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

Adanya berbagai faktor pada *game* TBS menimbulkan beberapa masalah, diantaranya adalah :

1. Bagaimana memunculkan kemampuan berikir dan berstrategi secara logis pada AI yang terdapat di *gameturn-based strategy*?
2. Bagaimana agar *gameturn-based strategy* dapat dimainkan kapan saja dan dimana saja?

Solusi yang dapat memecahkan masalah tersebut adalah dengan memakai algoritma *resource assignment* sebagai pembuat keputusan pada AI.Algorima resource assignment memiliki kemampuan untuk membuat satu keputusan dari berbagai keputusan yang ada berdasarkan kondisi dan skala prioritas pada setiap keputusan tersebut.

III.1.2. Analisis Game Yang Akan Dikembangkan

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

III.1.2.1. *Story Line*

Storyline merupakan cerita pada game dimana pada *storyline* sebuah game akan ditentukan bagaimana desainnya dan aturan permainannya. Dengan tema perang moderen dan mengambil *setting* tahun

2106, diceritakan ketika tahun 2087, planet bumi ada pada kondisi yang sangat mengkawatirkan dikarenakan efek pemanasan global berada pada puncaknya. Karena dikhawatirkan akan mengancam kelangsungan hidup umat manusia, maka dibawah Perserikatan Bangsa Bangsa (PBB) umat manusia memulai proyek terbesar sepanjang sejarah yaitu membuat alat yang akan mendinginkan bumi dengan memperkuat medan magnetik pada kutub bumi. PBB kemudian membagi negara-negara kepada dua fraksi yaitu *Northern Alliance* yang dipimpin oleh Amerika Serikatdan *Southern Alliance* yang dipimpin oleh Russia.Tugas masing-masing fraksi adalah membuat alat pada masing-masing kutub magnetik di bumi untuk mengurangi efek pemanasan global pada bumi.

Pada tahun 2104 alat untuk memperkuat medan magnetik bumi telah selesai. Alat tersebut kemudian dioperasikan pada 19 Oktober 2104. Namun diluar dugaan, medan magnetik bumi menjadi terlalu kuat sehingga intensitas cahaya matahari yang masuk ke bumi menjadi sangat rendah. Kini bumi mengalami zaman es ke-3. Bumi bagian utara dan selatan kini tak bias ditinggali manusia. Umat manusia pindah ke bagian ekuator dimana sinar matahari, namun mereka terpecah menjadi dua kubu yang saling menyalahkan atas tragedy menyelamatkan bumi dari pemanasan global yang berujung pada pendinginan global.Kubu UNION yang berpusat di Columbia dan *Great Eastern Republic*(GER)yang berpusat di Indonesia.

Pada tahun 2106 dimulailah *The Icycle War* yaitu perang besar antara UNION dan GER pada zaman es ke-3.Perang kini tidak menggunakan manusia pada pertempuran mengingat jumlah manusia telah berkurang banyak akibat ganasnya zaman es ke-3.*Combat Gadget Droid* (CGD) adalah sebuah robot yang menyerupai manusia diciptakan untuk menggantikan para prajurit manusia dalam berperang. Mereka dibagi menjadi tiga kelas utama.Infantry yang memiliki pergerakan lincah, tank yang memiliki armor yang tebal dan kuat, serta artillery yang memiliki jangkauan dan kekuatan tembak yang tinggi. Dengan adanya CGD, tidak akan ada lagi manusia yang hilang nyawanya dalam perang. Namun perang juga tidak akan pernah berakhir.

Pada game ini, player akan menjadi komandan pasukan GER yang akan bertempur dengan mengontrol *Combat Gadget Droid* yang merupakan unit pada *game*. Berdasarkan cerita diatas maka game ini diberi judul "*The Icycle War - 2106*".

Jenis misi yang terdapat pada game ini adalah mendominasi wilayah pada *map* yang terdapat pada setiap *stage*. Sistem pendominasian terletak pada jumlah *node* yang dikusai, semakin banyak *node* yang dikusai maka semakin besar dominasi pada *map*. *Player* yang menguasai seluruh *node* pada *map* adalah pemenangnya.

III.1.2.2. 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 *node*	6	Jumlah *node* pada *map*
Jumlah *node player*	1	Jumlah *node* milik *player* yang diberikan pada awal permainan
Jumlah *node* musuh	1	Jumlah *node* milik musuh yang diberikan pada awal permainan
Jumlah unit *player*	8	Terdiri dari 5 *Infantry*, 2 *Tank*, 1 *Artillery*
Jumlah unit musuh	8	Terdiri dari 5 *Infantry*, 2 *Tank*, 1 *Artillery*
Jumlah *resource point player*	100	Jumlah *resource point* yang diberikan pada player pada awal permainan
Jumlah *resource point*	100	Jumlah *resource point* yang

musuh		diberikan pada musuh pada awal permainan

2. Stage 2

Detail tingkat kesulitan pada stage 2 dijabarkan pada Tabel III-2

Tabel III-2 Detail Tingkat Kesulitan Pada Stage 2

	Jumlah	Keterangan
Jumlah *node*	6	Jumlah *node* pada *map*
Jumlah *node player*	1	Jumlah *node* milik *player* yang diberikan pada awal permainan
Jumlah *node* musuh	1	Jumlah *node* milik musuh yang diberikan pada awal permainan
Jumlah unit *player*	8	Terdiri dari 5 *Infantry*, 2 *Tank*, 1 *Artillery*
Jumlah unit musuh	10	Terdiri dari 5 *Infantry*, 3 *Tank*, 2 *Artillery*
Jumlah *resource point player*	80	Jumlah *resource point* yang diberikan pada player pada awal permainan
Jumlah *resource point* musuh	80	Jumlah *resource point* yang diberikan pada musuh pada awal permainan

3. Stage 3

Detail tingkat kesulitan pada stage 3 dijabarkan pada Tabel III-3.

Tabel III-3 Detail Tingkat Kesulitan Pada Stage 3

	Jumlah	Keterangan
Jumlah *node*	8	Jumlah *node* pada *map*
Jumlah *node player*	1	Jumlah *node* milik *player* yang diberikan pada awal permainan
Jumlah *node* musuh	1	Jumlah *node* milik musuh yang

		diberikan pada awal permainan
Jumlah unit *player*	8	Terdiri dari 5 *Infantry*, 2 *Tank*, 1 *Artillery*
Jumlah unit musuh	10	Terdiri dari 5 *Infantry*, 3 *Tank*, 2 *Artillery*
Jumlah *resource point player*	100	Jumlah *resource point* yang diberikan pada player pada awal permainan
Jumlah *resource point* musuh	150	Jumlah *resource point* yang diberikan pada musuh pada awal permainan

III.1.2.3. *Gameplay*

Analisis *gameplay* dilakukan untuk menggambarkan aturan-aturan dalam game. Analisis *gameplay* terdiri dari deskripsi alur permainan, deskripsi aturan game dan deskripsi kondisi pada game.

1. Deskripsi Alur Permainan

Alur permainan merupakan urutan proses yang dilalui player secara bertahap dalam game. Pada game ini alur permainan sangat bergantung pada keputusan player sehingga alur permainan tidak dapat dipastikan. Hanya saja player harus mengalahkan musuhnya dengan menguasai *node* yang ada pada permainan.

Node dapat dikuasai oleh player dengan cara menempatkan pasukan pada *node* tersebut. Player akan diberi sejumlah pasukan pada awal permainan. Untuk memperbanyak pasukannya, player harus memproduksi pasukan sehingga dapat digunakan untuk mengusai *node*.Dalam memproduksi pasukan player harus menggunakan resource point untuk memproduksi pasukan.Resource point didapat dari setiap *node* yang dikuasai. Setiap *node* yang dikuasai akan memberikan 10 resource point setiap turn-nya.

2. Deskripsi Aturan Game

Aturan (*rule*) pada game dibuat untuk menjaga keseimbangan permainan sehingga permainan tidak memberatkan atau meringankan player. Berikut adalah aturan utama yang akan diterapkan pada game ini :

(a) Player membuat keputusan terhadap sumber daya yang dimilikinya yang berupa *unit* ,*resource*, dan *node*.

(b) Kondisi untuk memenangkan pertempuran adalah ketika seluruh *node* pada *map* berhasil dikuasai.

Dalam game ini, player harus mengambil keputusan yang dapat diambil pada setiap turn. Berikut adalah aturan pengambilan keputusan untuk setiap unit :

(a) Jenis keputusan yang dapat diambil adalah pass, build unit, attack, assign unit.

(b) Pass adalah keputusan player untuk tidak melakukan apa-apa pada *turn*-nya.

(c) Build unit adalah keputusan untuk membuat unit, unit dapat dibuat dengan uang yang dimiliki player.

