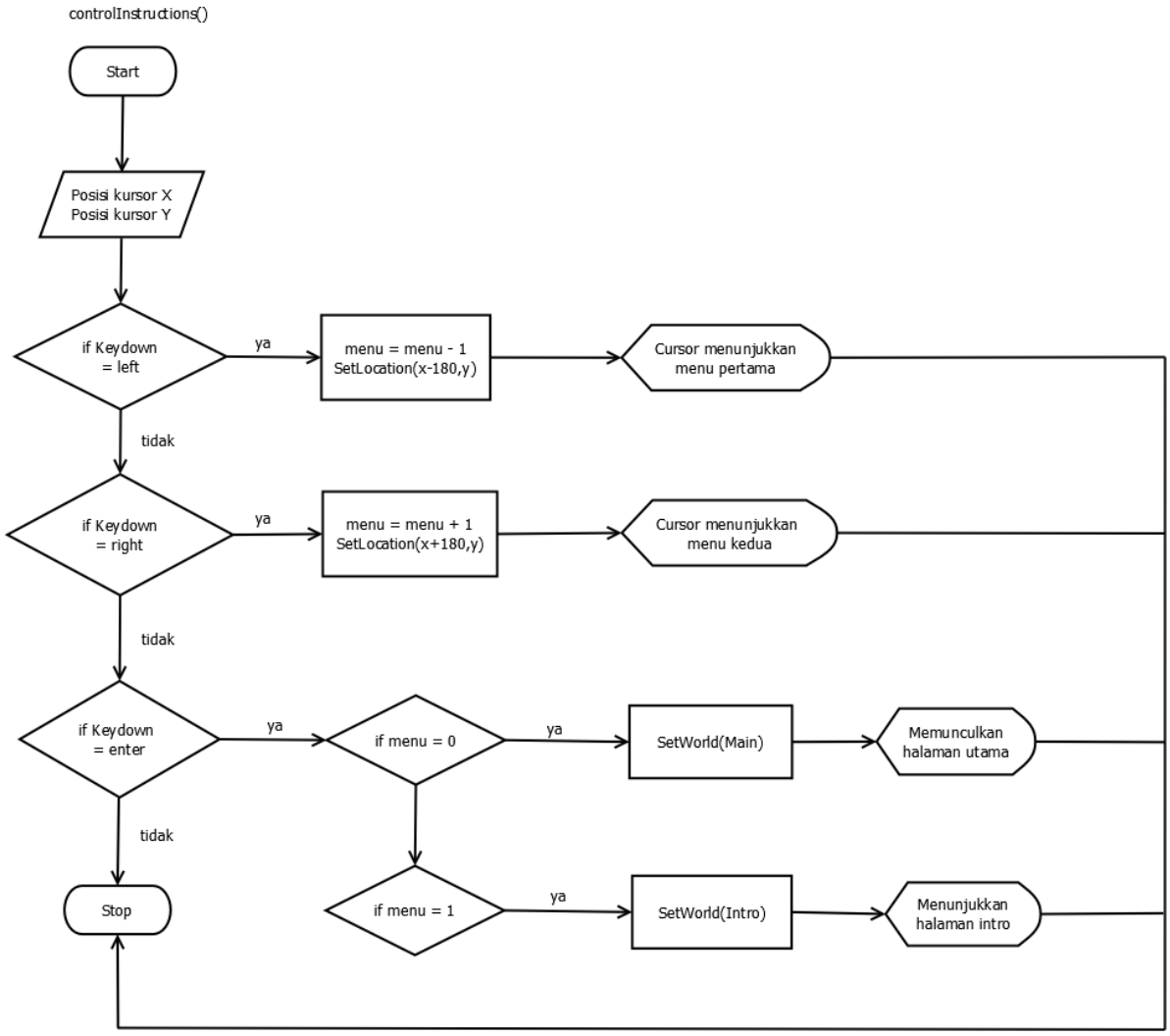
Method Control() dipanggil ketika memilih menu di main



Gambar 3.48 Flowchart Method Control()

