BAB III
ANALISIS DAN PERANCANGAN SISTEM

III.1 Analisis Sistem

Analisis sistem merupakan kegiatan penguraian suatu sistem informasi yang utuh dan nyata ke dalam bagian-bagian atau komponen-komponen komputer yang bertujuan untuk mengidentifikasi serta mengevaluasi masalah-masalah yang muncul, hambatan-hambatan yang mungkin terjadi dan kebutuhan-kebutuhan yang diharapkan sehingga mengarah kepada solusi dengan kebutuhan serta perkembangan teknologi.

III.1.1 Analisis Masalah

Kecerdasan dalam mengatur strategi adalah faktor terpenting dalam *game turn-based strategy* (*TBS*). Kemampuan dalam mengambil satu dari banyak keputusan harus dimiliki oleh AI yang digunakan pada *game TBS*. Ada banyak algoritma yang dapat diterapkan dalam *game* sebagai AI, namun setiap algoritma memiliki kelebihan tersendiri tergantung pada jenis *game*. Tingkat kesulitan yang dimiliki *NPC* juga perlu untuk diperhatikan, karena *NPC* yang terlalu pintar membuat game menjadi sulit untuk dimainkan, begitu pula sebaliknya, *NPC* yang berfikir lambat akan membuat game menjadi terlalu mudah untuk dimainkan

Solusi yang dapat memecahkan masalah tersebut adalah penerapan algoritma minimax dengan Alpha-Beta Pruning pada *game turn-based strategy* (*TBS*) sebagai pemilih keputusan pada AI.

III.1.2 Analisis *Game* Sejenis

Dalam pembahasan, penulis akan membahas *game* dengan jenis *turn based strategy*. *Game* yang akan dibahas dan menjadi acuan disini adalah *game* Gadget Trial.

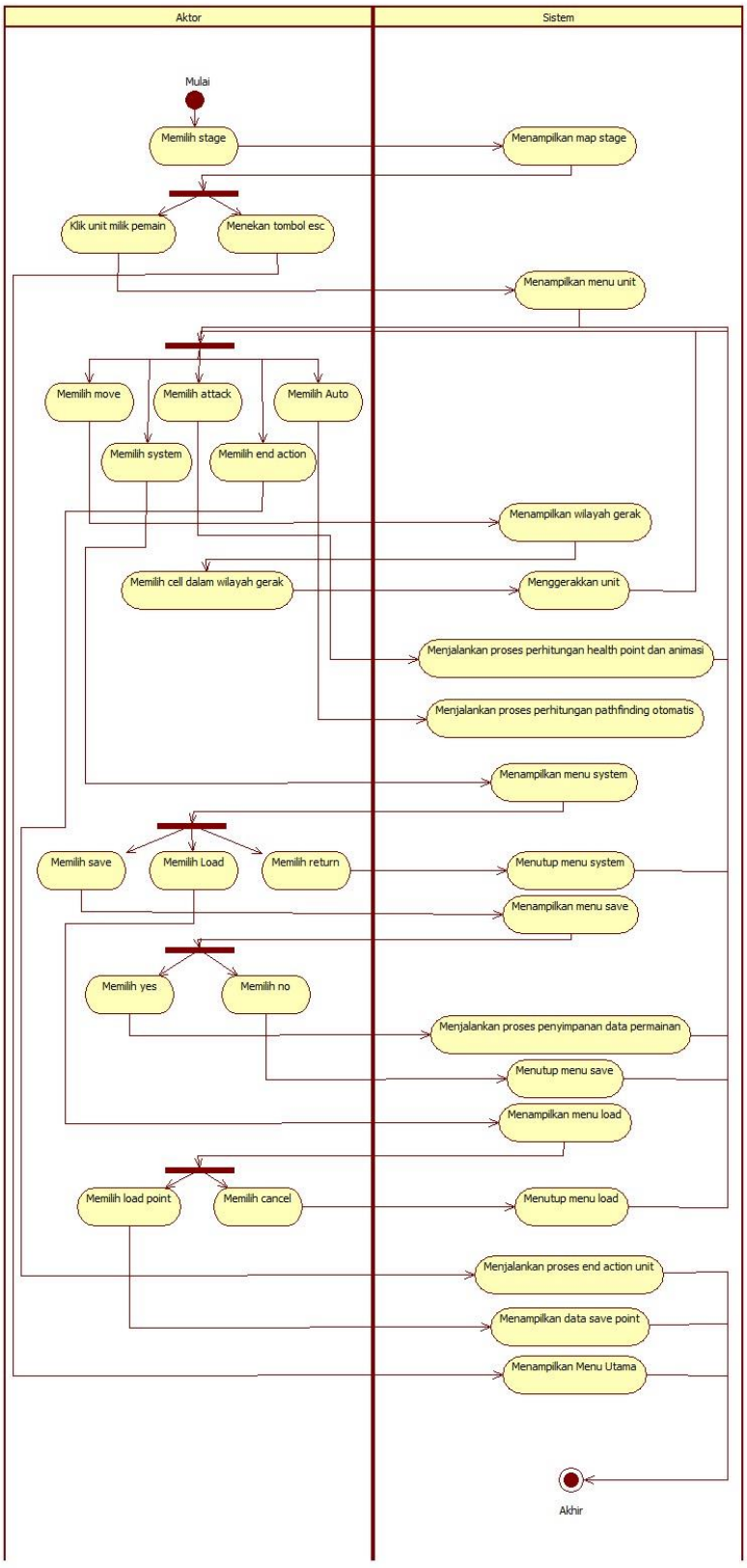
III.1.2.1 Deskripsi *Game* Gadget Trial

Gadget Trial adalah *game* yang mengkombinasikan *game turn-based strategy* dan *visual novel*. Diterbitkan di jepang tanggal 23 Juni 2006 oleh KOGADO Studio's Kuma-san Team. Gadget Trial menggunakan desain *anime* untuk desain karakternya, dan hanya dapat dimainkan oleh 1 orang pemain (*single player*).

Pada *game* Gadget Trial, seorang pemain memiliki sejumlah unit yang memiliki karakteristik yang berbeda, baik dalam jarak serang maupun dalam jumlah langkah yang dapat ditempuh dalam sekali giliran. Pemain harus dapat merebut bangunan utama milik musuh. Dalam hal pergerakan unit, gadget trial tidak mengelompokkannya dalam satu kali giliran, tetapi mengurutkan giliran gerak setiap unit berdasarkan tingkat kecepatan gerak unitnya, sehingga pergerakan unit milik pemain dan unit yang digerakkan oleh AI akan tercampur dan pergantian giliran tidak begitu terasa. Analisis pergerakan unit pada setiap stage dapat dilihat dalam bentuk activity diagram pada Gambar III.2.



Gambar III. 1 Tampilan *game* Gadget Trial



Gambar III. 2 Activity diagram Gadget Trial

III.1.3 Analisis *Game* Yang Dikembangkan

Analisis *game* yang akan dikembangkan merupakan bagian yang mendeskripsikan *game* yang akan dikembangkan. Pada bagian ini terdiri dari pengenalan *game*, *story line*, *gameplay* dan tingkat kesulitan.

III.1.3.1 Pengenalan

Game yang akan dibuat berjudul Rise of Majapahit, dengan menggunakan model *Board games* (Permainan Papan) yang membutuhkan suatu papan atau media yang terbagi dalam sektor-sektor tertentu (dengan garis-garis) dan didalamnya terdapat sejumlah unit yang dapat digerakkan. *Game* ini menggunakan sistem *Turn Based Strategy* dimana pengguna melakukan penyerangan secara bergiliran dengan lawan. Pada dasarnya mirip dengan permainan catur, namun kelebihan *game* ini dibandingkan dengan permainan catur biasa adalah :

1. Setiap unit dapat bergerak bebas dalam ruang lingkup posisi awal unit tersebut, tidak seperti pada permainan catur biasa yang memiliki aturan pergerakan seperti pion yang hanya bisa bergerak maju 1 langkah atau knight/kuda yang hanya bisa berjalan dengan membentuk huruf L.
2. Setiap unit memiliki *Health Point* dimana ketika setiap kali penyerangan oleh musuh, unit yang terkena serangan tidak akan langsung kalah, tetapi setiap besar serangan yang dilakukan penyerang akan mengurangi jumlah *Health Point* milik unit yang diserang.
3. Untuk mendapat kemenangan, setiap unit musuh harus dikalahkan, tidak seperti pada catur biasa yang menggunakan unit raja sebagai kunci kemenangan.
4. Pada awal permainan, unit-unit tidak disusun berdampingan, namun memiliki posisi awal yang tersebar, sehingga setiap *player* dapat memilih strategi yang berbeda, sedangkan pada catur biasa, setiap unit disusun sesuai dengan peraturan, sehingga lawan dapat memprediksi langkah awal dari *player*, begitupun sebaliknya.

III.1.4.2 Storyline

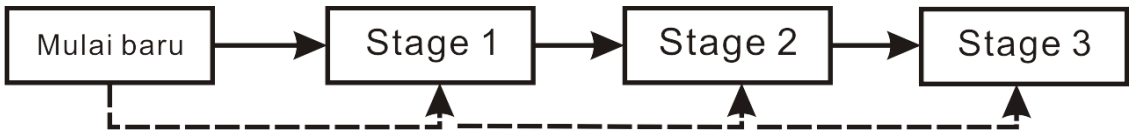
Game Rise of Majapahit ini menceritakan Raden Wijaya sebelum berdirinya Majapahit. Singhasari telah menjadi kerajaan paling kuat di Jawa. Hal ini menjadi perhatian Kubilai Khan, penguasa Dinasti Yuan di Tiongkok. Ia mengirim utusan yang bernama Meng Chi ke Singhasari yang menuntut upeti. Kertanagara, penguasa kerajaan Singhasari yang terakhir menolak untuk membayar upeti dan mempermalukan utusan tersebut dengan merusak wajahnya dan memotong telinganya. Kublai Khan marah dan lalu memberangkatkan ekspedisi besar ke Jawa tahun 1293.

Ketika itu, Jayakatwang, adipati Kediri, sudah menggulingkan dan membunuh Kertanegara. Atas saran Aria Wiraraja, Jayakatwang memberikan pengampunan kepada Raden Wijaya, menantu Kertanegara, yang datang menyerahkan diri. Raden Wijaya kemudian diberi hutan Tarik. Ia membuka hutan itu dan membangun desa baru. Desa itu dinamai Majapahit, yang namanya diambil dari buah maja, dan rasa "pahit" dari buah tersebut. Ketika pasukan Mongol tiba, Wijaya bersekutu dengan pasukan Mongol untuk bertempur melawan Jayakatwang. Setelah berhasil menjatuhkan Jayakatwang, Raden Wijaya berbalik menyerang sekutu Mongol sehingga memaksa mereka menarik pulang kembali pasukannya. Cerita dibagi menjadi 3 *stage*, yang masing-masing menceritakan usaha Raden Wijaya untuk mendapatkan kekuasaan adipati Kediri.

1. Pada *stage* pertama, menceritakan usaha Raden wijaya pada pertama kalidibebaskan Jayakatwang dan diberi huntan tarik, yang kemudian digunakan untuk membangun sebuah desa baru.
2. Pada *stage* kedua, menceritakan persekutuan Raden Wijaya dengan pasukan Mongol utusan Kublai Khan untuk menjatuhkan kekuasaan Jayakatwang.
3. Pada *stage* Ketiga, menceritakan pemberontakan Raden Wijaya pada pasukan Mongol.

III.1.5.3 *Game*play

Dalam aplikasi *game* ini, *player* memiliki tugas untuk menghadapi strategi musuh pada setiap stage, dimana dalam setiap stage, jumlah musuh akan terus bertambah. Bagi *player* pemula, stage yang dapat dimainkan hanya stage 1, dan apabila menyelesaikan stage 1, akan dilanjutkan dengan stage 2, kemudian stage 3. Apabila *player* sudah pernah menyelesaikan stage 1 atau stage 3, maka *player* dapat memainkan stage tersebut kembali tanpa harus melewati stage 1.