(d) Attack adalah keputusan menyerang *node* netral atau *node* milik musuh dengan mengirimkan beberapa unit yang ada ke *node* tersebut.

(e) Assign unit adalah keputusan menempatkan unit pada node yang dimiliki player dengan tujuan mempertahankan node tersebut dari serangan musuh.

Turn-based strategy game mempunyai sistem *turn* pada *gameplay*-nya.*Turn* adalah kondisi dimana suatu keputusan harus diambil.*Turn* dimulai dari *player* dan dilanjutkan dengan musuhnya (AI), lalu kembali dilanjutkan oleh *player* dan seterusnya hingga permainan berakhir ketika seluuh *node* dikuasai oleh salah satu pihak.

Unit pada game ini dibagi menjadi tiga kelas yaitu *infantry*, *tank* dan *artillery*. Sistem pembagian kelas ini memungkinkan timbulnya faktor keuntungan dan kerugian ketika suatu kelas unit berhadapan dengan kelas unit yang lain.Perbedaan kelas pada unit terdapat pada kekuatannya, dimana

infantry lebih lemahdari *artillery* dan *tank*, *tank* lebih kuat dari *infantry* dan lebih lemah dari *artillery*, sedangkan *artillery* lebih kuat *tank* dan *infantry*.

3. Deskripsi Kondisi Pada Game

   (a) Keadaan awal (*starting condition*)

   Pada saat memulai game, player dan AI diberikan kondisi sebagai berikut :

   (i) Terdapat 1 *node* yang telah dikuasai.

   (ii) Diberikan sejumlah resource untuk memproduksi unit.

   (iii) Diberikan sejumlah unit untuk mengusai atau mempertahankan *node*.

   (b) Keadaan dalam game (*in-game condition*)

   Pada saat bermain, player harus melakukan tugas-tugas sebagai berikut :

   (i) Mempertahankan *node* yang dimiliki.

   (ii) Menyerang dan mengambil alih *node* musuh atau *node* netral.

   (iii) Memproduksi unit.

   (iv) Tidak melakukan apa-apa (*skip turn*)

   (c) Keadaan akhir (*final condition*)

   Gameakan berakhir dengan kondisi menang (*victory*) jika seluruh *node* berhasil dikuasai oleh player. Sedangkan game akan berakhir dengan kondisi kalah (*lose*) jika seluruh *node* dikuasai oleh musuh (AI) atau player keluar dari game.

III.1.2.4. *Scoring*

Sistem pemberian skor (*scoring*) pada game ini terjadi ketika *player* telah menyelesaikan game pada suatu stage. Pemberian skor pada *player* tidak hanya ketika *player* memenangkan permainan tetapi ketika *player* kalah pada permainan. Skor yang diberikan berupa jumlah *resource* yang telah *player* habiskan selama permainan.

III.1.3. Analisis Struktur Data

Struktur data yang digunakan didalam penelitian ini adalah struktur data *linked list*. Parameter yang menggunakan *linked list* antara lain adalah parameter *node* yang dideklarasikan dalam bentuk berikut :

```
Type Node : <
      Nodeid : integer
      Stat : integer
      Infantry : integer
      Tank : integer
      Artillery : integer
>
Type List : <
      First : integer,
      Nodes : array [1..8] of Node
>
```

Parameter *node* menggunakan *linked list* yang berisikan atribut nodeid, stat, infantry, tank dan artillery. Atribut tersebut merupakan *resource* yang akan berpengaruh dalam perhitungan nilai keputusan menggunakan algoritma *resource assignment*.

III.1.4. Analisis Kebutuhan Game

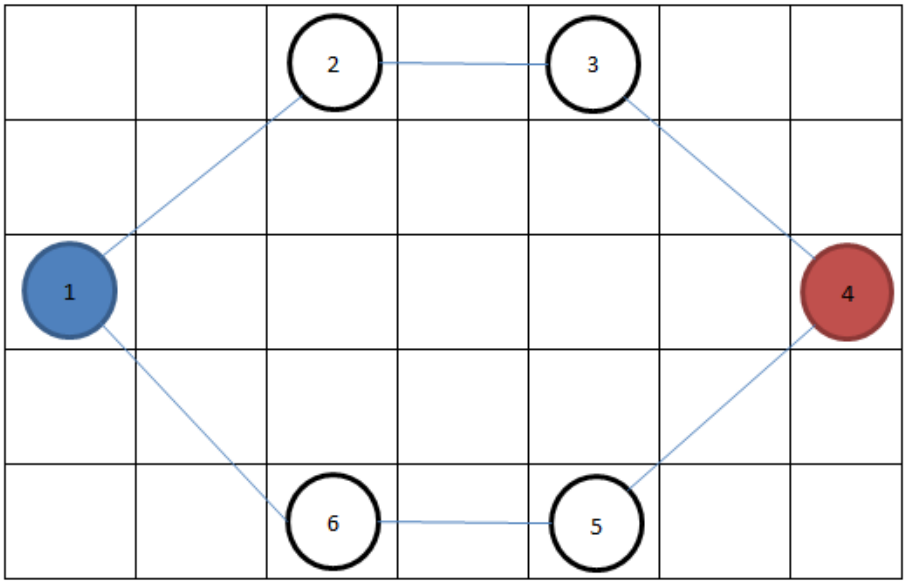
Pada *game* ini terdapat objek-objek dan *resource* yang akan berperan dalam jalannya permainan ini. Objek dan *resource* dapat dilihat pada Tabel III. 4.

Tabel III. 4 Objek dan Resource Permainan

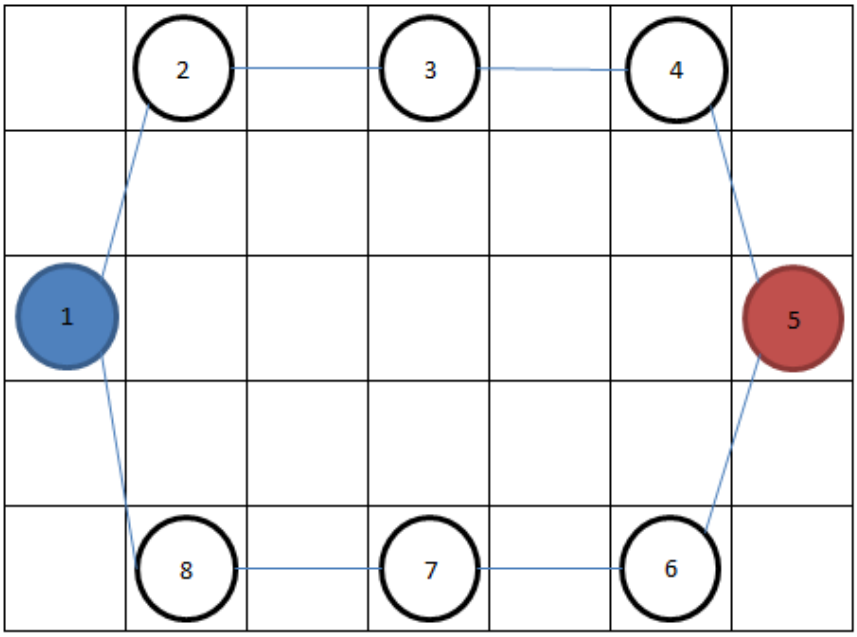
Nama	Keterangan
Node	Merupakan objek yang menampung *resource* pada game.
Infantry	Merupakan salah satu *resource* pada game.
Tank	Merupakan salah satu *resource* pada game.
Artillery	Merupakan salah satu *resource* pada game.
Modifier	Merupakan nilai yang terdapat pada setiap keputusan yang akan dijalankan oleh AI. Semakin besar nilai ini, semakin diprioritaskan pula sebuah keputusan akan dijalankan oleh AI.

Map	Merupakan arena pada permainan di setiap *stage* yang direpresentasikan menggunakan Graph dengan *node* sebagai simpulnya.

Selain objek dan *resource*, permainan ini juga membutuhkan map yang dibentuk yang direpresentasikan dalam bentuk graph tertutup dan tidak berarah. Di dalam *map* terdapat *node*, yaitu simpul-simpul yang harus dikuasai oleh *player*. Pada bagian perancangan ini, *node* digambarkan dengan bulatan berwarna. Angka dalam *node* merepresentasikan urutan *node*. Warna pada *node* merepresentasikan status kepemilikan *node* tersebut. Warna merah untuk *node* milik musuh (AI), warna biru untuk *node* milik *player* dan warna putih untuk *node* netral. Setiap *node* dihubungkan dengan menggunakan garis penghubung yang merepresentasikan hubungan tetangga dari sebuah *node*, contohnya adalah *node* 1 memiliki dua tetangga yaitu *node* 2 dan *node 6* pada Gambar III-2. Penggambaran inisialisasi map pada permainan ini dapat ini dapat dilihat pada Gambar III-2 dan Gambar III-3.