C. Method controlInstructions()

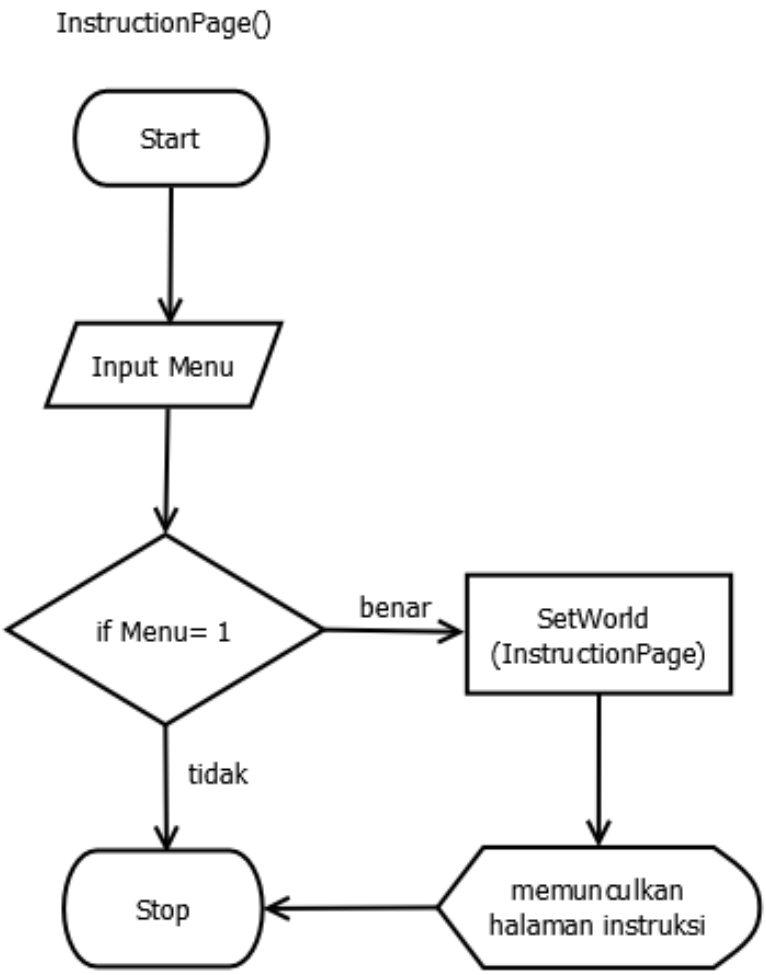
Method Control() dipanggil ketika memilih menu di InstructionPage



Gambar 3.49 Flowchart Method controlInstruction()

D. Method InstructionPage()

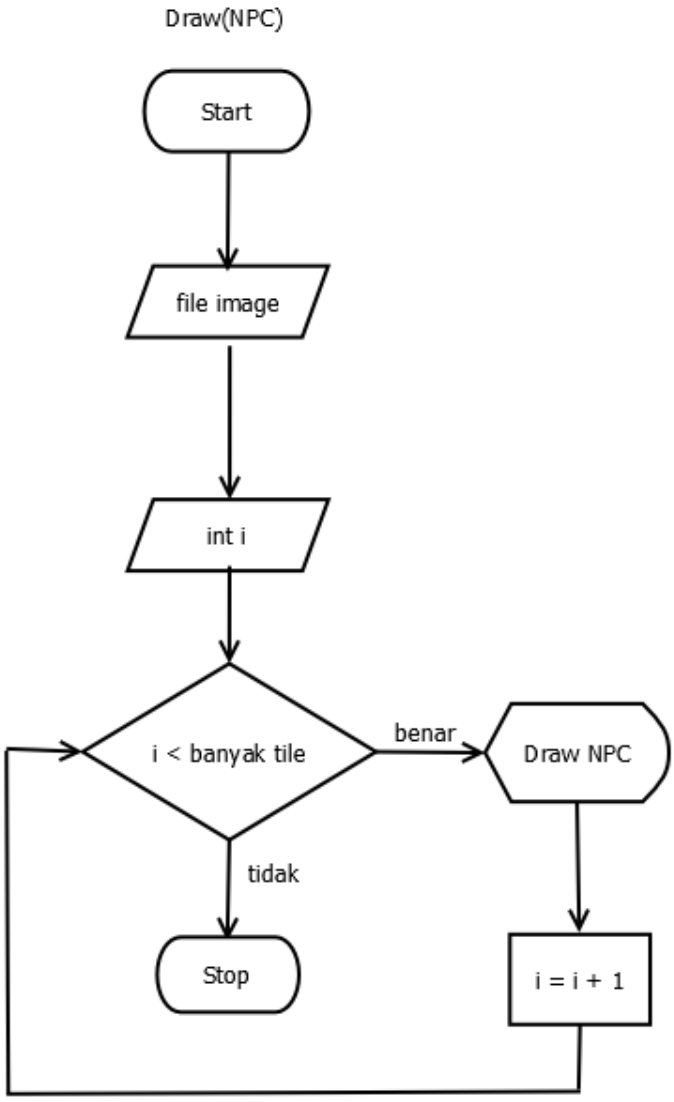
Method IntructionPage() dipanggil ketika memilih menu Instructions