Gambar III. 3 Alur Permainan

Ketika memulai permainan pada setiap stage, *player* diberi giliran pertama dalam mengatur strategi, dan menjalankan semua unitnya. Setelah *player* selesai menggerakkan seluruh unitnya , dilanjutkan dengan penggerakan unit musuh oleh AI. Unit dapat menyerang musuh apabila musuh sudah berada di area serang unit. *Player* akan meneruskan permainan apabila telah mengalahkan semua unit musuh, tetapi *player* akan mengalami kekalahan apabila Raden Wijaya (*hero*) mati. Pada setiap stage terdapat peraturan seperti berikut:

1. *Player* memilih *piece* atau unit yang akan digerakkan, diantaranya prajurit, pemanah, penunggang kuda dan hero dimana setiap *piece* memiliki ruang lingkup gerak yang berbeda.
2. Setelah memilih *piece* yang digerakkan, *player* menentukan blok yang menjadi tujuan gerak, blok tujuan bisa ditempati oleh *piece* musuh maupun blok kosong tanpa *piece*.
3. Setelah melakukan pergerakan pada satu *piece*, maka giliran musuh yang menggerakan *piece*, yang kemudian dilanjutkan kembali dengan pergerakan *piece* milik *player*
4. *Player* dinyatakan menang apabila dapat menghabiskan seluruh *piece* milik musuh, dan dinyatakan kalah apabila kehilangan *piece* hero.

III.1.5.4 Tingkat Kesulitan

Game ini memiliki tingkat kesulitan yang berbeda pada setiap *stage*. Tingkat kesulitan terdapat pada keadaan awal yang diberikan pada *player*. Tingkat kesulitan yang akan dihadapi oleh *player* adalah sebagai berikut :

1. Stage 1

   Detail tingkat kesulitan pada stage 1 ditunjukan pada Tabel III.1.

Tabel III. 1 Detail Tingkat Kesulitan Pada Stage 1

	Jumlah	Keterangan
Jumlah unit *player*	6	Terdiri dari 3 prajurit, 2 pemanah dan 1 *hero*
Jumlah unit musuh	6	Terdiri dari 4 prajurit dan 2 pemanah
Jenis *terrain*	2	Terdiri dari hutan dan gunung, hutan bisa dilewati oleh prajurit, pemanah dan *hero* sedangkan gunung hanya dapat dilewati oleh prajurit dan hero

2. Stage 2

   Detail tingkat kesulitan pada stage 2 ditunjukan pada Tabel III.2.

Tabel III. 2 Detail Tingkat Kesulitan Pada Stage 2

	Jumlah	Keterangan
Jumlah unit *player*	10	Terdiri dari 4 prajurit, 2 pemanah, 3 penunggang kuda dan 1 *hero*
Jumlah unit musuh	10	Terdiri dari 4 prajurit, 2 pemanah, 3 penunggang kuda dan 1 *hero*
Jenis *terrain*	2	Terdiri dari hutan dan gunung, hutan bisa dilewati oleh prajurit, pemanah dan *hero* sedangkan gunung hanya dapat dilewati oleh prajurit dan hero

3. Stage 3

Detail tingkat kesulitan pada stage 3 ditunjukan pada Tabel III.3.

Tabel III. 3 Detail Tingkat Kesulitan Pada Stage 3

	Jumlah	Keterangan
Jumlah unit *player*	16	Terdiri dari 7 prajurit, 5 pemanah, 3 penunggang kuda dan 1 *hero*
Jumlah unit musuh	16	Terdiri dari 9 prajurit, 3 pemanah, 3 penunggang kuda dan 1 *hero*
Jenis *terrain*	2	Terdiri dari hutan dan gunung, hutan bisa dilewati oleh prajurit, pemanah dan *hero* sedangkan gunung hanya dapat dilewati oleh prajurit dan hero

III.1.6 Analisis Algoritma

Algoritma ini mulai dikembangkan dari teori *game* zero-sum. Teori ini mendeskripsikan situasi dimana jika terdapat pemain yang mengalami pendapatan, pemain lain akan mengalami kehilangan dengan nilai yang sama dari pendapatan tersebut, dan sebaliknya. Jumlah pendapatan dari pemain yang dikurangi dengan jumlah kehilangan akan berjumlah nol[4].

Teori minimax menyatakan :

Untuk setiap dua orang pemain dalam zero-sum *game*, terdapat nilai V dari strategi yang dimiliki pemain seperti :

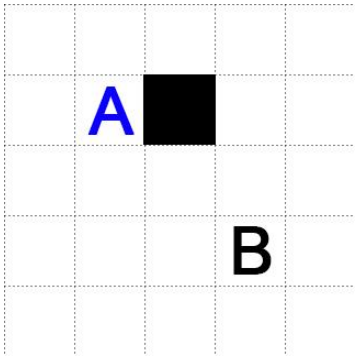
1. Stratregi yang ditentukan pemain kedua akan menghasilkan konsekuensi kemungkinan untuk pemain pertama, V
2. Strategi yang dutentukan pemain pertama akan menghasilkan konsekuensi kemungkinan untuk pemain pertama, -V

Secara setara, strategi pemain pertama akan memastikan suatu nilai V tanpa memperdulikan strategi pemain kedua, dan bersamaan dengan itu pemain kedua akan memastikan dirinya kehilangan nilai sebesar –V. Algoritma Minimax merupakan algoritma dasar pencarian DFS (Depth-First Search) untuk melakukan

traversal dalam pohon. DFS akan mengekspansi simpul paling dalam terlebih dahulu. Setelah simpul akar dibangkitkan, algoritma ini akan membangkitkan simpul pada tingkat kedua, yang akan dilanjutkan pada tingkat ketiga, dst. Dalam melakukan treversal, misalkan dimulai dari suatu simpul i, maka simpul selanjutnya yang akan dikunjungi adalah simpul tetangga j, yang bertetangga dengan simpul k, selanjutnya pencarian dimulai lagi secara rekursif dari simpul j. Ketika telah mencapai simpul m, dimana semua simpul yang bertetangga dengannya telah dikunjungi, pencarian akan dirunut balik ke simpul terakhir yang dikunjungi sebelumnya dan mempunyai simpul yang belum dikunjungi. Selanjutnya pencarian dimulai kembali dari j. Ketika tidak ada lagi simpul yang belum dikunjungi yang dapat dicapai dari simpul yang telah dikunjungi maka pencarian selesai.

III.1.6.1 Kasus Algoritma Minimax

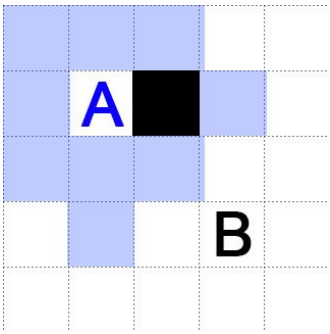
Diumpamakan terdapat satu *piece* milik player ditunjukkan oleh simpul B dan satu *piece* milik *NPC* ditunjukkan oleh simpul A yang terhalangi oleh kotak berwarna hitam, dimana kotak tersebut tidak dapat dilewati oleh seluruh *piece* seperti ditunjukkan pada Gambar III.2.



Gambar III. 4 Posisi Simpul A dan Simpul B

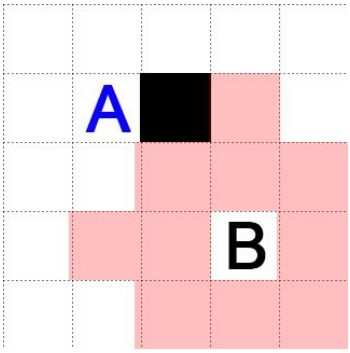
Langkah-langkah algoritma minimax pada *NPC* adalah sebagai berikut : Masukkan *starting node* (simpul A) ke *Open list* karena pada *Open list* hanya terdapat satu simpul (simpul A).

Bangkitkan seluruh wilayah gerak yang dimiliki oleh simpul A. wilayah gerak yang dimiliki oleh simpul A ditunjukkan oleh warna biru seperti ditunjukkan pada Gambar III.5, kemudian hitung bobot yang dimiliki oleh setiap wilayah gerak.



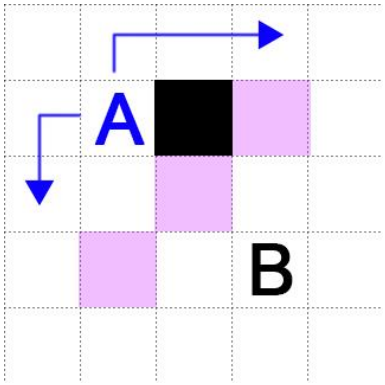
Gambar III. 5 Wilayah Pergerakan Simpul A

Bangkitkan seluruh wilayah gerak yang dimiliki oleh simpul B. wilayah gerak yang dimiliki oleh simpul B ditunjukkan oleh warna merah seperti ditunjukkan pada Gambar III.6, kemudian hitung bobot yang dimiliki oleh setiap wilayah gerak.



Gambar III. 6 Wilayah Pergerakan Simpul B

Asumsikan simpul A hanya memiliki 2 pola pergerakan. Pola pergerakan yang lain tidak disertakan dikarenakan algoritma mendapatkan kesamaan wilayah pergerakan dengan simpul B dimana wilayah tersebut dapat menjadi wilayah yang dapat diserang terlebih dahulu oleh simpul B seperti ditunjukkan pada Gambar III.7.



Gambar III. 7 Dua Pola Pergerakan Simpul A

Setiap pola pergerakan dievaluasi kembali menggunakan langkah 2 dan tiga hingga membentuk sebuah pohon kemungkinan. Pola-pola pergerakan tersebut dihitung hingga menghasilkan nilai maksimum dan minimum. Berikut adalah perhitungan-perhitungan dari beberapa kondisi.

1. jumlah tile gerak

   h1 = tile – tileObjWall

   keterangan :

   h1 = jumlah tile yang dapat dilalui pada area gerak

   tile = jumlah keseluruhan tile dalam area gerak

   tileObjWall = jumlah tile yang memiliki *obstacle* berupa gunung atau hutan

2. jumlah musuh

   h2 = jumlah musuh

   keterangan :

   h2 = jumlah unit musuh yang berada dalam area gerak

3. jarak terdekat

   h3 = (abs(finish.x - start.x) + abs(finish.y - start.y))/GRID_SIDE

   keterangan :

   h3 = jarak unit musuh terdekat

   start.x = nilai koordinat horizontal untuk gerak awal

   start.y = nilai koordinat vertikal untuk gerak awal

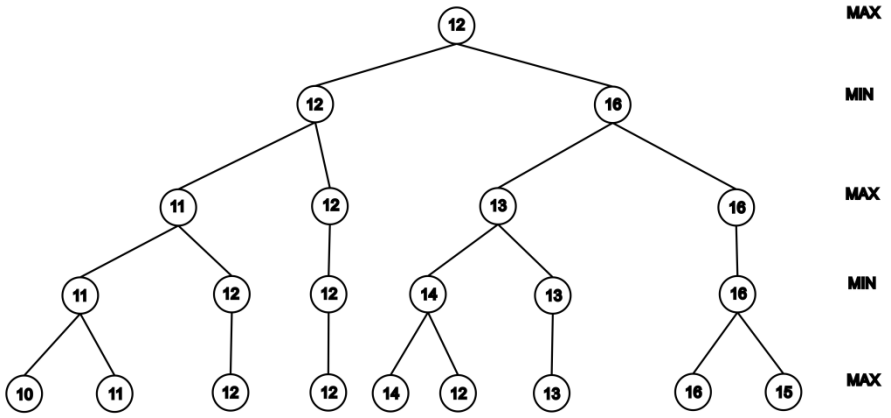
   finish.x = nilai koordinat horizontal untuk unit musuh terdekat

finish.y = nilai koordinat vertikal untuk unit musuh terdekat

dari ketiga nilai heuristik dapat dihasilkan nilai untuk setiap simpul dengan perhitungan berikut.

$$h = h1 + h2 + h3$$

Kemudian didapat nilai minimum dan maksimum pada setiap anak simpul hingga didapatkan jalur dari anak simpul terbawah menuju simpul teratas seperti ditunjukkan pada gambar III.8.