Gambar III-2 *Map Stage 1 dan Stage 2*



Gambar III-3 *Map Stage 3*

Pada stage 1 dan 2, map tersusun dari 6 node yang mana pada awal permainan dimiliki node 1 dimiliki oleh player dan node 4 dimiliki oleh AI. Kepemilikan node dibedakan dengan warna yang mana apabila node berwarna biru, merupakan kepemilikan player dan merah merupakan kepemilkan AI. Pada Gambar III-2 terlihat bahwa setiap node memiliki tetangga yang ditunjukan pada tabel berikut.

Tabel III. 5 Daftar Tetangga Untuk Setiap Node Pada Map Stage 1 dan Stage 2

Node	Tetangga 1	Tetangga 2
1	2	6
2	1	3
3	2	4
4	3	5
5	4	6
6	5	1

Pada Stage 3, map tersusun dari 8 node yang mana pada awal permainan dimiliki node 1 dimiliki oleh player dan node 5 dimiliki oleh AI. Kepemilikan node dibedakan dengan warna yang mana apabila node berwarna biru, merupakan kepemilikan player dan merah merupakan kepemilkan AI, sama seperti pada stage 1 dan 2. Pada Gambar III-3 terlihat bahwa setiap node memiliki tetangga yang ditunjukan pada tabel berikut.

Tabel III. 6 Daftar Tetangga Untuk Setiap Node Pada Map Stage 1

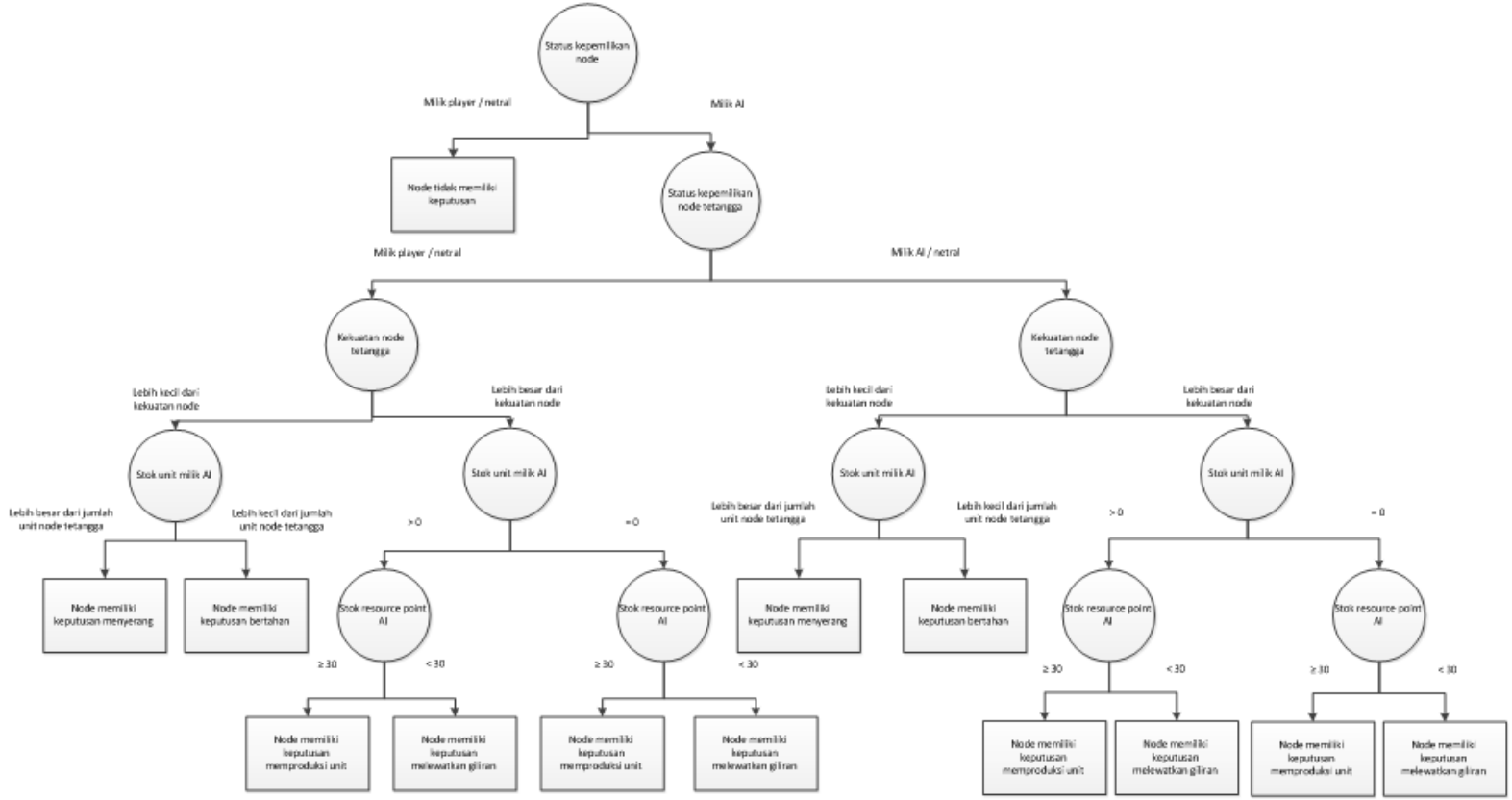
Node	Tetangga 1	Tetangga 2
1	2	6
2	1	3
3	2	4
4	3	5
5	4	6
6	5	1

III.1.5. Analisis Algoritma

Analisis algoritma yang dilakukan dalam penelitian ini adalah meneliti bagaimana cara kera algoritma *resource assignment* dalam menentukan keputusan pada AI di dalam *game*.

III.1.3.1. Algoritma Resource Assignment

Algoritma *resource assignment* (RAA) adalah sebuah algoritma yang dapat menentukan sebuah nilai untuk sebuah objek yang memiliki *resource* berdasarkan nilai *resource* yang berasal dari objek lain maupun objek itu sendiri. Cara kerja algoritma *resource assignment* adalah dengan memeriksa status kepemilikan *node* dan menghitung nilai *modifier* pada setiap keputusan. Keputusan dengan nilai *modifier* terbesar adalah keputusan yang dipilih dan ditempatkan pada *node* tersebut. Setelah seluruh *node* diperiksa, maka nilai *modifier* pada setiap *node* diurutkan dan dipilih *node* dengan nilai *modifier* terbesar. *Node* tersebut adalah *node* yang keputusannya akan dijalankan oleh AI. Penggambaran penentuan keputusan pada setiap *node* ditunjukan pada Gambar III-4.



Gambar III-4 Pohon Keputusan Dalam Penentuan Keputusan Pada Setiap *Node*

Secara keseluruhan algoritma ini terbagi menjadi empat bagian yaitu *gather task*, *generate all possible assignments*, *sort possible assignment according to score* dan *assign to task doers*. Penerapan bagian-bagian tersebut pada proses pengambilan keputusan pada *game* ini adalah sebagai berikut:

1. *Gather Task*

   Proses dimana menentukan banyaknya *node* pada *map* serta penentuan tetangga untuk setiap *node*.

2. *Generate All Possible Assignments*

   Proses penentuan nilai *modifier* yang dihitung berdasarkan semua *resource* (*infantry, tank, artillery*) yang kemudian setelah memperhitungkan nilai *modifier*nya akan diambil keputusan apakah akan menyerang, bertahan, memproduksi unit, melewatkan giliran dan tidak memiliki keputusan sama sekali. Langkah-langkah dalam menentukan nilai *modifier* pada setiap *node* ditentukan dari aturan pada permainan. Langkah-langkah tersebut adalah sebagai berikut :

   a. Periksa status kepemilikan setiap *node*, jika status kepemilikan *node* adalah milik *player* atau netral, maka nilai *modifier* untuk *node* tersebut adalah 0 dan *node* tersebut tidak mempunyai keputusan (nilai keputusan = -1).

   b. Jika status kepemilikan *node* adalah milik AI maka dilakukan perhitungan nilai modifier untuk setiap keputusan berikut :

   I. Keputusan menyerang

      Pada keputusan ini, dilakukan pemeriksaan kondisi pada *node* sebagai berikut :

      (i) Apakah status *node* tetangga dari *node* yang sedang diperiksa adalah milik *player* atau netral atau milik musuh (AI). Bila status kepemilikannya adalah milik *player* atau netral, maka nilai *modifier* pada keputusan ini ditambah 1. Bila status kepemilikan adalah milik musuh (AI), maka nilai *modifier* tidak bertambah.