Gambar 3.50 Flowchart Method InstructionPage()

E. Method Draw()

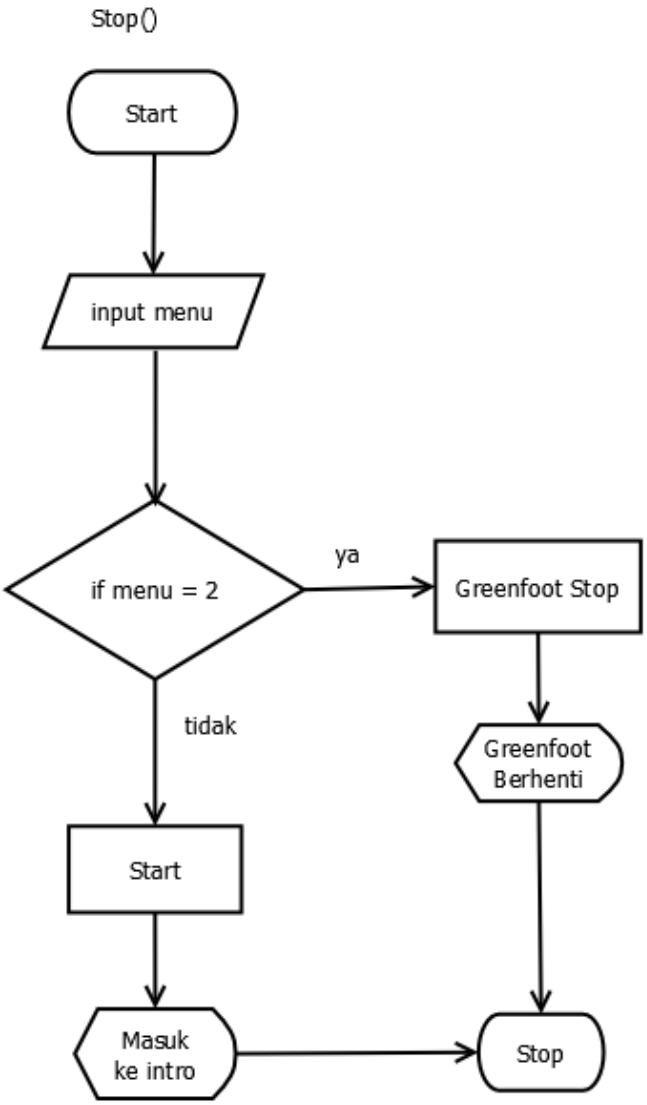
Method Draw() dipanggil ketika pemain sudah memasuki permainan



Gambar 3.51 Flowchart Method Draw()