Gambar III. 8 Pohon Keputusan

Membandingkan nilai heuristik pada minimax dilakukan dengan mengasumsikan *Max* sebagai langkah yang diambil oleh *NPC* sedangkan *Min* diasumsikan sebagai langkah yang diambil oleh pemain. Penelusurannya akan dilakukan secara DFS yaitu dimulai dari atas ke kiri kemudian menuju ke simpul yang lebih dalam hingga ke akar yang kemudian dibandingkan nilai heuristik setiap simpul yang sejajar kemudian naik kembali ke simpul di atasnya kemudian penelusuran diulang kembali pada simpul sebelah kanan hingga mendapatkan solusi.

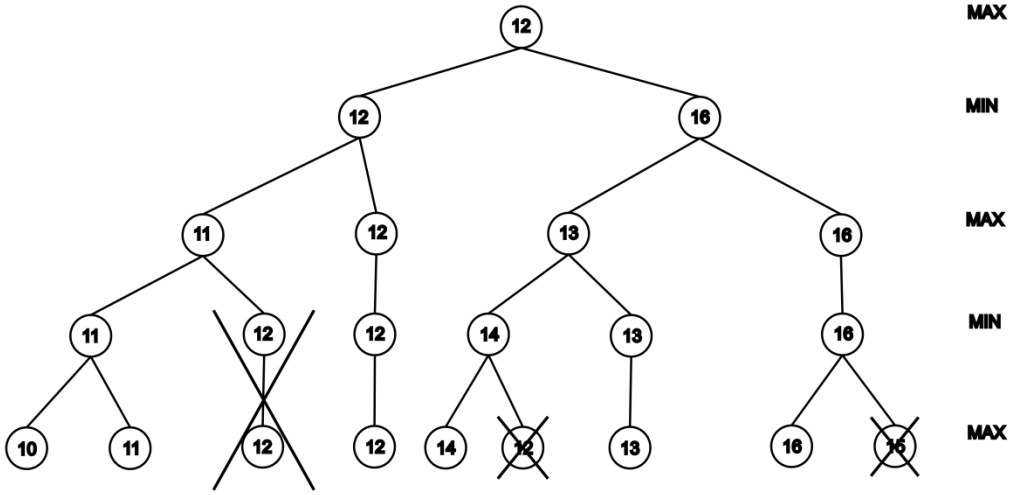
Penelusuran diawali pada level 0 atau max, kemudian ditelusuri ke simpul dibawahnya yang paling kiri hingga mencapai akar simpul paling kiri yang memiliki nilai heuristik yaitu h=10. Nilai heuristik simpul pertama akan disimpan pada *parent*-nya satu level diatasnya yaitu level 3 sebagai nilai heuristik sementara. Kemudian nilai heuristik *parent* yaitu h=10 dibandingkan dengan nilai heuristik simpul kedua yaitu h=11. Karena level 4 bernilai *max*, nilai simpul

kedua yaitu h=11 akan menggantikan nilai *parent* yang bernilai h=10 karena *max* selalu menghasilkan nilai terbesar.

Kemudian nilai simpul pada level 3 dibandingkan kembali dimana nilai pada simpul pertama pada level 3 bernilai h=11 dan simpul kedua. Sebelumnya akan ditelusuri terlebih dahulu anak dari simpul kedua, dilakukan perbandingan seperti pada simpul pertama hingga didapatkan nilai untuk simpul kedua yaitu h=12 Pada level 3 nilai yang dihasilkan adalah nilai *min* maka dicari nilai terkecil, yaitu nilai simpul pertama yaitu h=11. Nilai simpul pertama akan diambil sebagai nilai *parent* sementara pada satu level diatasnya yaitu level 2. Pada level 2 dilakukan evaluasi perbandingan untuk mencari nilai *max.* dimana simpul pertama memiliki nilai h=11 dan untuk simpul kedua dilakukan penelusuran pada anak simpulnya hingga menghasilkan nilai h=12. Nilai dari simpul kedua yaitu h=12 diambil sebagai nilai *parent,* karena memiliki nilai lebih besar dari simpul pertama yaitu h=11. Penelusuran berlanjut pada level 1 dimana nilai simpul pertama yaitu h=12 dibandingkan dengan nilai simpul kedua. Nilai simpul kedua pada level 1 didapat dari hasil penelusuran dan pembandingan terlebih dahulu pada anak-anak simpul kedua, hingga dihasilkan nilai simpul kedua pada level 1 adalah h=16. Pada level 1 menghasilkan nilai *min* sehingga nilai simpul pertama yaitu 12 akan menjadi nilai parent, karena nilai simpul pertama lebih kecil dari nilai simpul kedua yaitu h=16. Penelusuran berhenti ketika simpul telah mencapai simpul teratas, sehingga nilai h=12 pada level 0 menjadi nilai solusi untuk menentukan langkah terbaik pada *NPC*.

Dan apabila menggunakan optimasi alpha-beta pruning, nilai heuristik, pertama penelusuran akan dimulai dari simpul paling kiri yaitu h=10, nilai paling kiri akan menjadi nilai *parent* sementara, dan kemudian simpul tetangganya yaitu simpul kedua akan dibandingkan dengan *parent* (β) sementara yang memiliki nilai h=10, apabila nilai tersebut lebih besar atau sama dengan beta maka simpul berikutnya akan dipruning, apabila tidak maka akan dibandingkan dengan alpha, dan apabila nilai simpul lebih besar dari alpha maka nilai alpha akan diganti dengan nilai simpul tersebut, nilai simpul kedua lebih besar dari simpul *parent* sementara, sehingga nilai simpul kedua yaitu h=11 akan menggantikan nilai

parent sementara yaitu h=10 dan simbul tetangganya yaitu h=12 tidak akan ditelusuri kembali. begitu seterusnya hingga didapat sebuah solusi. Perbedaan dalam langkah penelusurannya ditunjukkan oleh gambar III.9.



Gambar III. 9 Phohon Keputusan Minimax Menggunakan Alpha-beta Pruning

III.1.7 Analisis Kebutuhan Non-fungsional

Analisis kebutuhan non fungsional dilakukan untuk mengetahui spesifikasi kebutuhan untuk sistem. Spesifikasi kebutuhan melibatkan analisis perangkat keras / hardware, analisis perangkat lunak / *software*.

III.1.7.1 Analisis Kebutuhan Perangkat Lunak

Perangkat lunak digunakan dalam sebuah sistem merupakan perintah perintah yang diberikan kepada perangkat keras agar bisa saling berinteraksi diantara keduanya. Perangkat lunak yang digunakan dalam pembangunan aplikasi *game* ini adalah sebagai berikut :

1. Sistem Operasi Window 7 64 bit
2. *Game Maker*

III.1.7.2 Analisis Kebutuhan Perangkat Keras

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

Pada aplikasi ini, perangkat keras yang digunakan adalah sebagai berikut :

1. Intel® Core™ i3-2310M CPU @ 2.10GHz
2. RAM 6 GB
3. VGA 2 GB
4. Harddisk 600 GB
5. Monitor
6. Mouse dan Keyboard

III.1.7.3 Analisis Kebutuhan Perangkat Pikir

Analisis kebutuhan perangkat piker dilakukan untuk mengetahui kemampuan pengguna (*player*) yang dapat memaikan *game* ini. Berikut adalah analisis kebutuhan perangkat pikir pada *game* ini :

Tabel III. 4 Tabel Kebutuhan Perangkat Pikir

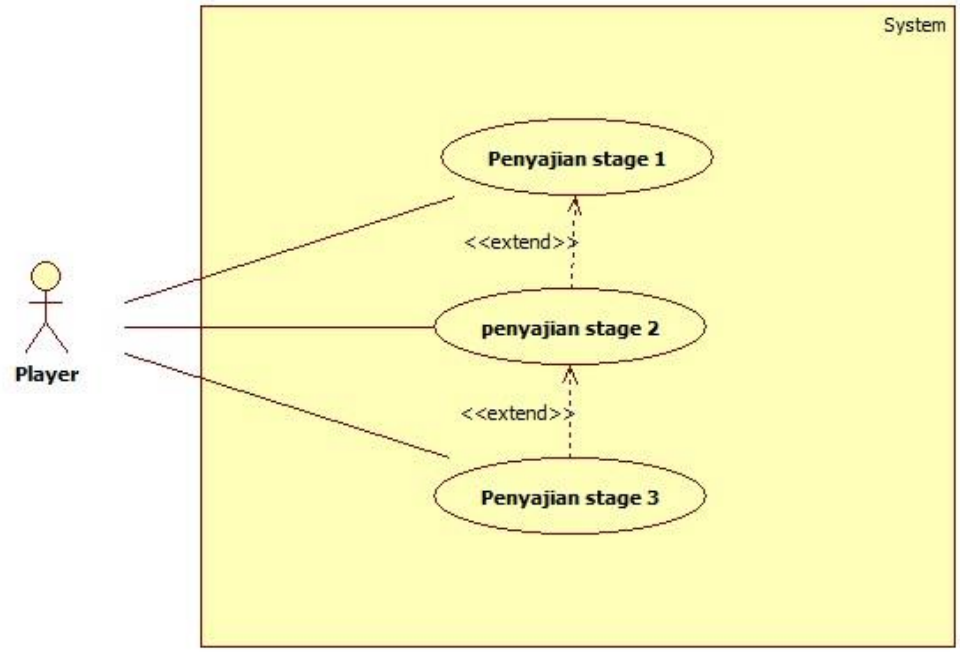
Karakteristik Pengguna	Keterangan
Computer Literacy	Kemampuan menggunakan komputer tingkat pemula
Application Experience	Tidak dibutuhkan
Education	Tidak dibutuhkan
Reading Level	Bahasa Indonesia dan Bahasa Inggris tingkat pemula
Typing Skill	Tidak dibutuhkan
Age	12 tahun keatas [7]
Gender	Pria dan Wanita
Disabilities	Buta total dan buta warna

III.1.8 Analisis Kebutuhan Fungsional

Analisis kebutuhan fungsional dilakukan untuk mengetahui gambaran sistem yang akan dibangun. Analisis kebutuhan fungsional terdiri dari pendefinisan fungsi *game* dengan menggunakan use case diagram, definisi aktor pada use case, definisi use case dan skenario use case.

1. Use Case Diagram

   Berikut adalah penggambaran fungsi *game* berdasarkan interaksi antar aktor dan objek pada sistem dengan menggunakan diagram use case.



Gambar III. 10 Use Case Diagram

2. Definisi Aktor

   Definisi aktor berfungsi menjelaskan peran setiap aktor yang terdapat pada use case.

Tabel III. 5 Definisi Aktor

No.	Aktor	Deskripsi
1.	*Player*	User / pengguna yang memainkan *game* ini

3. Definisi Use Case

Definisi Use Case berfungsi untuk menjelaskan proses yang terdapat pada setiap Use Case.