      (ii) Bandingkan kekuatan *node* yang sedang diperiksa dengan kekuatan *node* tetangganya. Apabila kekuatan *node* yang sedang diperiksa

lebih besar kekuatannya dibandingkan dengan *node* tetangganya maka nilai *modifier* pada keputusan ini ditambah 1. Jika kekuatan *node* yang sedang diperiksa lebih kecil kekuatannya dibandingkan dengan *node* tetangganya maka nilai *modifier* pada keputusan ini tidak ditambahkan.

(iii) Bandingkan stok *unit* AI dengan jumlah unit pada *node* tetangga. Apabila stok unit AI lebih banyak jumlahnya debandingkan dengan stok unit pada *node* tetangga, maka nilai *modifier* pada keputusan ini ditambah 1. stok unit AI lebih banyak jumlahnya debandingkan dengan stok unit pada *node* tetangga, maka nilai *modifier* pada keputusan ini tidak ditambahkan.

(iv) Nilai *modifier* akhir untuk keputusan ini adalah jumlah dari nilai *modifier* pada kondisi yang cocok dibagi dengan jumlah seluruh kondisi yaitu 6.

II. Keputusan bertahan

Pada keputusan ini, dilakukan langkah sebagai berikut :

(i) Apakah status *node* tetangga dari *node* yang sedang diperiksa adalah milik *player* atau netral atau milik musuh (AI). Bila status kepemilikannya adalah milik *player* atau netral, maka nilai *modifier* pada keputusan ini ditambah 1. Bila status kepemilikan adalah milik musuh (AI), maka nilai *modifier* tidak bertambah.

(ii) Bandingkan kekuatan *node* yang sedang diperiksa dengan kekuatan *node* tetangganya. Apabila kekuatan *node* yang sedang diperiksa lebih kecil kekuatannya dibandingkan dengan *node* tetangganya maka nilai *modifier* pada keputusan ini ditambah 1. Jika kekuatan *node* yang sedang diperiksa lebih besar kekuatannya dibandingkan dengan *node* tetangganya maka nilai *modifier* pada keputusan ini tidak ditambahkan.

(iii) Periksa stok unit milik AI, jika AI memiliki stok unit, maka nilai *modifier* pada keputusan ini ditambah 1. Jika AI tidak memiliki

stok unit, maka nilai *modifier* pada keputusan ini tidak ditambahkan.

(iv) Nilai *modifier* akhir untuk keputusan ini adalah jumlah dari nilai *modifier* pada kondisi yang cocok dibagi dengan jumlah seluruh kondisi yaitu 5.

III. Keputusan memproduksi unit

Pada keputusan ini, dilakukan langkah sebagai berikut :

(i) Apakah status *node* tetangga dari *node* yang sedang diperiksa adalah milik *player* atau netral atau milik musuh (AI). Bila status kepemilikannya adalah milik musuh (AI) atau netral, maka nilai *modifier* pada keputusan ini ditambah 1. Bila status kepemilikan adalah milik *player*, maka nilai *modifier* tidak bertambah.

(ii) Bandingkan kekuatan *node* yang sedang diperiksa dengan kekuatan *node* tetangganya. Apabila kekuatan *node* yang sedang diperiksa lebih besar kekuatannya dibandingkan dengan *node* tetangganya maka nilai *modifier* pada keputusan ini ditambah 1. Jika kekuatan *node* yang sedang diperiksa lebih kecil kekuatannya dibandingkan dengan *node* tetangganya maka nilai *modifier* pada keputusan ini tidak ditambahkan.

(iii) Periksa stok unit milik AI, jika AI tidak memiliki stok unit, maka nilai *modifier* pada keputusan ini ditambah 1. Jika AI memiliki stok unit, maka nilai *modifier* pada keputusan ini tidak ditambahkan.

(iv) Periksa jumlah *resource* (julah dana untuk memproduksi unit) milik AI. Jika *resource* yang dimiliki lebih dari 30 maka nilai *modifier* pada keputusan ini ditambah 1. Jika *resource* yang dimiliki kurang dari 30 maka nilai *modifier* pada keputusan ini tidak ditambahkan.

(v) Nilai *modifier* akhir untuk keputusan ini adalah jumlah dari nilai *modifier* pada kondisi yang cocok dibagi dengan jumlah seluruh kondisi yaitu 6.

IV. Keputusan melewatkan giliran

Pada keputusan ini, dilakukan langkah sebagai berikut :

(i) Apakah status *node* tetangga dari *node* yang sedang diperiksa adalah milik *player* atau netral atau milik musuh (AI). Bila status kepemilikannya adalah milik musuh (AI) atau netral, maka nilai *modifier* pada keputusan ini ditambah 1. Bila status kepemilikan adalah milik *player*, maka nilai *modifier* tidak bertambah.

(ii) Bandingkan kekuatan *node* yang sedang diperiksa dengan kekuatan *node* tetangganya. Apabila kekuatan *node* yang sedang diperiksa lebih besar kekuatannya dibandingkan dengan *node* tetangganya maka nilai *modifier* pada keputusan ini ditambah 1. Jika kekuatan *node* yang sedang diperiksa lebih kecil kekuatannya dibandingkan dengan *node* tetangganya maka nilai *modifier* pada keputusan ini tidak ditambahkan.

(iii) Periksa stok unit milik AI, jika AI tidak memiliki stok unit, maka nilai *modifier* pada keputusan ini ditambah 1. Jika AI memiliki stok unit, maka nilai *modifier* pada keputusan ini tidak ditambahkan.

(iv) Periksa jumlah *resource* (julah dana untuk memproduksi unit) milik AI. Jika *resource* yang dimiliki kurang dari 30 maka nilai *modifier* pada keputusan ini ditambah 1. Jika *resource* yang dimiliki lebih dari 30 maka nilai *modifier* pada keputusan ini tidak ditambahkan.

(v) Nilai *modifier* akhir untuk keputusan ini adalah jumlah dari nilai *modifier* pada kondisi yang cocok dibagi dengan jumlah seluruh kondisi yaitu 6.

3. Sort Assignments According to Score

Proses yang merupakan tahap mengurutkan nilai *modifier* yang didapatkan dari tahap *Generate All Possible Assignments* sehingga didapat nilai *modifier* yang terbesar. Proses ini terjadi pada pengurutan nilai *modifier* pada setiap keputusan untuk menentukan nilai *modifier* untuk sebuah *node*, serta pada

proses pengurutan nilai *modifier* pada seluruh *node* sehingga didapat *node* dengan nilai *modifier* terbesar.

4. *Assign to Task Doers*

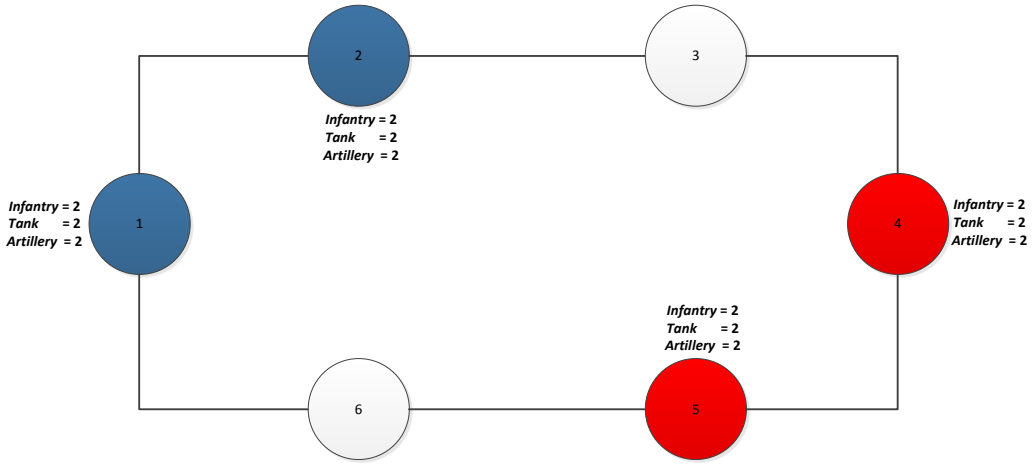
   Proses menjalankan keputusan yang didapatkan berdasarkan hasil pengurutan nilai *modifier* pada tahap *Sort Assignments According to Score* dengan menyeleksi nilai keputusan yang terdapat pada sebuah *node*. Jika nilai keputusan adalah 1, maka keputusan yang diambil adalah menyerang. Jika nilai keputusan adalah 2, maka keputusan yang diambil adalah bertahan. Jika nilai keputusan adalah 3, maka keputusan yang diambil adalah memproduksi unit. Jika nilai keputusan adalah 4, maka keputusan yang diambil adalah melewatkan giliran.