F. Method stop()

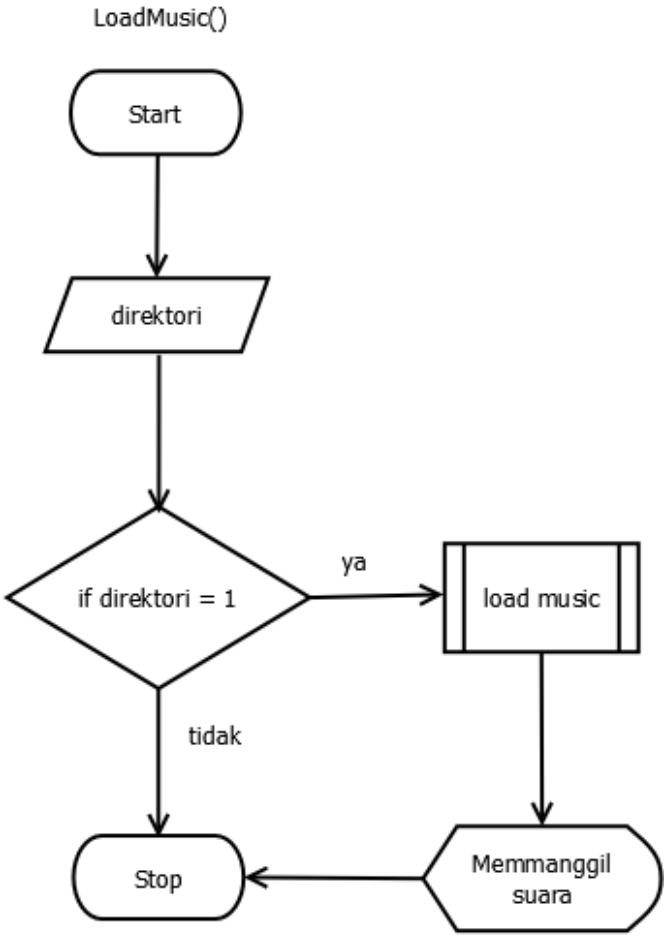
Method stop() akan dipanggil ketika memilih menu exit



Gambar 3.52 Flowchart Method stop()

G. Method LoadMusic()

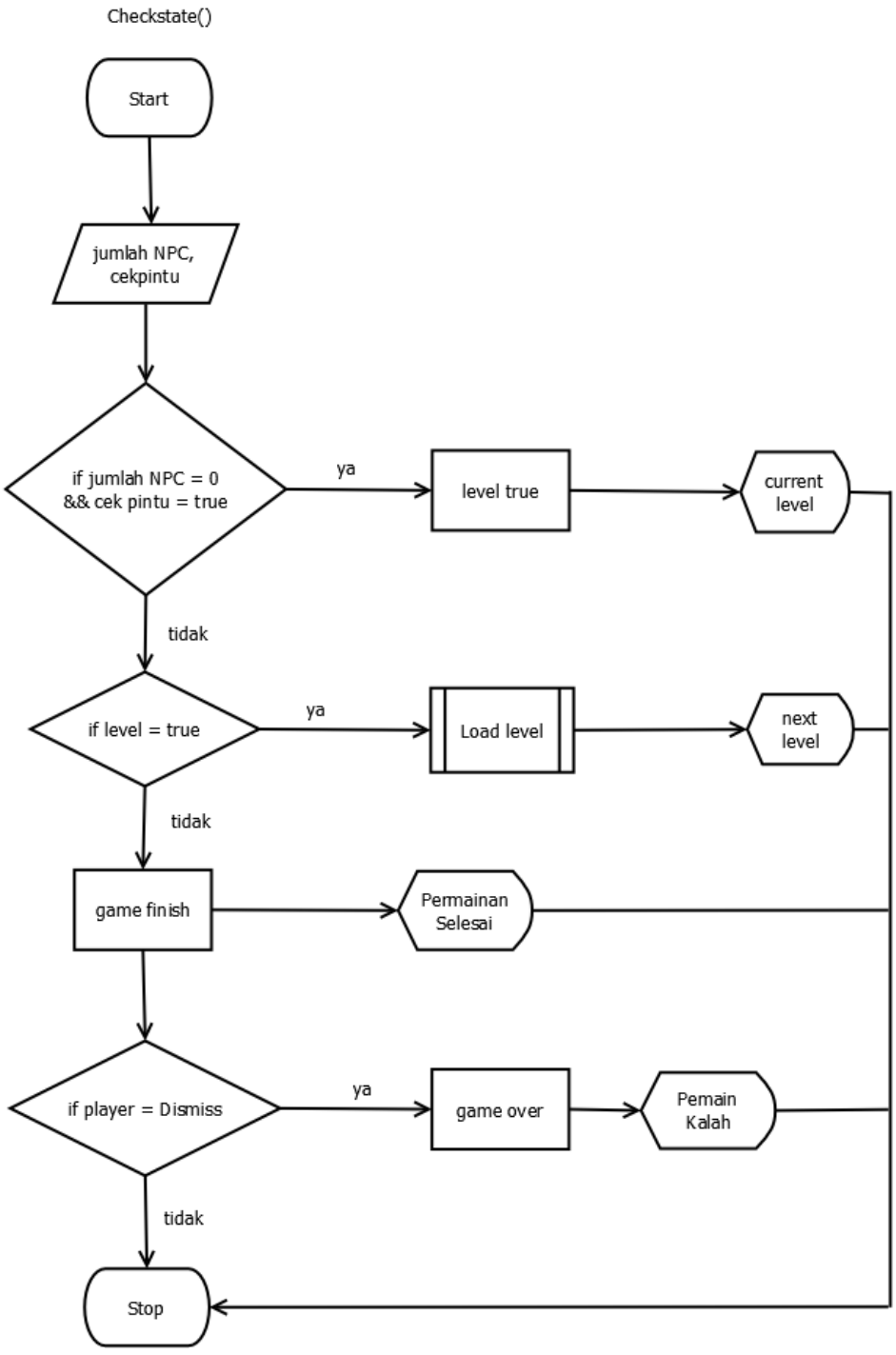
Method LoadMusic() dipanggil ketika sudah memasuki permainan