Tabel III. 6 Definisi Use Case

No.	Use Case	Deskripsi
1	Penyajian *Stage* 1	Proses permainan pada *stage* 1
2	Penyajian *Stage* 2	Proses permainan pada *stage* 2
3	Penyajian *Stage* 3	Proses permainan pada *stage* 3

4. Skenario Use Case

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

Tabel III. 7 Skenario Use Case Penyajian *Stage* 1

Identifikasi	
Nomor	1
Nama	Penyajian *Stage* 1
Tujuan	Memainkan *stage* 1
Deskripsi	Proses permainan pada *stage* 1
Aktor	*Player*
Skenario Utama	
Kondisi Awal	*Aktor* berada di dalam permainan pada *stage* 1

Aksi Aktor	Reaksi Sistem
1. Memilih *stage* 1	2. Menampilkan *room stage* 1
3. Memilih *pathfinder* milik aktor	4. Menampilkan wilayah gerak dan wilayah serang
5. Menekan tombol *end turn*	6. Menjalankan proser *end turn*
7. Menekan tombol keluar	8. Menampilkan pesan keluar dari *game*
9. Memilih *cell* dalam wilayah gerak	10. Menggerakkan *pathfinder*
11. Memilih *pathfinder* musuh dalam wilayah serang	12. Menjalankan proses perhitungan
Skenario Tambahan	
Kondisi Awal	Aktor berada di pesan keluar dari *game*
1. Menekan tombol Ya	2. Menampilkan menu utama
3. Menekan tombol Tidak	4. Menampilkan *room stage* 1

Tabel III. 8 Skenario Use Case Penyajian *Stage* 2

Identifikasi	
Nomor	2
Nama	Penyajian *Stage* 2
Tujuan	Memainkan *stage* 2
Deskripsi	Proses permainan pada *stage* 2
Aktor	*Player*
Skenario Utama	
Kondisi Awal	*Aktor* berada di dalam permainan pada *stage* 2
Aksi Aktor	Reaksi Sistem
1. Memilih *stage* 2	2. Menampilkan *room stage* 2

3. Memilih *pathfinder* milik aktor	4. Menampilkan wilayah gerak dan wilayah serang
5. Menekan tombol *end turn*	6. Menjalankan proser *end turn*
7. Menekan tombol keluar	8. Menampilkan pesan keluar dari *game*
9. Memilih *cell* dalam wilayah gerak	10. Menggerakkan *pathfinder*
11. Memilih *pathfinder* musuh dalam wilayah serang	12. Menjalankan proses perhitungan
Skenario Tambahan	
Kondisi Awal	Aktor berada di pesan keluar dari *game*
5. Menekan tombol Ya	6. Menampilkan menu utama
7. Menekan tombol Tidak	8. Menampilkan *room stage* 2

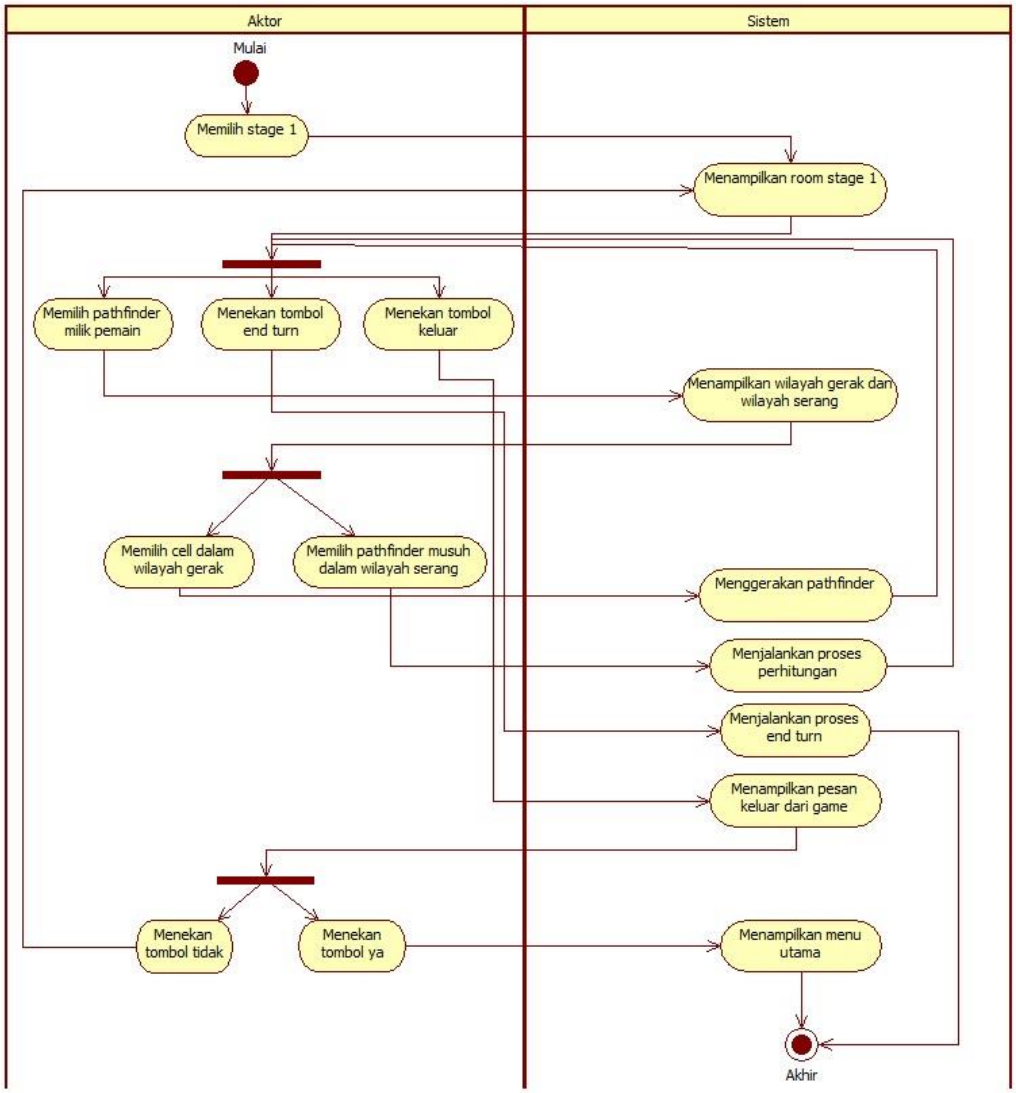
Tabel III. 9 Skenario Use Case Pilihan Mulai Baru

Identifikasi	
Nomor	3
Nama	Penyajian *Stage* 3
Tujuan	Memainkan *stage* 3
Deskripsi	Proses permainan pada *stage* 3
Aktor	*Player*
Skenario Utama	
Kondisi Awal	*Aktor* berada di dalam permainan pada *stage* 3
Aksi Aktor	Reaksi Sistem
1. Memilih *stage* 3	2. Menampilkan *room stage* 3
3. Memilih *pathfinder* milik aktor	4. Menampilkan wilayah gerak dan wilayah serang
5. Menekan tombol *end turn*	6. Menjalankan proser *end turn*
7. Menekan tombol keluar	8. Menampilkan pesan keluar dari

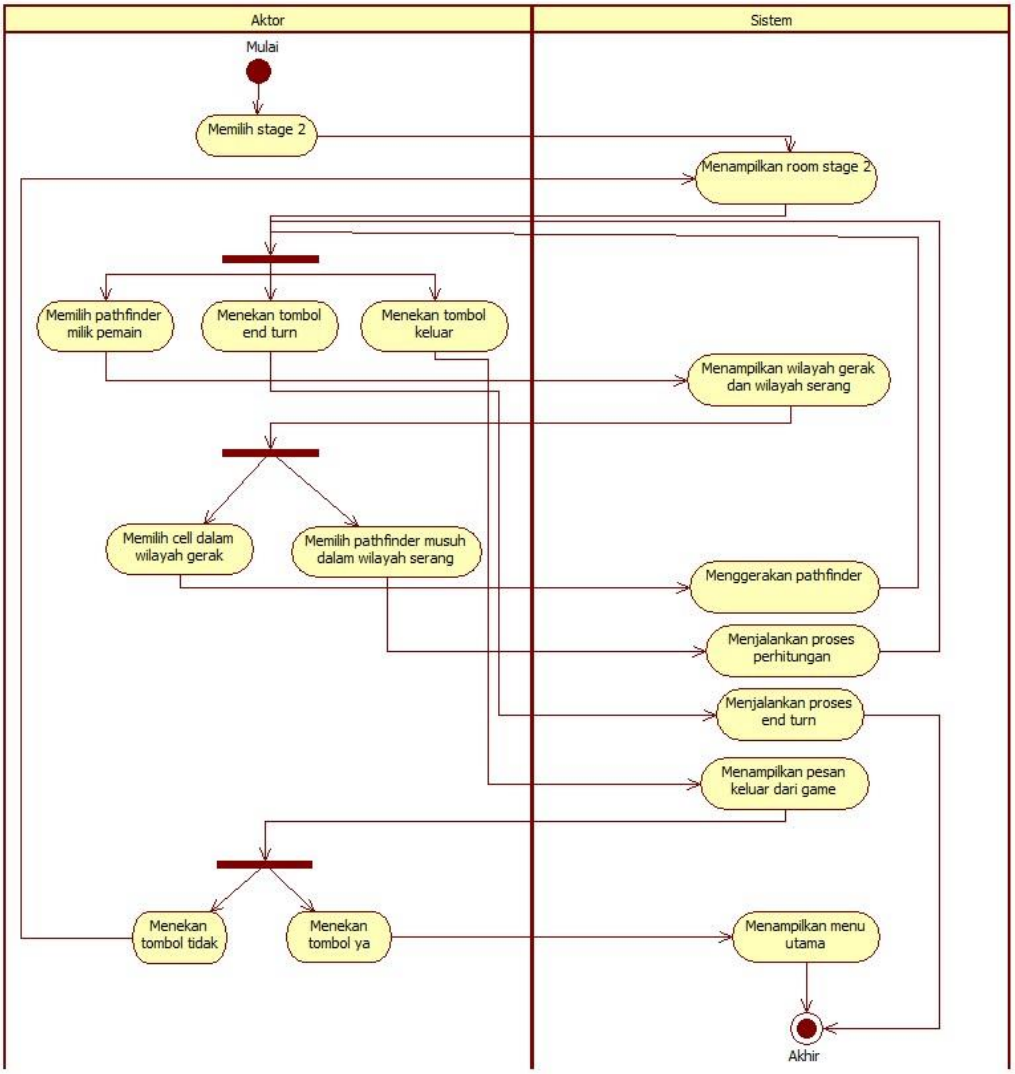
	game
9. Memilih *cell* dalam wilayah gerak	10. Menggerakkan *pathfinder*
11. Memilih *pathfinder* musuh dalam wilayah serang	12. Menjalankan proses perhitungan
Skenario Tambahan	
Kondisi Awal	Aktor berada di pesan keluar dari *game*
9. Menekan tombol Ya	10. Menampilkan menu utama
11. Menekan tombol Tidak	12. Menampilkan *room stage* 3

5. Activity Diagram

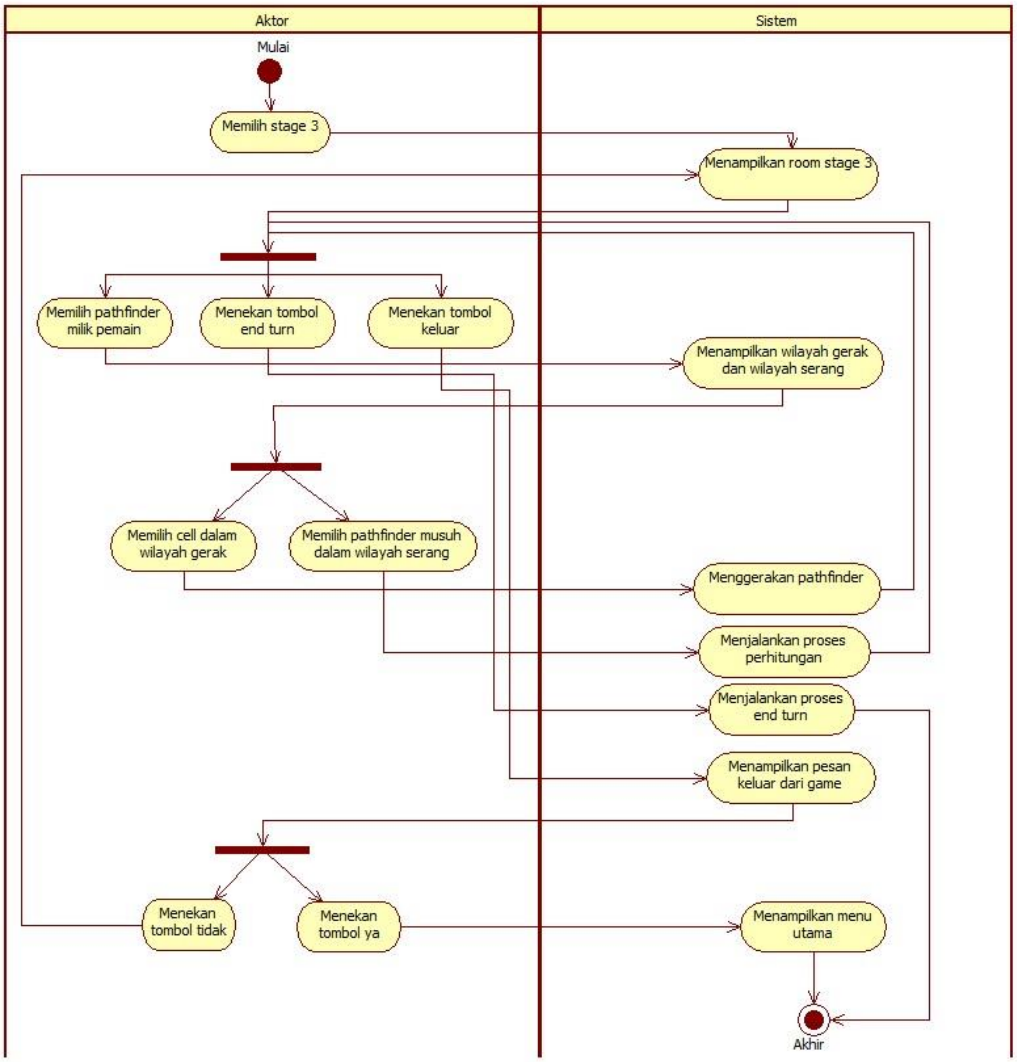
Activity diagram menggambarkan berbagai alir aktivitas dalam sistem yang sedang dirancang, bagaimana masing-masing alir berawal, *decision* yang mungkin terjadi, dan bagaimana mereka berakhir. *Activity diagram* juga dapat menggambarkan proses pararel yang mungkin terjadi pada eksekusi. *Activity diagram* merupakan *state diagram* khusus, dimana sebagian besar *state* adalah *action* dan sebagian besar transisi di-*trigger* oleh selesainya *state* sebelumnya (*internal processing).* Oleh karena itu *activity diagram* tidak menggambarkan behavior internal sebuah sistem (dan interaksi antar subsistem) secara eksak, tetapi lebih menggambarkan proses-proses dan jalur-jalur aktivitas dari level atas secara umum. Sebuah aktivitas dapat direalsasikan oleh satu *Use Case* atau lebih. Aktivitas menggambarkan proses yang berjalan, sementara *Use Case* menggambarkan bagaimana actor menggunakan sisitem untuk melakukan aktivitas. Berikut adalah activity diagram pada *game* :