III.1.3.2. Kasus Algoritma Resource Assignment

Salah satu kasus yang akan dijalankan adalah kasus ketika permainan pada stage 1. Pada suatu kondisi ketika permainan sedang berjalan yang terjadi pada sebuah giliran (*turn*), didapatkan keadaan awal (*initial state*) *node* pada sebuah giliran di permainan yang dapat dilihat pada Tabel III-7. Status kepemilikan *node*, direpresentasikan menggunakan angka yaitu 1 untuk *node* yang dimiliki oleh *player*, 2 untuk *node* yang dimiliki oleh AI dan 0 untuk *node* netral. Pada Gambar III-5, Gambar III-6 dan Gambar III-8, *node* milik *player* direpresentasikan dengan warna biru, *node* netral direpresentasikan dengan warna putih dan *node* milik AI direpresentasikan dengan warna merah.

Tabel III-7 Kondisi *Node* Pada Contoh Kasus

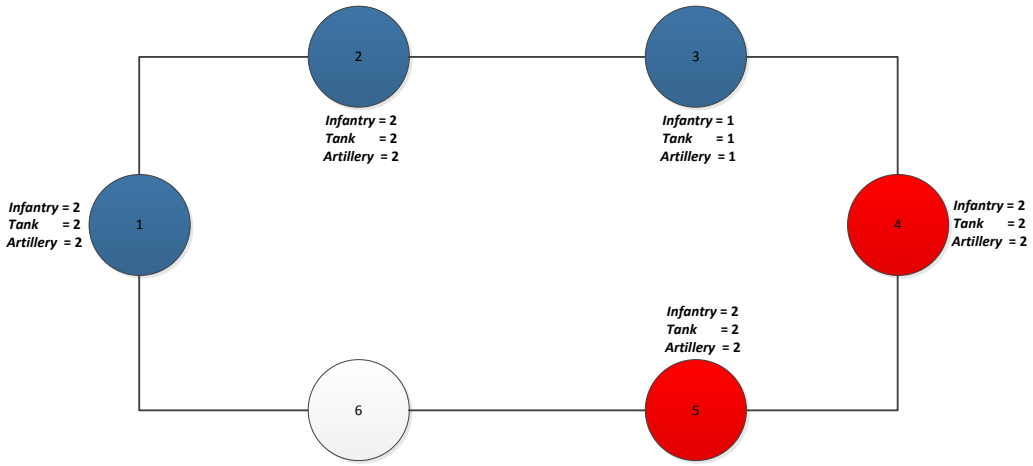
Node	Status	*Infantry*	*Tank*	*Artillery*
1	1	2	2	2
2	1	2	2	2
3	0	0	0	0
4	2	2	2	2
5	2	2	2	2
6	0	0	0	0



Gambar III-5 Kondisi Pada Contoh Kasus

Deskripsi alur permainan :

c. *Player* memeutuskan untuk menguasai *node* 3 dan menempatkan satu *infantry*, satu *tank* dan satu *artillery*. Kondisi dapat dilihat pada Gambar III-6.



Gambar III-6 Keadaan Setelah *Player* Mengambil Keputusan

d. Setelah *player* menyudahi gilirannya, maka giliran AI untuk menentukan langkah dalam mengambil keputusan. Langkah-langkah yang dikerjakan oleh AI adalah sebagai berikut:

   a. Periksa *node* 1. Karena staus *node* 1 adalah 1 (milik *player*), maka diberikan nilai *modifier* 0 dan nilai keputusan -1.

Node	Modifier	Keputusan

1	0	-1

b. Selanjutnya adalah pemeriksaan *node* 2. Karena status *node* 2 adalah 1 (milik *player*), maka nilai modifier untuk *node* 2 adalah 0 dan nilai keputusan untuk *node* 2 adalah -1.

Node	Modifier	Keputusan
1	0	-1
2	0	-1

c. Selanjutnya adalah pemeriksaan *node* 3. Karena status *node* 3 adalah 1 (milik *player*), maka nilai modifier untuk *node* 3 adalah 0 dan nilai keputusan untuk *node* 3 adalah -1.

Node	Modifier	Keputusan
1	0	-1
2	0	-1
3	0	-1

d. Selanjutnya adalah pemeriksaan *node* 4. Karena status *node* 4 adalah 2, maka dilakukan perhitungan untuk setiap keputusan yaitu :

V. Keputusan menyerang

Pada keputusan ini, dilakukan langkah sebagai berikut :

(i) Apakah status *node* tetangga pertama (*node* 3) dari *node* 4 adalah 0, 1 atau 2. Bila statusnya 0 dan 1, maka nilai *modifier* ditambah 1 dan lakukan pemeriksaan kekuatan dari *node* 3 dan *node* 4. Bila status bernilai 2, maka nilai *modifier* tidak bertambah dan lakukan pemeriksaan kekuatan dari *node* 3 dan *node* 4. Pada kondisi ini, status *node* tetangga pertama adalah 1, maka nilai *modifier* ditambahkan dengan 1 dan dilanjutkan dengan pemeriksaan kekuatan pada *node*.

Modifier = modifier + 1	Modifier = 0 + 1 = 1

(ii) Pada langkah ini, apabila kekuatan dari *node* 4 lebih besar dari *node* 3, maka nilai *modifier* ditambah 1 dan kemudian membandingakan jumlah unit di *node 3* dengan jumlah stok unit AI, bila tidak langkah selanjutnya membandingakan jumlah unit di *node 3* dengan jumlah stok unit AI. Karena pada langkah ini kekuatan pada *node* 4 lebih besar dibandingkan *node* 3, maka nilai *modifier* ditambah 1 dan dilakukan langkah selanjutnya yaitu membandingakan jumlah unit di *node 3* dengan jumlah stok unit AI. Stok untit AI, dapat dilihat pada gambar Gambar III-7.

Modifier = modifier + 1	Modifier = 1 + 1 = 2

***Infantry* = 10**
***Tank* = 10**
***Artillery* = 10**
Resource = 110

Gambar III-7 Stok Unit dan Resource AI

(iii) Apabila stok unit pada AI lebih besar dari jumlah unit di *node 3*, maka nilai *modifier* ditambahkan dengan 1 dan lakukan pemeriksaan *node* lain yang bertetangga dengan *node* 4. Bila tidak, periksa *node* tetangga yang lain dari *node* 4 tanpa menambahkan nilai *modifier*. Pada kondisi ini, unit pada *node* 3 memiliki jumlah yang lebih sedikit dibandingkan stok unit AI, maka nilai modifier ditambah 1 dan lakukan pemeriksaan terhadap *node* lain yang bertetangga dengan *node* 4, yaitu *node* 5.

Modifier = modifier + 1	Modifier = 2 + 1 = 3

(iv) Pada kondisi ini, status dari *node 5* adalah 2. Karena status *node* adalah 2, makan nilai *modifier* tidak ditambah. Setelah itu, lakukan pemeriksaan kekuatan antara *node* 5 dan *node* 2.

Modifier = modifier	Modifier = 3

(v) Karena kekuatan pada *node* 5 dan *node* 2 sama, langkah selanjutnya adalah membandingkan stok unit pada *node* 5 dengan stok unit AI. Pada langkah ini, stok unit AI (dapat dilihat pada Gambar III-7 Stok Unit dan Resource AI) lebih besar dari *node* 5, maka nilai *modifier* ditambah 1.

Modifier = modifier + 1	Modifier = 3 + 1 = 4

(vi) Karena tidak ada *node* lain lagi yang bertetangga dengan *node* 4, langkah selanjutnya adalah menghitung nilai akhir *modifier* keputusan menyerang, yaitu nilai *modifier* dibagi dengan jumlah kondisi yaitu 6.

Langkah Mendapatkan Nilai *Modifier*	Jumlah *Modifier*
Langkah ke – (i)	0 + 1 = 1
Langkah ke – (ii)	1 + 1 = 2
Langkah ke – (iii)	2 + 1 = 3
Langkah ke – (v)	3 + 1 = 4

Tabel III-8 Perhitungan *modifier* perhitungan nilai keputusan menyerang.

4 / 6 = 0.67

Keputusan	Nilai Akhir *Modifier*
1	0.67

VI. Keputusan bertahan

Pada keputusan ini, dilakukan langkah sebagai berikut :

(i) Apakah status *node* tetangga pertama (*node* 3) dari *node* 4 adalah 0, 1 atau 2. Bila statusnya 0 dan 1, maka nilai *modifier* ditambah 1 dan lakukan pemeriksaan kekuatan dari *node* 3 dan *node* 4. Bila status bernilai 2, maka nilai *modifier* tidak bertambah dan lakukan pemeriksaan kekuatan dari *node* 3 dan *node* 4. Pada kondisi ini, status

node tetangga pertama adalah 1, maka nilai *modifier* ditambahkan dengan 1 dan dilanjutkan dengan pemeriksaan kekuatan pada *node*.