Gambar 3.52 Flowchart Method LoadMusic()

H. Method CheckState()

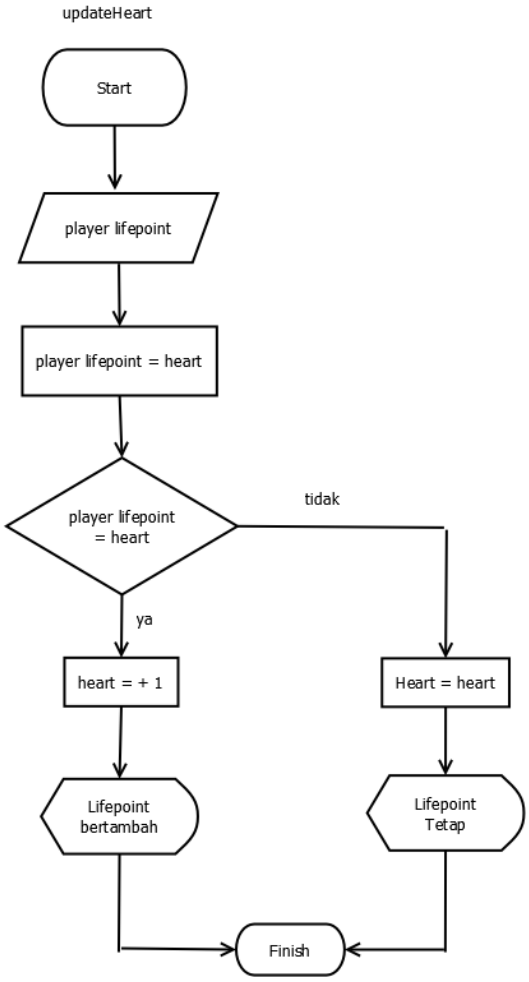
Method CheckState() akan dipanggil ketika sudah memasuki permainan



Gambar 3.53 Flowchart Method Checkstate()