Gambar III. 11 Activity Diagram Penyajian *Stage* 1



Gambar III. 12 Activity Diagram Penyajian *Stage* 2



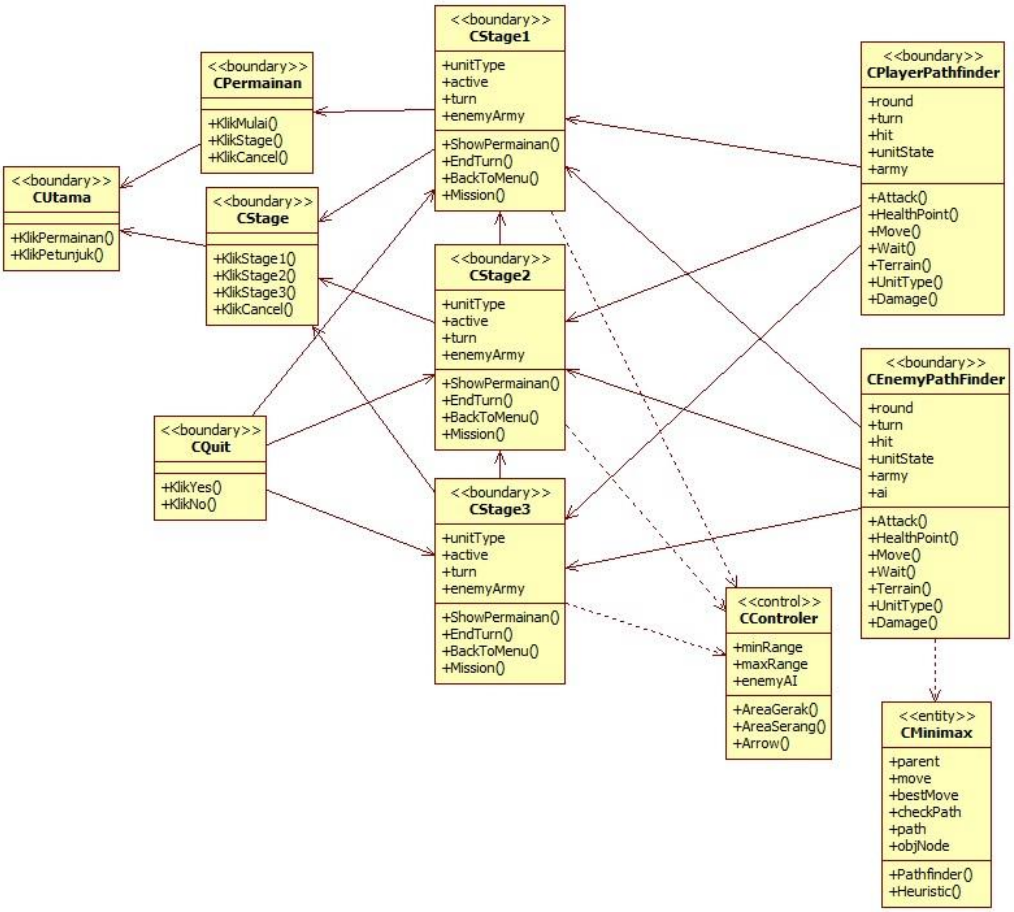
Gambar III. 13 Activity Diagram Penyajian *Stage* 3

6. Class Diagram

Perancangan kelas digambarkan dengan *class diagram*. Diagram kelas atau *class diagram* menggambarkan struktur sistem dari segi pendefinisian kelas-kelas yang akan dibuat untuk membangun sistem. Kelas memiliki apa yang disebut atribut dan metode atau operasi. *Class diagram* ditunjukan pada Gambar III-16. Pendefinisian kelas ditunjukan pada Tabel III-20.

Tabel III. 10 Tabel Pendefinisian Kelas

Jenis Kelas	Nama Kelas
Boundary	CUtama
	CPermainan
	CStage
	CQuit
	CStage1
	CStage2
	CStage3
	CPlayerPathfinder
	CEnemyPathfinder
Entity	CMinimax
Control	CControler

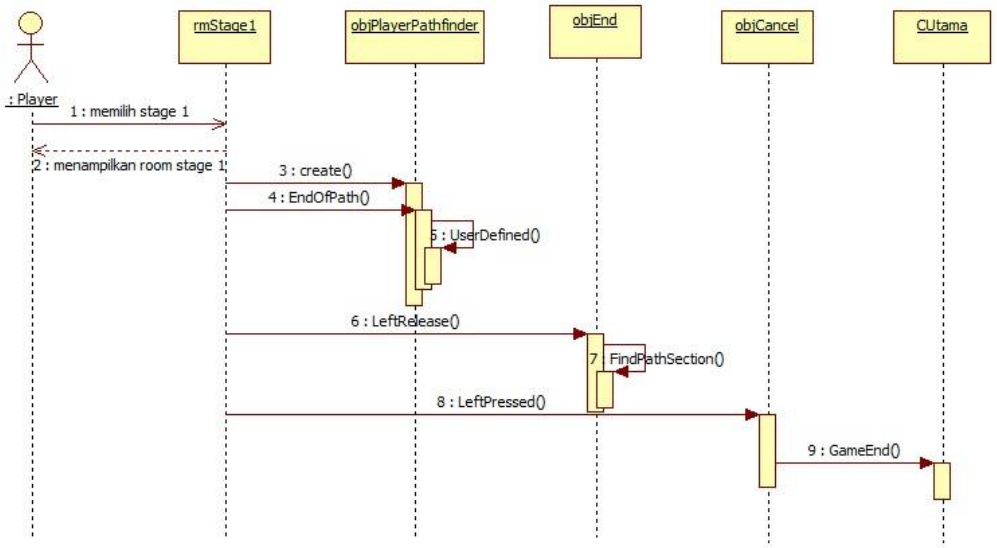


Gambar III. 14 Class Diagram

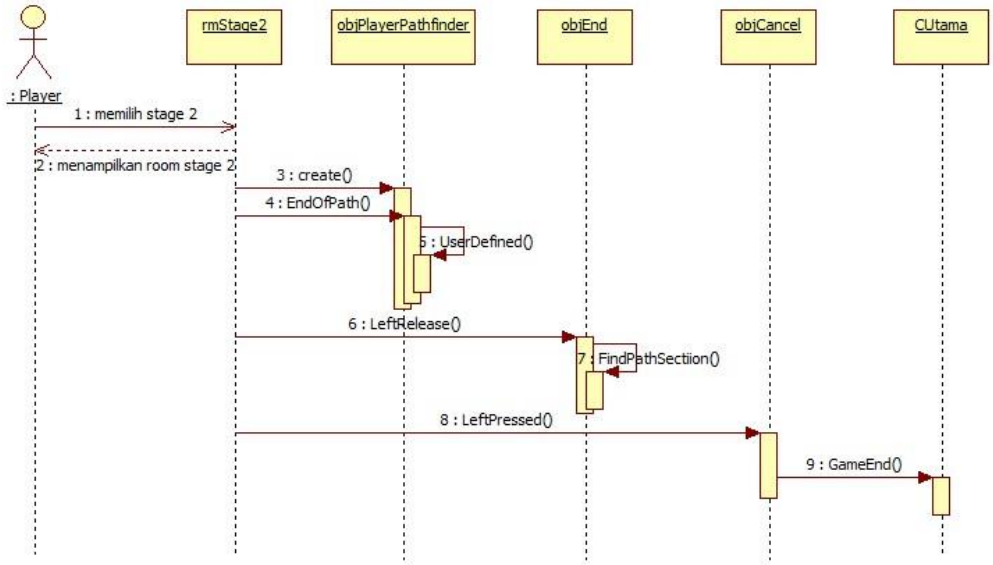
7. Sequence Diagram

Sequence diagram menggambarkan interaksi antar objek di dalam dan disekitar sistem (termasuk pengguna, *display*, dan sebagainya). *Sequence diagram* terdiri atas dimensi *vertical* (waktu) dan dimensi *horizontal* (objek-objek yang terkait).

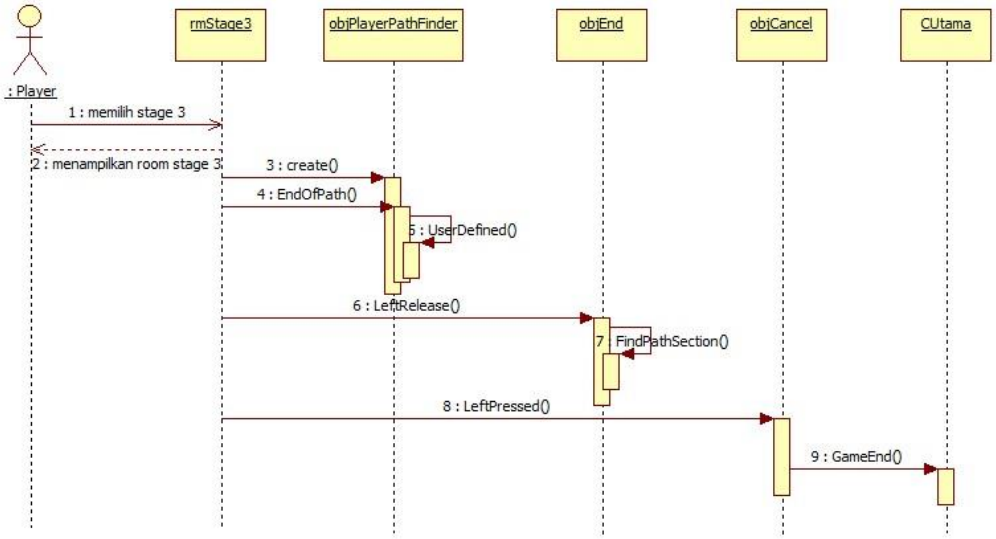
Pada *sequence diagram* ini menggambarkan scenario atau rangkaian langkah-langkah yang dilakukan pada sistem sebagai respon dari sebuah *event* untuk menghasilkan *output* tertentu.