Modifier = modifier + 1	Modifier = 0 + 1 = 1

(ii) Pada langkah ini, apabila kekuatan dari *node* 4 lebih kecil dari *node* 3, maka nilai *modifier* ditambah 1 dan lakukan pemeriksaan *node* lain yang bertetangga dengan *node* 4. Bila tidak, periksa *node* tetangga yang lain dari *node* 4 tanpa menambahkan nilai *modifier*. Pada kondisi ini, kekuatan *node* 4 memiliki lebih besar dibandingkan dengan *node* 3, maka nilai *modifier* tidak bertambah dan lakukan pemeriksaan terhadap *node* lain yang bertetangga dengan *node* 4, yaitu *node* 5.

Modifier = modifier	Modifier = 1

(iii) Pada kondisi ini, status dari *node 5* adalah 2. Karena status *node* adalah 2, maka nilai *modifier* tidak ditambah. Kemudian, lakukan pemeriksaan kekuatan antara *node* 5 dan *node* 4.

Modifier = modifier	Modifier = 1

(iv) Karena kekuatan pada *node* 5 dan *node* 4 sama, maka nilai *modifier* tidak bertambah.

Modifier = modifier	Modifier = 1

(v) Karena tidak ada *node* lain lagi yang bertetangga dengan *node* 4, langkah selanjutnya adalah memeriksa stok unit AI, apakah stok unit AI lebih dari 0 atau tidak. Jika lebih dari 0, nilai *modifier* ditambah 1. Jika tidak, nilai *modifier* tidak bertambah. Pada kondisi ini stok unit AI lebih dari 0, maka nilai *modifier* ditambah 1. Setelah itu, hitung nilai *modifier* akhir untuk keputusan bertahan, yaitu nilai *modifier* dibagi dengan jumlah kondisi yaitu 5.

Modifier = modifier	Modifier = 1

Langkah Mendapatkan Nilai *Modifier*	Jumlah *Modifier*
Langkah ke – (i)	0 + 1 = 1
Langkah ke – (v)	1 + 1 = 2

Tabel III-9 Perhitungan nilai *modifier* untuk keputusan bertahan.

2 / 5 = 0.40

Keputusan	Nilai Akhir *Modifier*
2	0.40

VII. Keputusan memproduksi unit

Pada keputusan ini, dilakukan langkah sebagai berikut :

(i) Apakah status *node* tetangga pertama (*node* 3) dari *node* 4 adalah 1 atau tidak . Bila statusnya 1, maka nilai *modifier* tidak ditambah 1, namun bila tidak, maka nilai *modifier* bertambah dan lakukan pemeriksaan kekuatan dari *node* 3 dan *node* 4. Pada kondisi ini, status *node* tetangga pertama adalah 1, maka nilai *modifier* tidak bertambah dan dilanjutkan dengan pemeriksaan kekuatan pada *node.*

Modifier = modifier + 1	Modifier = 0 + 1 = 1

(ii) Pada langkah ini, apabila kekuatan dari *node* 4 lebih besar dari *node* 3, maka nilai *modifier* ditambah 1 dan lakukan pemeriksaan *node* lain yang bertetangga dengan *node* 4. Bila tidak, periksa *node* tetangga yang lain dari *node* 4 tanpa menambahkan nilai *modifier*. Pada kondisi ini, kekuatan pada *node* 4 lebih besar dibandingkan dengan *node* 3, maka nilai *modifier* bertambah 1 dan lakukan pemeriksaan terhadap *node* lain yang bertetangga dengan *node* 4, yaitu *node* 5.

Modifier = modifier + 1	Modifier = 1 + 1 = 2

(iii) Pada kondisi ini, status dari *node 5* adalah 2. Karena status *node* adalah 2, maka nilai *modifier* ditambah 1. Kemudian, lakukan pemeriksaan kekuatan antara *node* 5 dan *node* 4.

Modifier = modifier + 1	Modifier = 2 + 1 = 3

(iv) Karena kekuatan pada *node* 5 dan *node* 4 sama, maka nilai *modifier* tidak bertambah.

Modifier = modifier	Modifier = 3

(v) Karena tidak ada *node* lain lagi yang bertetangga dengan *node* 4, langkah selanjutnya adalah memeriksa stok unit AI, apakah stok unit AI lebih kecil dari 0 atau tidak. Jika lebih kecil dari 0, maka nilai *modifier* ditambah 1. Jika tidak, nilai *modifier* tidak bertambah. Pada kondisi ini stok unit AI lebih dari 0, maka nilai *modifier* tidak bertambah.

Modifier = modifier	Modifier = 3

(vi) Setelah itu, periksa apakah *resource* yang dimiliki lebih dari sama dengan 30 atau tidak. Bila *resource* lebih dari sama dengan 30, maka nilai ditambah 1 dan dilanjutkan ke langkah selanjutnya yaitu menghitung nilai akhir *modifier*. Bila tidak, nilai *modifier* tidak bertambah dan hitung nilai akhir *modifier*. Pada kondisi ini, *resource* yang dimiliki AI lebih dari 30, yaitu 110. Maka nilai *modifier* tidak bertambah. Setelah itu hitung nilai akhir nilai *modifier* keputusan membangun unit.

Modifier = modifier	Modifier = 3

Langkah Mendapatkan Nilai *Modifier*	Jumlah *Modifier*
Langkah ke – (i)	0 + 1 = 1
Langkah ke – (iii)	1 + 1 = 2
Langkah ke – (vi)	2+ 1 = 3

Tabel III-10 Perhitungan nilai *modifier* untuk keputusan bertahan.

3 / 6 = 0.50

Keputusan	Nilai Akhir *Modifier*
3	0.50

VIII. Keputusan melewatkan giliran

Pada keputusan ini, dilakukan langkah sebagai berikut :

(i) Apakah status *node* tetangga pertama (*node* 3) dari *node* 4 adalah 1 atau tidak . Bila statusnya 1, maka nilai *modifier* tidak ditambah 1, namun bila tidak, maka nilai *modifier* bertambah dan lakukan pemeriksaan kekuatan dari *node* 3 dan *node* 4. Pada kondisi ini, status *node* tetangga pertama adalah 1, maka nilai *modifier* tidak bertambah dan dilanjutkan dengan pemeriksaan kekuatan pada *node.*

Modifier = modifier	Modifier = 0

(ii) Pada langkah ini, apabila kekuatan dari *node* 4 lebih besar dari *node* 3, maka nilai *modifier* ditambah 1 dan lakukan pemeriksaan *node* lain yang bertetangga dengan *node* 4. Bila tidak, periksa *node* tetangga yang lain dari *node* 4 tanpa menambahkan nilai *modifier*. Pada kondisi ini, kekuatan pada *node* 4 lebih besar dibandingkan dengan *node* 3, maka nilai *modifier* bertambah 1 dan lakukan pemeriksaan terhadap *node* lain yang bertetangga dengan *node* 4, yaitu *node* 5.

Modifier = modifier + 1	Modifier = 0 + 1 = 1

(iii) Pada kondisi ini, status dari *node 5* adalah 2. Karena status *node* adalah 2, maka nilai *modifier* ditambah 1. Kemudian, lakukan pemeriksaan kekuatan antara *node* 5 dan *node* 4.

Modifier = modifier + 1	Modifier = 1 + 1 = 2

(iv) Karena kekuatan pada *node* 5 dan *node* 4 sama, maka nilai *modifier* tidak bertambah.

Modifier = modifier	Modifier = 2

(v) Karena tidak ada *node* lain lagi yang bertetangga dengan *node* 4, langkah selanjutnya adalah memeriksa stok unit AI, apakah stok unit AI lebih kecil dari 0 atau tidak. Jika lebih kecil dari 0, maka nilai *modifier* ditambah 1. Jika tidak, nilai *modifier* tidak bertambah. Pada kondisi ini stok unit AI lebih dari 0, maka nilai *modifier* tidak bertambah.

Modifier = modifier	Modifier = 2

(vi) Setelah itu, periksa apakah *resource* yang dimiliki lebih dari sama dengan 30 atau tidak. Bila *resource* lebih dari sama dengan 30, maka nilai *modifier* tidak bertambah dan dilanjutkan ke langkah selanjutnya yaitu menghitung nilai akhir *modifier*. Bila tidak, nilai *modifier* ditambah 1, kemudian menghitung nilai akhir *modifier*. Pada kondisi ini, *resource* yang dimiliki AI lebih dari 30, yaitu 110. Maka nilai *modifier* tidak bertambah. Setelah itu hitung nilai akhir nilai *modifier* keputusan membangun unit.

Modifier = modifier	Modifier = 2

Langkah Mendapatkan Nilai *Modifier*	Jumlah *Modifier*
Langkah ke – (i)	0 + 1 = 1
Langkah ke – (iii)	1 + 1 = 2

Tabel III-11 Perhitungan nilai *modifier* untuk keputusan melewatkan giliran.