I. Method UpdateHeart()

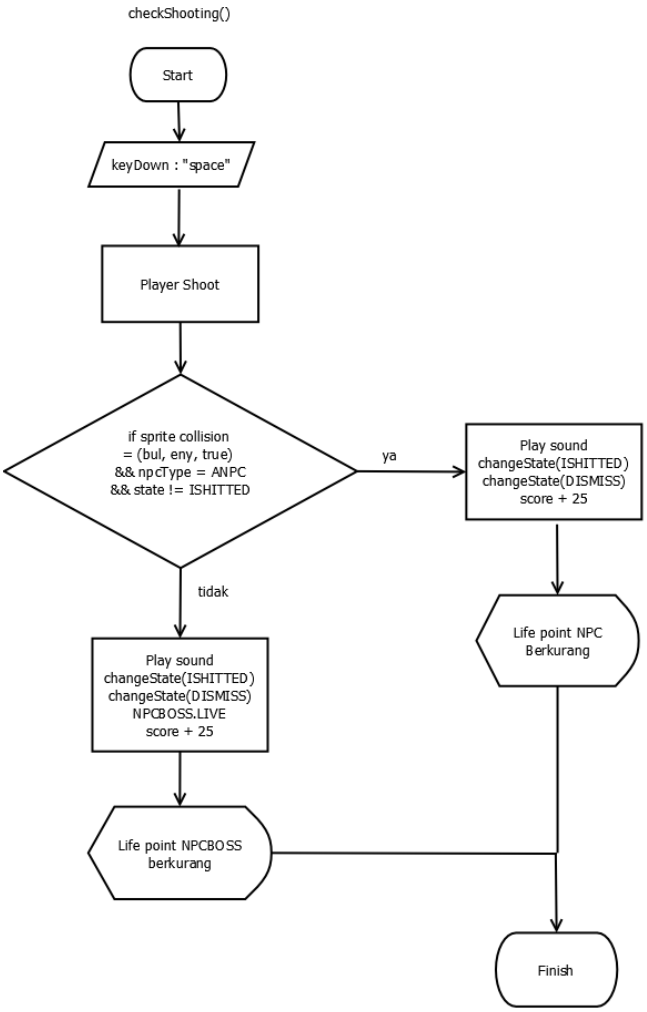
Method UpdateHeart() akan dipanggil ketika sudah memasuki permainan



Gambar 3.54 Flowchart Method UpdateHeart()

J. Method checkShooting()

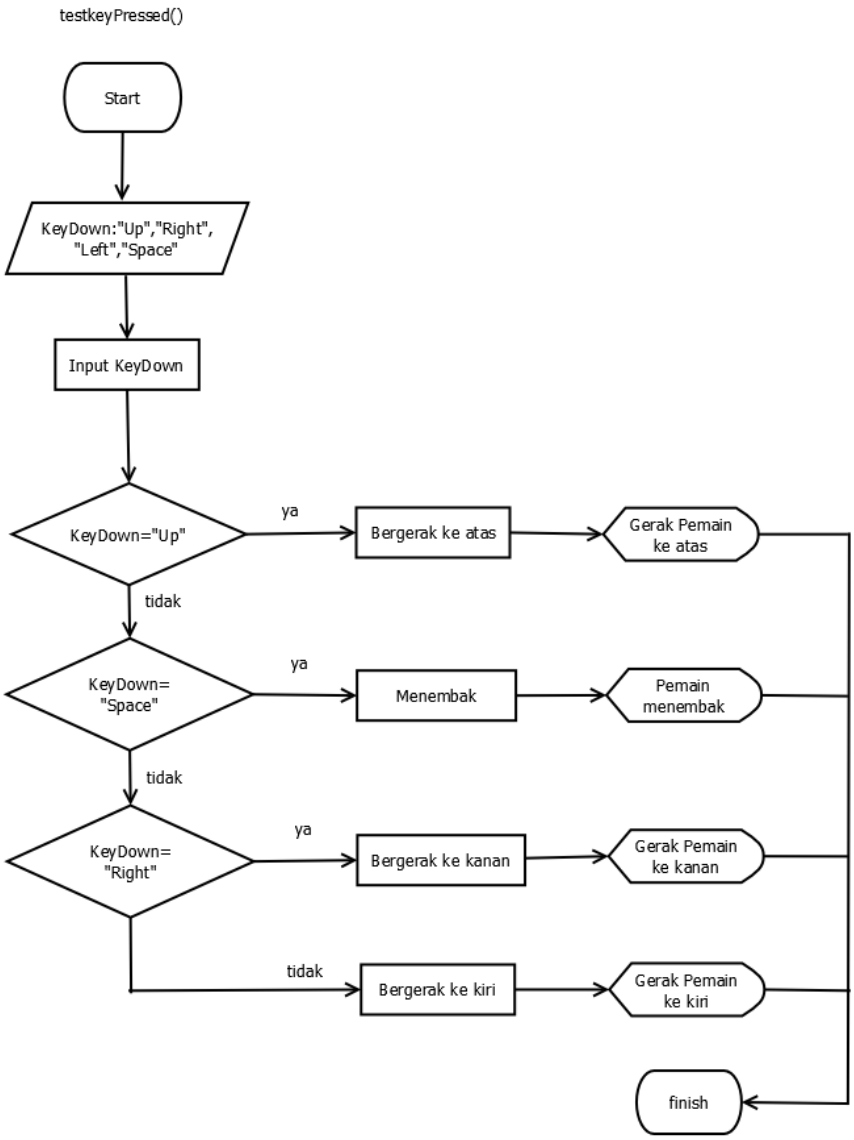
Method checkShooting() akan dipanggil ketika sudah memasuki permainan