Gambar III. 15 Sequence Diagram Penyajian *Stage* 1



Gambar III. 16 Sequence Diagram Penyajian *Stage* 2



Gambar III. 17 Sequence Diagram Penyajian *Stage* 3

III.2 Perancangan Sistem

Tahap perancangan merupakan kelanjutan dari proses analisis dimana dilakukan perubahan-perubahan terhadap sistem yang sedang berjalan. Hal ini dilakukan untuk mengatasi kekurangan yang ada, memudahkan pekerjaan yang dilakukan oleh orang yang terlibat dan menghemat waktu pekerjaan.

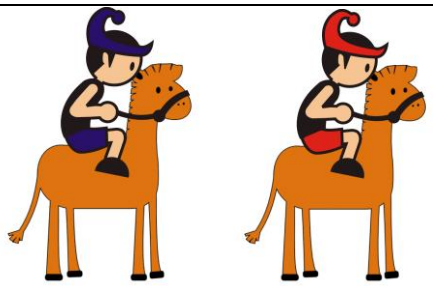
Perancangan *game* ini meliputi perancangan struktur menu, perancangan antarmuka, perancangan pesan, jaringan semantik, perancangan prosedural. Semua perancangan yang telah disebutkan diatas harus saling berhubungan agar menciptakan suatu sistem yang baik.

III.2.1 Karakter

Perancangan karakter adalah bagian dimana karakter pada *game* dijelaskan ciri-cirinya baik itu secara fungsionalitas peran maupun secara desain. Pada *game* ini, karakter merupakan setiap unit pada *game*, sehingga pada perancangan karakter akan mendeskripsikan setiap kelas *piece*.

Tabel III. 11 Karakter

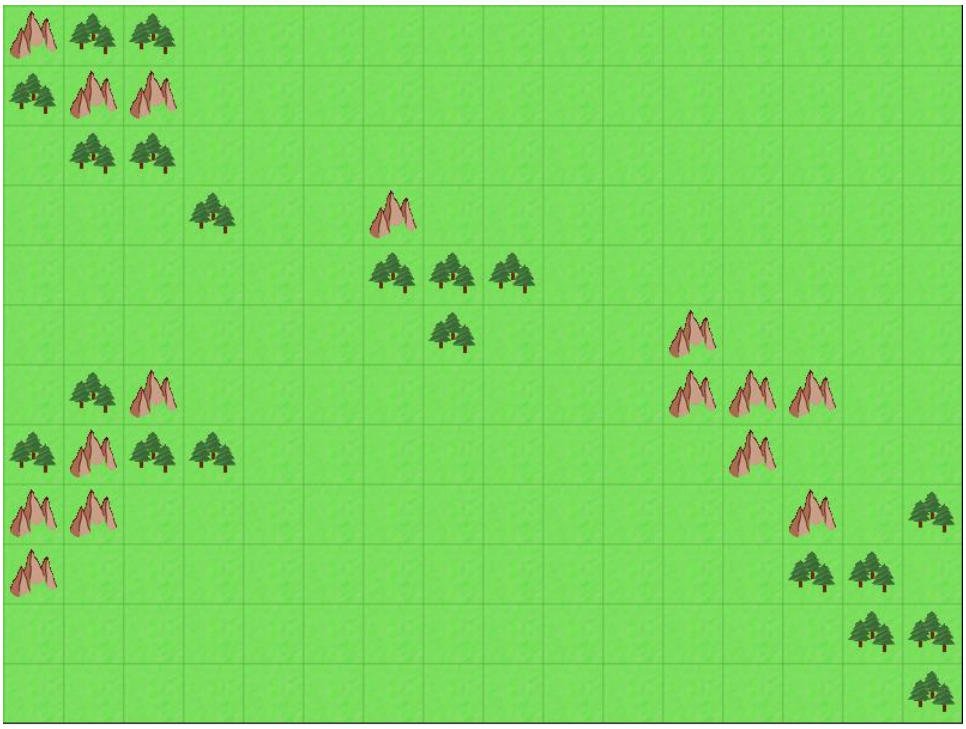
No.	Tampilan Karakter	Nama Karakter	Jarak Gerak	Jarak Serang	Keterangan
1		Raden Wijaya (*Hero)*	2 Kotak	2 Kotak	*Playable Character*
2		Jayakatwang (*Hero)*	2 Kotak	2 Kotak	*NPC*
3		Ike Mese (*Hero)*	2 Kotak	2 Kotak	*NPC*
4		Prajurit	1 Kotak	1 Kotak	*Playable Character* (berwarna biru)/*NPC* (berwarna merah)
5		Pemanah	1 Kotak	2 Kotak	*Playable Character* (berwarna biru)/*NPC* (berwarna merah)

6		Penunggang Kuda	2 Kotak	1 Kotak	*Playable Character* (berwarna biru)/*NPC* (berwarna merah)

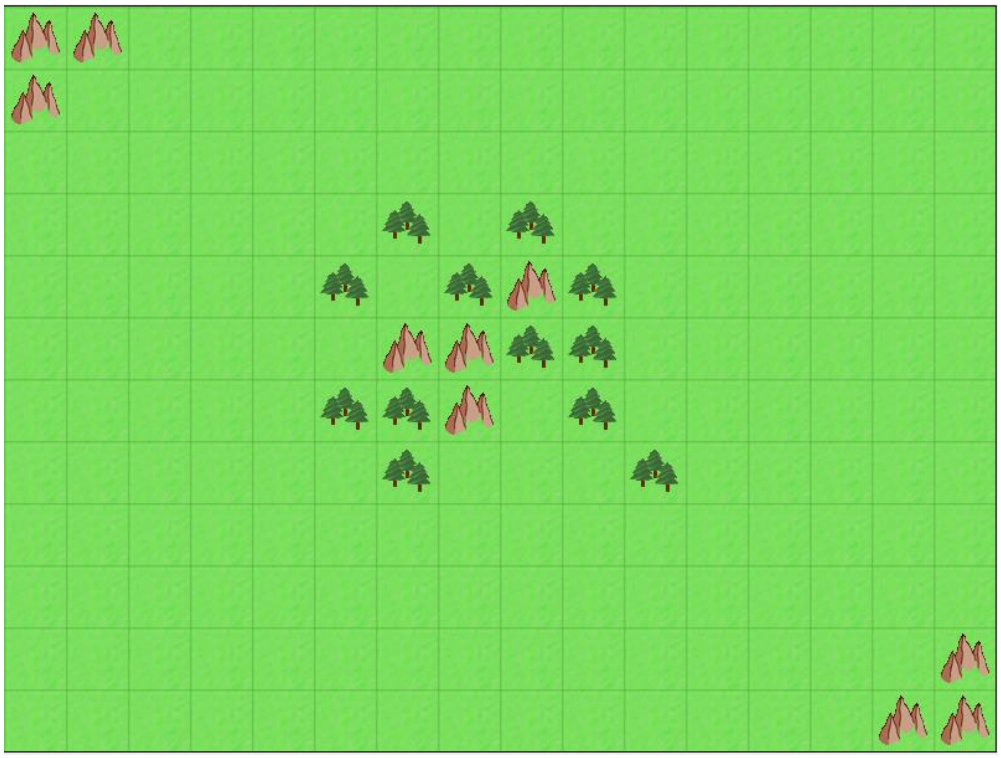
III.2.2 Storyboard

Storyboard adalah sketsa gambar yang disusun berurutan sesuai dengan naskah, dengan storyboard, ide cerita dapat disampaikan kepada pengguna aplikasi dengan lebih mudah, sehingga menghasilkan persepsi yang sama pada ide cerita.

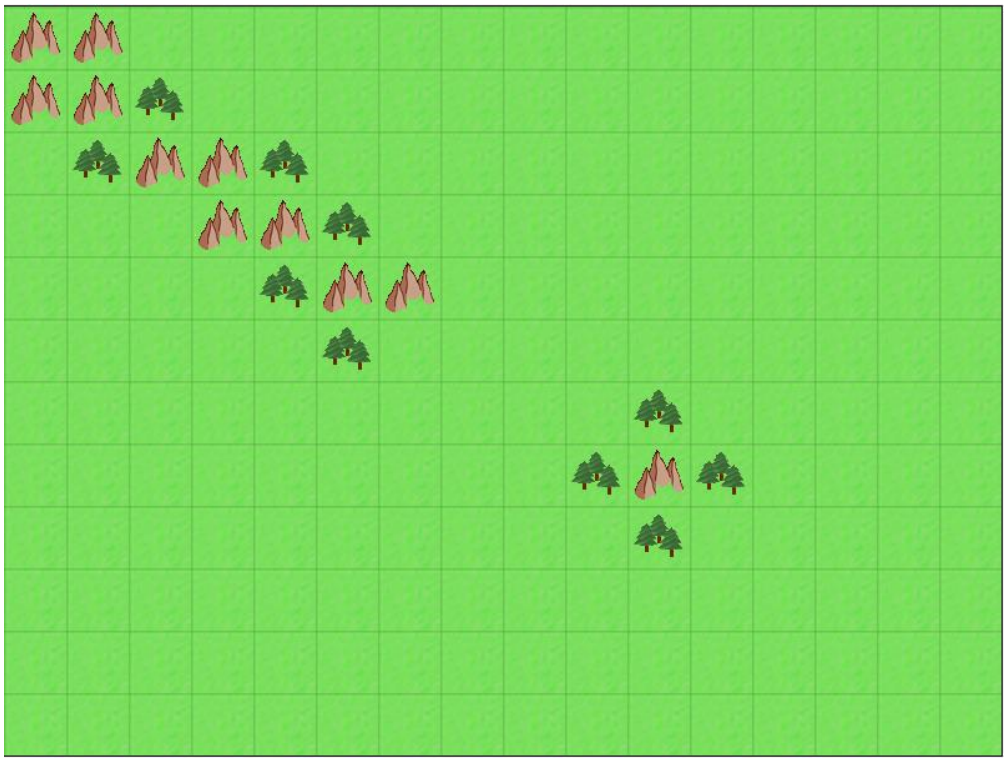
Berikut merupakan storyboard dari Aplikasi Rise of Majapahit :