$2 / 6 = 0.33$

Keputusan	Nilai Akhir *Modifier*
3	0.33
Keputusan	Modifier
1	0.67

2	0.40
3	0,50
4	0,33

Langkah selanjutnya adalah mengurutkan nilai modifier tersebut secara menaik, maka hasilnya adalah :

Keputusan	Modifier
4	0,33
2	0,40
3	0,50
1	0.67

Dengan demikian nilai *modifier* pada *node* 4 adalah nilai yang terbesar yaitu 0.67 dengan nilai keputusan adalah 1.

Node	Modifier	Keputusan
1	0	-1
2	0	-1
3	0	-1
4	0.67	1

e. Selanjutnya adalah pemeriksaan *node* tetangga dari *node* 5 karena status *node* 5 adalah 2. Dilakukan perhitungan untuk setiap keputusan yaitu :

I. Keputusan menyerang

Pada keputusan ini, dilakukan langkah sebagai berikut :

(i) Apakah status *node* tetangga pertama (*node* 4) dari *node* 5 adalah 0, 1 atau 2. Bila statusnya 0 dan 1, maka nilai *modifier* ditambah 1 dan lakukan pemeriksaan kekuatan dari *node* 5 dan *node* 4. Bila status bernilai 2, maka nilai *modifier* tidak bertambah dan lakukan pemeriksaan kekuatan dari *node* 5 dan *node* 4. Pada kondisi ini, status

node tetangga pertama adalah 2, maka nilai tidak bertambah dan dilanjutkan dengan pemeriksaan kekuatan pada *node.*

Modifier = modifier	Modifier = 0

(ii) Pada langkah ini, apabila kekuatan dari *node* 5 lebih besar dari *node* 4, maka nilai *modifier* ditambah 1 dan kemudian membandingakan jumlah unit di *node* 4 dengan jumlah stok unit AI, bila tidak langkah selanjutnya membandingakan jumlah unit di *node* 4 dengan jumlah stok unit AI. Karena pada langkah ini kekuatan pada *node* 5 sama dengan *node* 4, maka nilai *modifier* tidak bertambah dan dilakukan langkah selanjutnya yaitu membandingakan jumlah unit di *node 3* dengan jumlah stok unit AI.

Modifier = modifier	Modifier = 0

(iii) Apabila stok unit pada AI lebih besar dari jumlah unit di *node 4*, maka nilai *modifier* ditambahkan dengan 1 dan lakukan pemeriksaan *node* lain yang bertetangga dengan *node* 5. Bila tidak, periksa *node* tetangga yang lain dari *node* 5 tanpa menambahkan nilai *modifier*. Pada kondisi ini, unit pada *node* 4 memiliki jumlah yang lebih sedikit dibandingkan dengan stok unit AI, maka nilai *modifier* tidak ditambah dan lakukan pemeriksaan terhadap *node* lain yang bertetangga dengan *node* 5, yaitu *node* 6.

Modifier = modifier + 1	Modifier = 0 + 1 = 1

(iv) Pada kondisi ini, status dari *node 6* adalah 0 (netral). Karena status *node* adalah 0, makan nilai *modifier* tidak ditambah. Setelah itu, lakukan pemeriksaan kekuatan antara *node* 5 dan *node* 6.

Modifier = modifier	Modifier = 1

(v) Karena kekuatan pada *node* 5 lebih besar dari *node* 6, maka nilai *modifier* ditambah 1 dan langkah selanjutnya adalah membandingkan stok unit pada *node* 6 dengan stok unit AI.

Modifier = modifier + 1	Modifier = 1 + 1 = 2

(vi) Pada langkah ini, stok unit AI (dapat dilihat pada Gambar III-7 Stok Unit dan Resource AI) lebih besar dari *node* 6, maka nilai *modifier* ditambah 1.

Modifier = modifier + 1	Modifier = 2 + 1 = 3

(vii) Karena tidak ada *node* lain lagi yang bertetangga dengan *node* 5, langkah selanjutnya adalah menghitung nilai akhir *modifier* keputusan menyerang, yaitu nilai *modifier* dibagi dengan jumlah kondisi yaitu 6.

Langkah Mendapatkan Nilai *Modifier*	Jumlah *Modifier*
Langkah ke – (iii)	0 + 1 = 1
Langkah ke – (v)	1 + 1 = 2
Langkah ke – (vi)	2 + 1 = 3

Tabel III. 12 Perhitungan nilai *modifier* untuk keputusan menyerang

3 / 6 = 0.50

Keputusan	Nilai Akhir *Modifier*
1	0.50

II. Keputusan bertahan

Pada keputusan ini, dilakukan langkah sebagai berikut :

(i) Apakah status *node* tetangga pertama (*node* 4) dari *node* 5 adalah 0, 1 atau 2. Bila statusnya 0 dan 1, maka nilai *modifier* ditambah 1 dan lakukan pemeriksaan kekuatan dari *node* 5 dan *node* 4. Bila status bernilai 2, maka nilai *modifier* tidak bertambah dan lakukan pemeriksaan kekuatan dari *node* 5 dan *node* 4. Pada kondisi ini, status

node tetangga pertama adalah 2, maka nilai *modifier* tidak bertambah dan dilanjutkan dengan pemeriksaan kekuatan pada *node.*

Modifier = modifier	Modifier = 0

(ii) Pada langkah ini, apabila kekuatan dari *node* 5 lebih kecil dari *node* 4, maka nilai *modifier* ditambah 1 dan lakukan pemeriksaan *node* lain yang bertetangga dengan *node* 5. Bila tidak, periksa *node* tetangga yang lain dari *node* 4 tanpa menambahkan nilai *modifier*. Pada kondisi ini, kekuatan *node* 4 sama dengan *node* 5, maka nilai *modifier* tidak bertambah dan lakukan pemeriksaan terhadap *node* lain yang bertetangga dengan *node* 5, yaitu *node* 6.

Modifier = modifier	Modifier = 0

(iii) Pada kondisi ini, status dari *node* 6 adalah 0. Karena status *node* adalah 0, maka nilai *modifier* bertambah 1. Kemudian, lakukan pemeriksaan kekuatan antara *node* 5 dan *node* 6.

Modifier = modifier + 1	Modifier = 0 + 1 = 1

(iv) Karena kekuatan pada *node* 5 lebih besar dari *node* 6, maka nilai *modifier* bertambah 1.

Modifier = modifier + 1	Modifier = 1 + 1 = 2

(v) Karena tidak ada *node* lain lagi yang bertetangga dengan *node* 5, langkah selanjutnya adalah memeriksa stok unit AI, apakah stok unit AI lebih dari 0 atau tidak. Jika lebih dari 0, nilai *modifier* ditambah 1. Jika tidak, nilai *modifier* tidak bertambah. Pada kondisi ini stok unit AI lebih dari 0, maka nilai *modifier* ditambah 1. Setelah itu, hitung nilai *modifier* akhir untuk keputusan bertahan, yaitu nilai *modifier* dibagi dengan jumlah kondisi yaitu 5.

Modifier = modifier + 1	Modifier = 2 + 1 = 3

Langkah Mendapatkan Nilai *Modifier*	Jumlah *Modifier*
Langkah ke – (iii)	0 + 1 = 1
Langkah ke – (iv)	1 + 1 = 2
Langkah ke – (v)	2 + 1 = 3

Tabel III-13 Perhitungan nilai *modifier* untuk keputusan bertahan

3 / 5 = 0.60

Keputusan	Nilai Akhir *Modifier*
2	0.60

III. Keputusan memproduksi unit

Pada keputusan ini, dilakukan langkah sebagai berikut :

(i) Apakah status *node* tetangga pertama (*node* 4) dari *node* 5 adalah 1 atau tidak . Bila statusnya 1, maka nilai *modifier* tidak ditambah 1, namun bila tidak, maka nilai *modifier* bertambah dan lakukan pemeriksaan kekuatan dari *node* 5 dan *node* 4. Pada kondisi ini, status *node* tetangga pertama adalah 2 atau tidak sama dengan 1, maka nilai *modifier* bertambah 1 dan dilanjutkan dengan pemeriksaan kekuatan pada *node.*

Modifier = modifier + 1	Modifier = 0 + 1 = 1

(ii) Pada langkah ini, apabila kekuatan dari *node* 5 lebih besar dari *node* 4, maka nilai *modifier* ditambah 1 dan lakukan pemeriksaan *node* lain yang bertetangga dengan *node* 5. Bila tidak, periksa *node* tetangga yang lain dari *node* 5 tanpa menambahkan nilai *modifier*. Pada kondisi ini, kekuatan pada *node* 5 sama dengan *node* 4, maka nilai *modifier* tidak bertambah dan lakukan pemeriksaan terhadap *node* lain yang bertetangga dengan *node* 5, yaitu *node 6*.