Gambar 3.55 Flowchart Method checkShooting()

K. Method testkeyPressed()

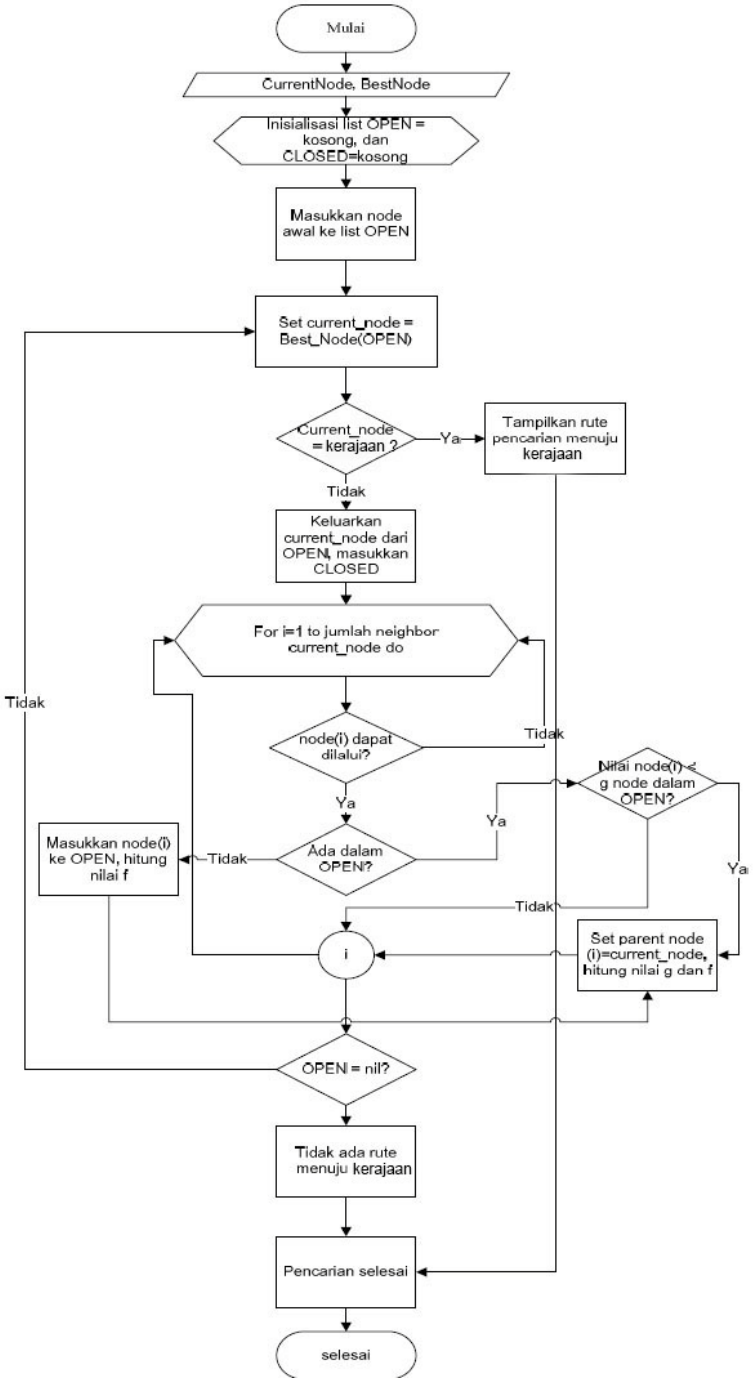
Method testkeyPressed() akan dipanggil ketika sudah memasuki permainan



Gambar 3.56 Flowchart Method testkeyPressed()

L. Method A*

Method A* dipanggil ketika sudah memasuki permainan



Gambar 3.57 Flowchart Method A*