Gambar III. 18 Storyboard Stage 1



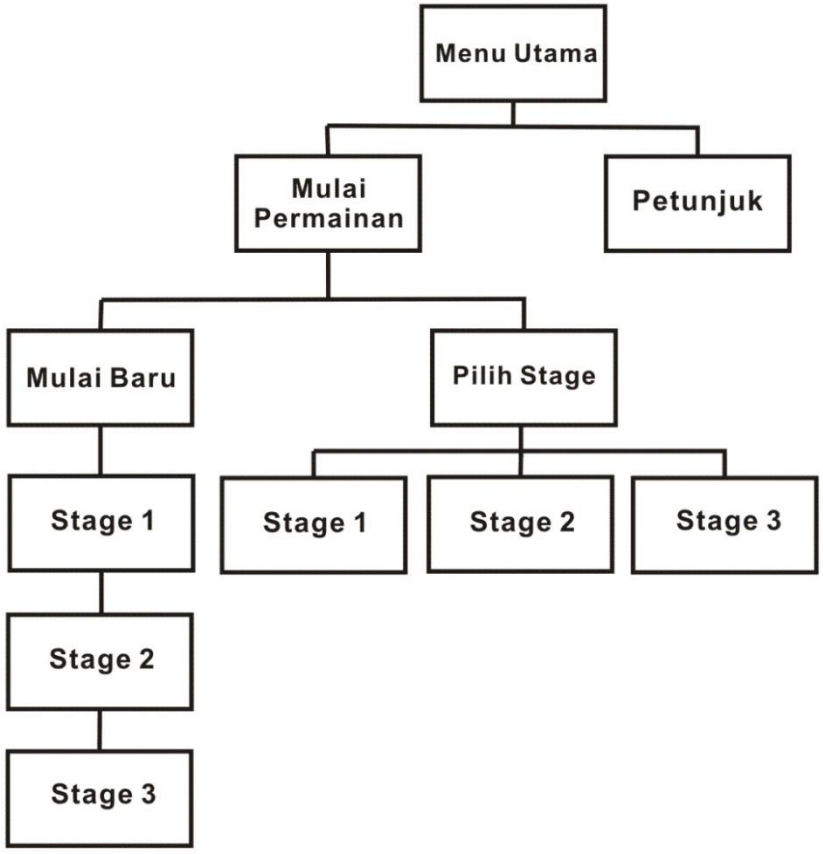
Gambar III. 19 Storyboard Stage 2



Gambar III. 20 Storyboard Stage 3

III.2.3 Perancangan Struktur Menu

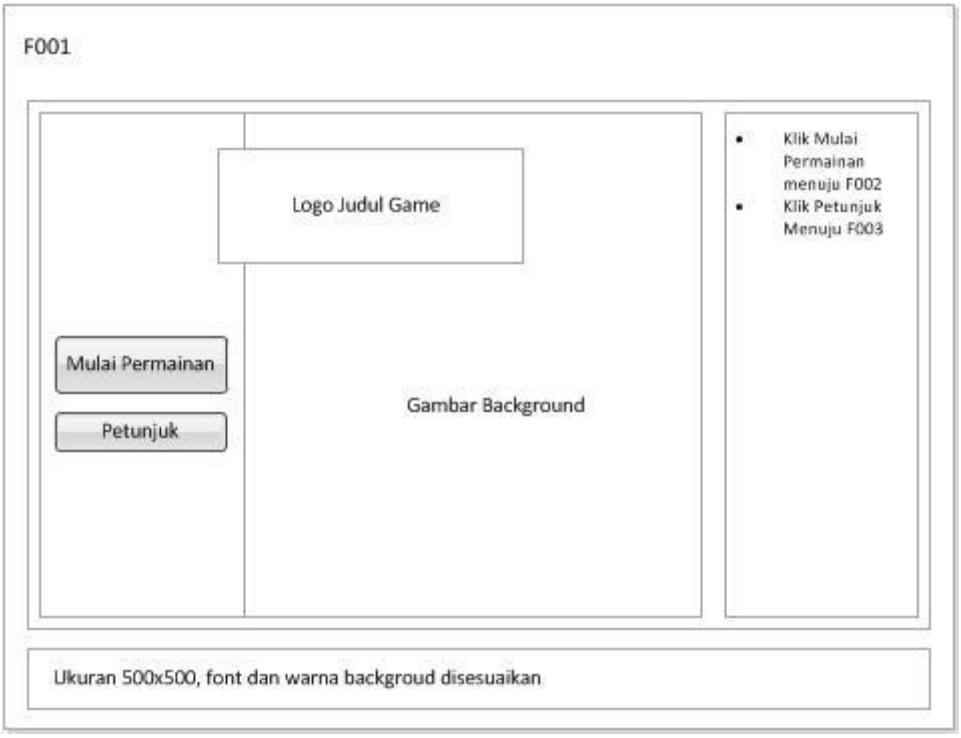
Dalam pembangunan aplikasi diperlukan sebuah struktur menu untuk menunjukan keterkaitan dan tingkatan antara menu utama dan sub menu, sehingga dapat lebih mudah di mengerti. Berikut struktur menu dari *game* yang akan di bangun :



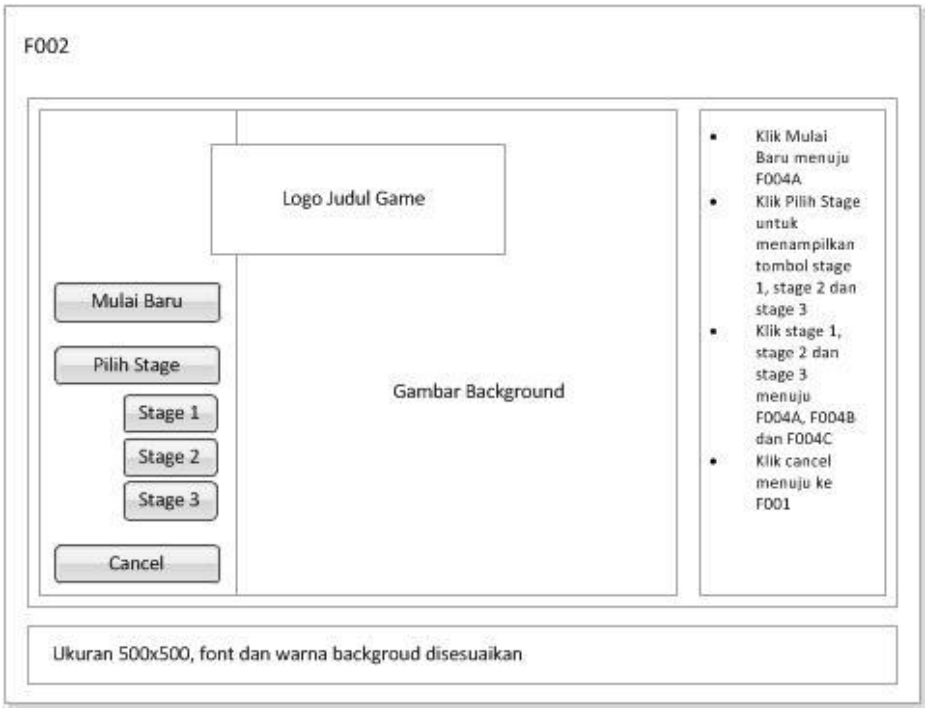
Gambar III. 21 Struktur Menu Dalam *Game*

III.2.4 Perancangan Antar Muka

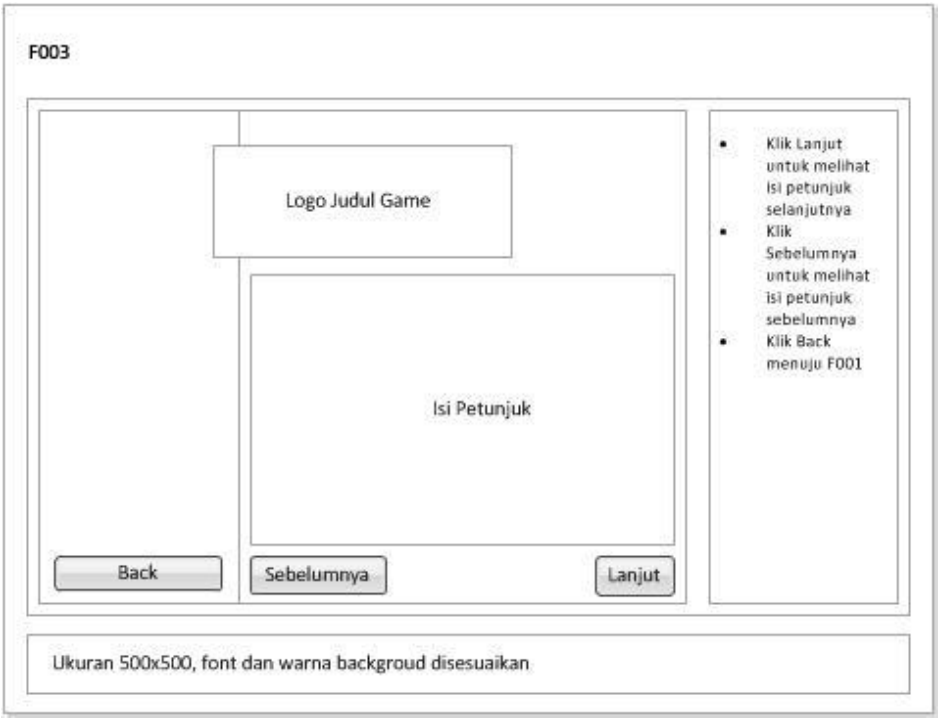
Perancangan antarmuka merupakan salah satu bagian penting dalam perancangan sistem karena nantinya antarmuka tersebut akan menjadi fasilitas yang menjembatani interaksi manusia dengan sistem. Perancangan antarmuka yang akan dibangun pada *game* ini adalah sebagai berikut:



Gambar III. 22 Perancangan Antarmuka Menu Utama



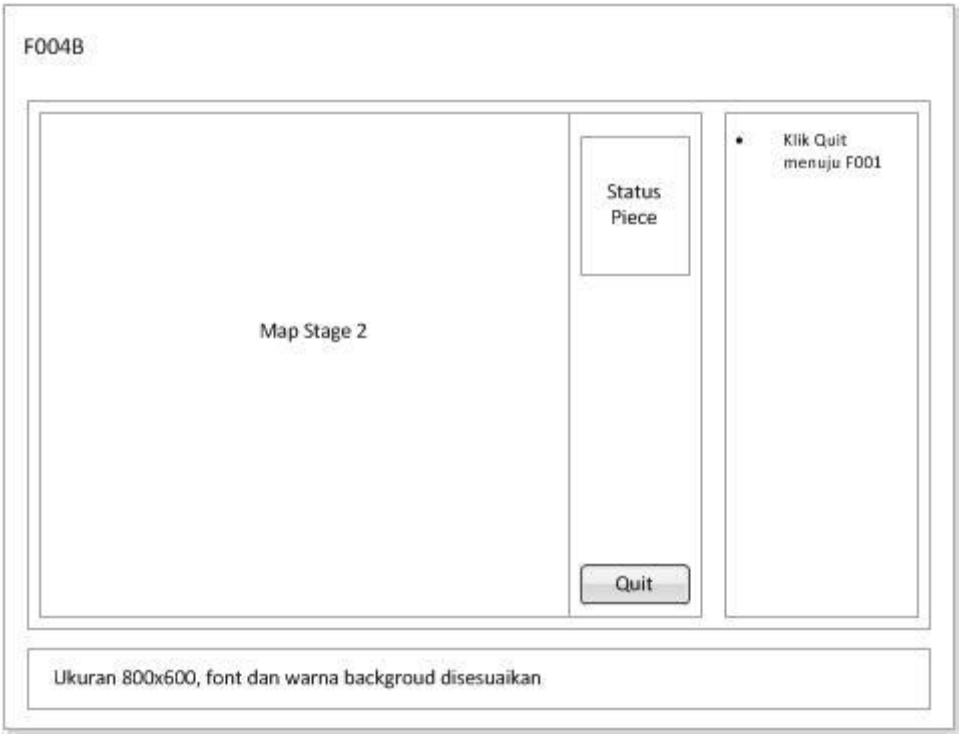
Gambar III. 23 Perancangan Antarmuka Mulai Permainan



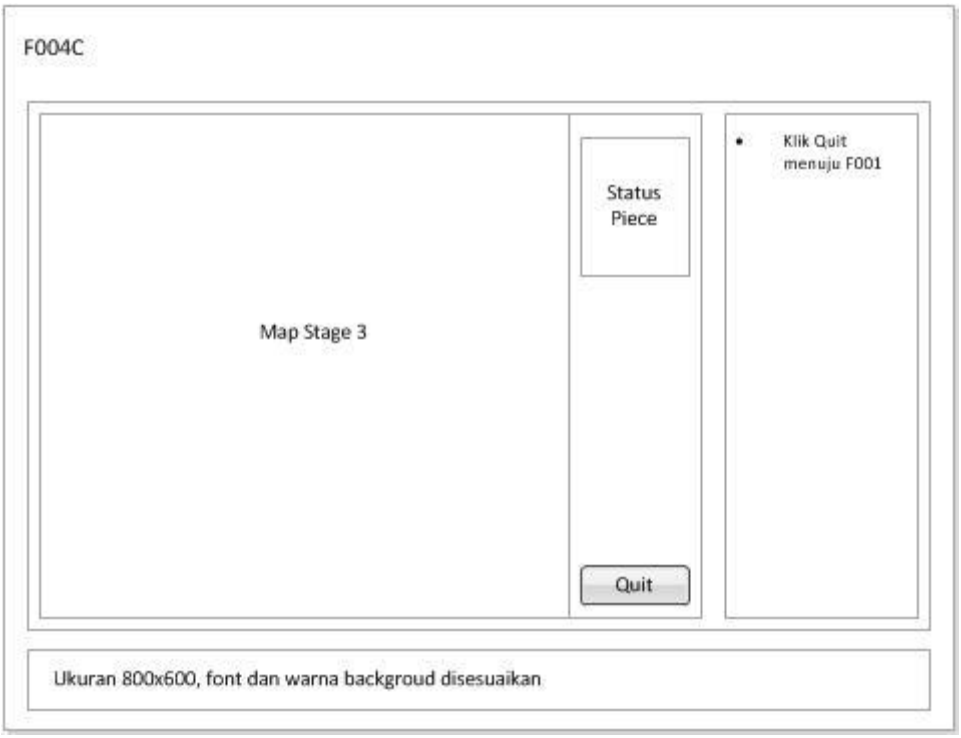
Gambar III. 24 Perancangan Antarmuka Petunjuk



Gambar III. 25 Perancangan Antarmuka Stage 1



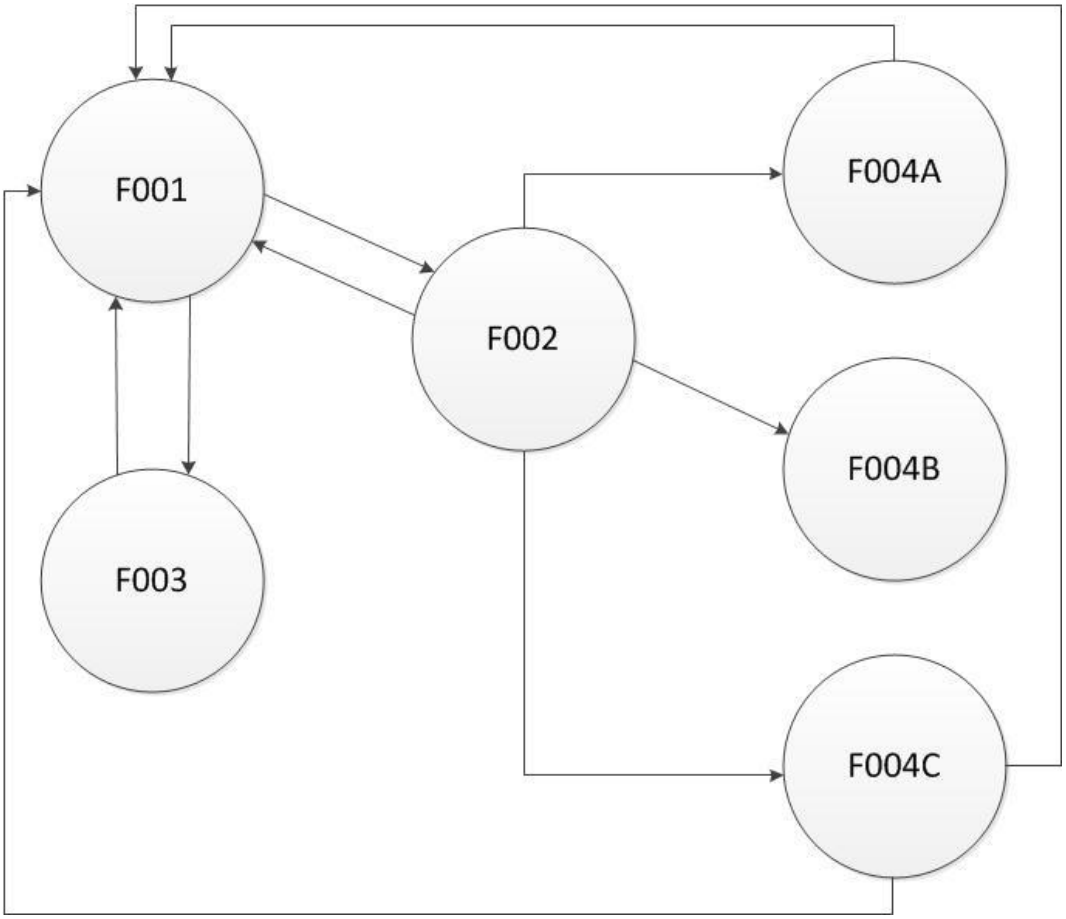
Gambar III. 26 Perancangan Antarmuka Stage 2



Gambar III. 27 Perancangan Antarmuka Stage 3

III.2.5 Jaringan Semantik

Jaringan semantik merupakan alat efektif untuk merepresentasikan pemetaan data, yang bertujuan mencegah terjadinya duplikasi data. Berikut adalah jaringan semantik pada *game* ini :



Gambar III. 28 Jaringan Semantik

III.2.6 Perancangan Method

Perancangan method merupakan pemaparan proses pada method-method yang digunakan pada setiap kelas. Perancangan method digambarkan dengan menggunakan Flow Chart.

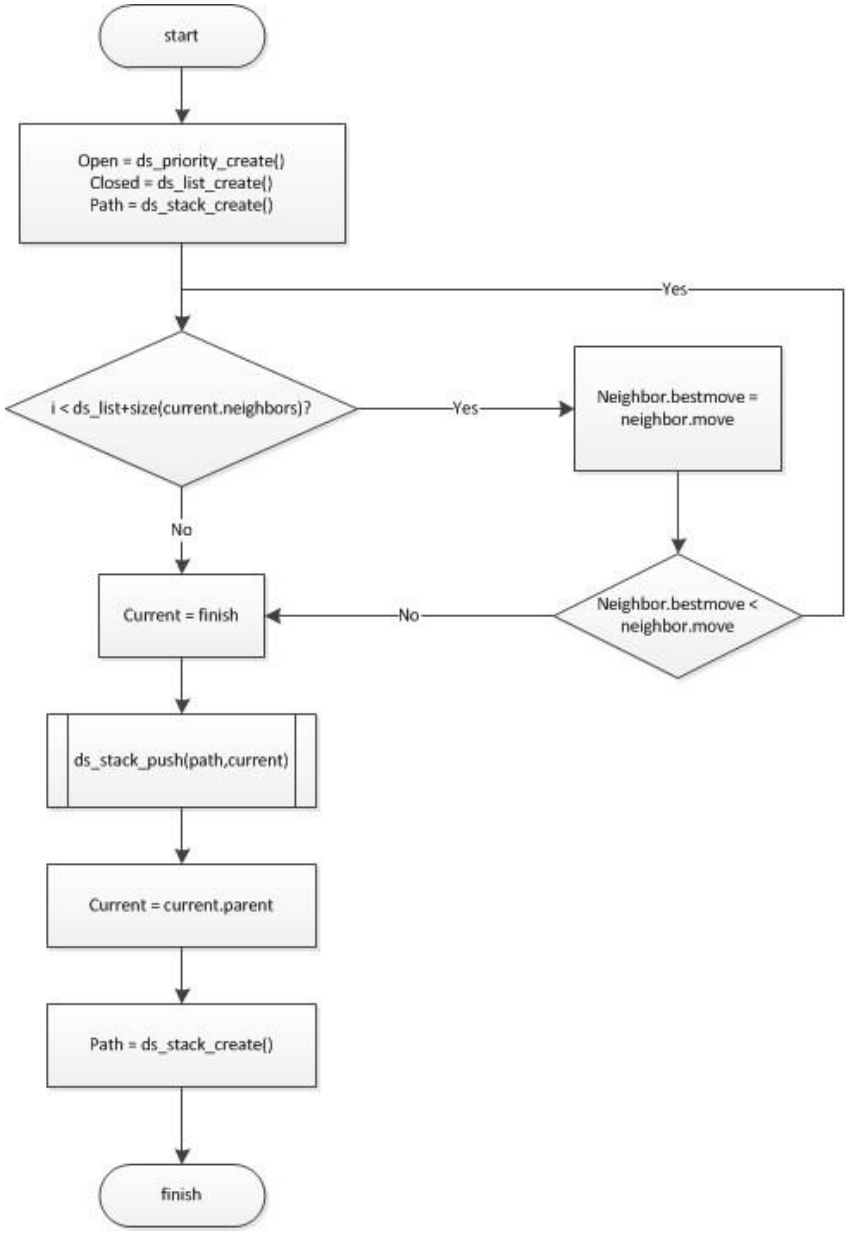
a. Method FindPathSection()

   Method FindPathSection() merupakan method pengecekan pada setiap *cell* dalam jangkauan gerak *piece*. Setiap *cell* dicek, sehingga menghasilkan satu nilai dengan nama *move*, yang nantinya akan dipilih satu nilai keputusan

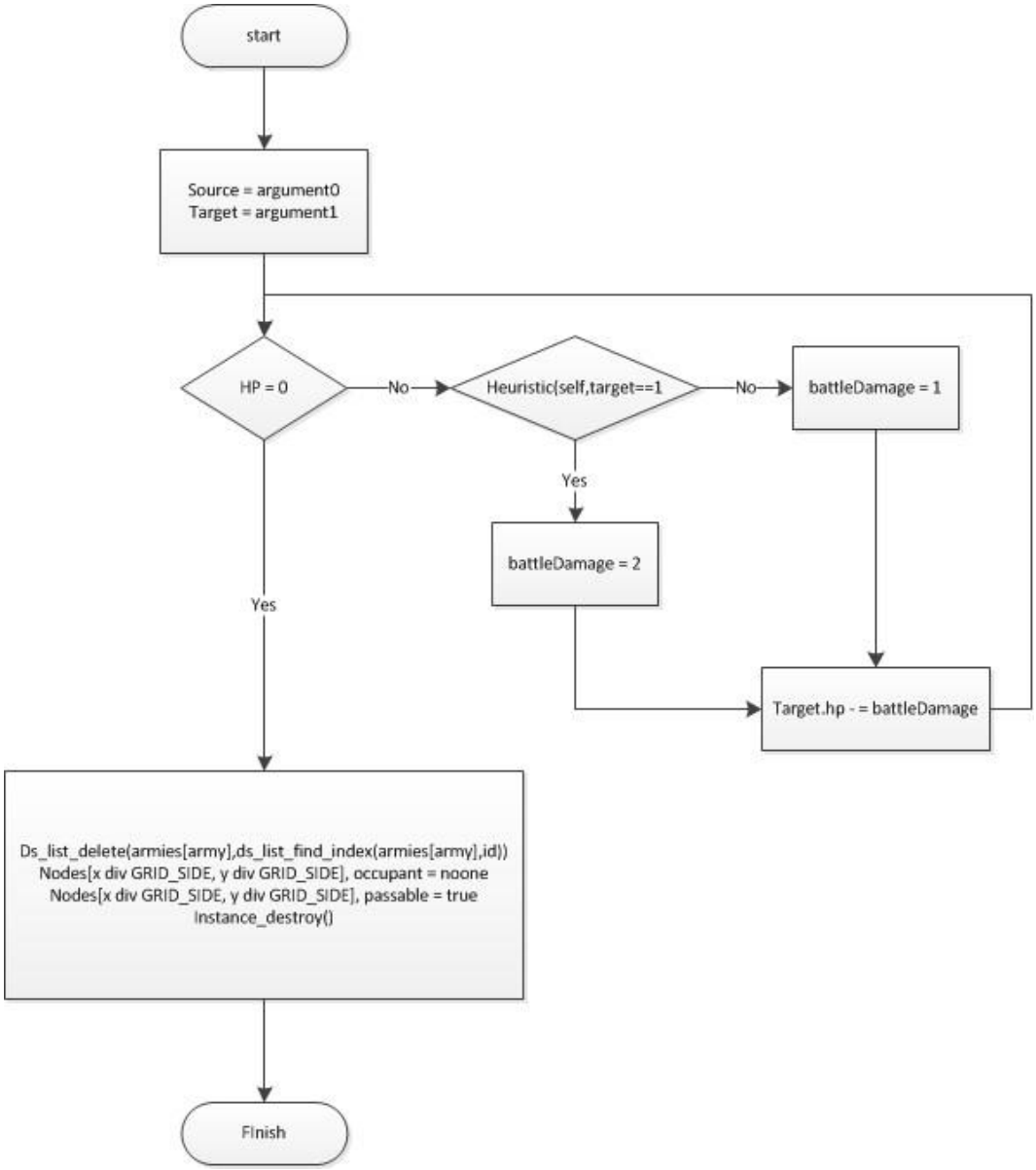
berdasarkan nilai *best move* dari seluruh *move*. Method ini digambarkan pada Gambar III.22.

b. Method Attack()

Method Attack() adalah method perhitungan untuk mengurangi atau menghapus *piece* yang mati. Bila *piece* mendapat serangan dari *piece* lawan, maka *health point* akan dikurangi, dan apabila jumlah *health point* = 0, *piece* akan dihapus. Method ini digambarkan pada Gambar III.23.



Gambar III. 29 Method FindPathSection()



Gambar III. 30 Method Heuristic()