Modifier = modifier	Modifier = 1

(iii) Pada kondisi ini, status dari *node 6* adalah 0. Karena status *node* adalah 0, maka nilai *modifier* tidak bertambah. Lakukan pemeriksaan kekuatan antara *node* 5 dan *node* 6.

Modifier = modifier	Modifier = 1

(iv) Karena kekuatan pada *node* 5 lebih besar dari *node* 6, maka nilai *modifier* bertambah 1.

Modifier = modifier + 1	Modifier = 1 + 1 = 2

(v) Karena tidak ada *node* lain lagi yang bertetangga dengan *node* 5, langkah selanjutnya adalah memeriksa stok unit AI, apakah stok unit AI lebih kecil dari 0 atau tidak. Jika lebih kecil dari 0, maka nilai *modifier* ditambah 1. Jika tidak, nilai *modifier* tidak bertambah. Pada kondisi ini stok unit AI lebih dari 0, maka nilai *modifier* tidak bertambah.

Modifier = modifier	Modifier = 2

(vi) Setelah itu, periksa apakah *resource* yang dimiliki lebih dari sama dengan 30 atau tidak. Bila *resource* lebih dari sama dengan 30, maka nilai ditambah 1 dan dilanjutkan ke langkah selanjutnya yaitu menghitung nilai akhir *modifier*. Bila tidak, nilai *modifier* tidak bertambah dan hitung nilai akhir *modifier*. Pada kondisi ini, *resource* yang dimiliki AI lebih dari 30, yaitu 110. Maka nilai *modifier* bertambah 1. Setelah itu hitung nilai akhir nilai *modifier* keputusan membangun unit.

Modifier = modifier + 1	Modifier = 2 + 1 = 3

Langkah Mendapatkan Nilai *Modifier*	Jumlah *Modifier*
Langkah ke – (i)	0 + 1 = 1
Langkah ke – (iv)	1 + 1 = 2

Langkah ke – (vii)	2 + 1 = 3

Tabel III-14 Perhitungan nilai *modifier* untuk keputusan memproduksi unit

3 / 6 = 0.50

Keputusan	Nilai Akhir *Modifier*
3	0.50

IV. Keputusan melewatkan giliran

Pada keputusan ini, dilakukan langkah sebagai berikut :

(i) Apakah status *node* tetangga pertama (*node* 4) dari *node* 5 adalah 1 atau tidak . Bila statusnya 1, maka nilai *modifier* tidak ditambah 1, namun bila tidak, maka nilai *modifier* bertambah dan lakukan pemeriksaan kekuatan dari *node* 5 dan *node* 4. Pada kondisi ini, status *node* tetangga pertama adalah 2, maka nilai ditambah 1 dan dilanjutkan dengan pemeriksaan kekuatan pada *node.*

Modifier = modifier + 1	Modifier = 0 + 1 = 1

(ii) Pada langkah ini, apabila kekuatan dari *node* 5 lebih besar dari *node* 4, maka nilai *modifier* ditambah 1 dan lakukan pemeriksaan *node* lain yang bertetangga dengan *node* 5. Bila tidak, periksa *node* tetangga yang lain dari *node* 4 tanpa menambahkan nilai *modifier*. Pada kondisi ini, kekuatan pada *node* 5 sama dengan *node* 4, maka nilai *modifier* tidak bertambah dan lakukan pemeriksaan terhadap *node* lain yang bertetangga dengan *node* 5, yaitu *node* 6.

Modifier = modifier	Modifier = 1

(iii) Pada kondisi ini, status dari *node* 6 adalah 0. Karena status *node* adalah 0, maka nilai *modifier* tidak bertambah. Kemudian, lakukan pemeriksaan kekuatan antara *node* 5 dan *node* 6.

Modifier = modifier	Modifier = 1

(iv) Karena kekuatan pada *node* 5 lebih besar dari *node* 6, maka nilai *modifier* bertambah 1.

Modifier = modifier + 1	Modifier = 1 + 1 = 2

(v) Karena tidak ada *node* lain lagi yang bertetangga dengan *node* 5, langkah selanjutnya adalah memeriksa stok unit AI, apakah stok unit AI lebih kecil dari 0 atau tidak. Jika lebih kecil dari 0, maka nilai *modifier* ditambah 1. Jika tidak, nilai *modifier* tidak bertambah. Pada kondisi ini stok unit AI lebih dari 0, maka nilai *modifier* tidak bertambah.

Modifier = modifier	Modifier = 2

(vi) Setelah itu, periksa apakah *resource* yang dimiliki lebih dari sama dengan 30 atau tidak. Bila *resource* lebih dari sama dengan 30, maka nilai *modifier* tidak bertambah dan dilanjutkan ke langkah selanjutnya yaitu menghitung nilai akhir *modifier*. Bila tidak, nilai *modifier* ditambah 1, kemudian menghitung nilai akhir *modifier*. Pada kondisi ini, *resource* yang dimiliki AI lebih dari 30, yaitu 110. Maka nilai *modifier* tidak bertambah. Setelah itu hitung nilai akhir nilai *modifier* keputusan membangun unit.

Modifier = modifier	Modifier = 2

Langkah Mendapatkan Nilai *Modifier*	Jumlah *Modifier*
Langkah ke – (i)	0 + 1 = 1
Langkah ke – (iv)	1 + 1 = 2

Tabel III-15 Perhitungan nilai *modifier* untuk keputusan melewatkan giliran.

2 / 6 = 0.33

Keputusan	Nilai Akhir *Modifier*
3	0.33

Maka didapat nilai modifier untuk setiap keputusan adalah sebagai berikut :

Keputusan	Modifier
1	0.50
2	0.60
3	0,50
4	0,33

Langkah selanjutnya adalah mengurutkan nilai modifier tersebut secara menaik, maka hasilnya adalah :

Keputusan	Modifier
4	0,33
3	0,50
1	0,50
2	0.60

Dengan demikian nilai *modifier* pada *node* 6 adalah nilai yang terbesar yaitu 0.60 dengan nilai keputusan adalah 2.

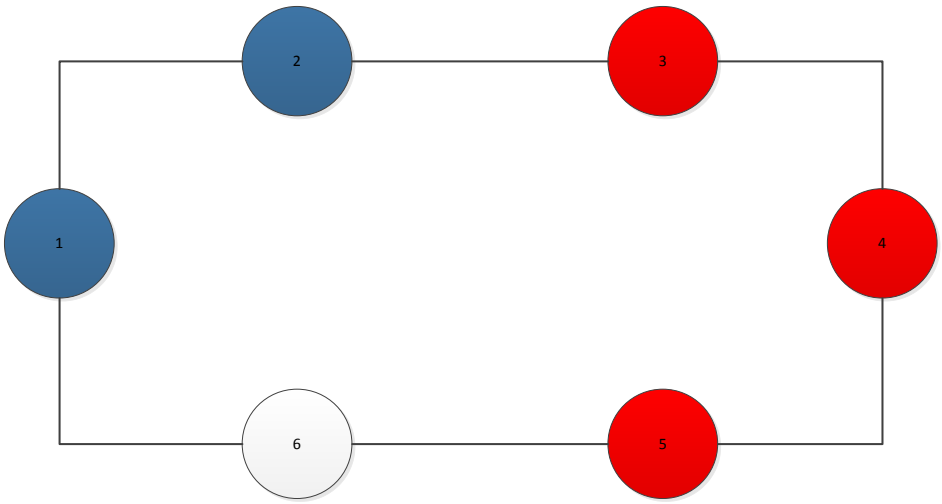
Node	Modifier	Keputusan
1	0	-1
2	0	-1
3	0	-1
4	0.67	1
5	0.60	2

f. Selanjutnya adalah pemeriksaan *node* 6. Karena status *node* 6 adalah 0 (netral), maka nilai *modifier* untuk *node* 6 adalah 0 dan nilai keputusan untuk *node* 6 adalah -1.

g.

Node	Modifier	Keputusan
1	0	-1
2	0	-1
3	0	-1
6	0	-1
5	0.60	2
4	0.67	1

Berdasarkan nilai *modifier* terbesar pada pada seluruh node, maka keputusan yang diambil oleh AI adalah menyerang dari *node* 4 dan dapat dilihat pada Gambar III-8.



Gambar III-8 Kondisi Setelah AI Menjalankan Keputusan Pada Contoh Kasus

Pada contoh kasus ini setelah AI menguasai *node* 3, permainan akan terus berlangsung sampai seluruh *node* dikuasai oleh *player* atau AI. Jika *player* berhasil menguasai seluruh *node*, maka *player* dinyatakan menang. Jika AI berhasil menguasai seluruh *node*, maka *player* dinyatakan kalah.

III.1.6. 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.1.4.1. Analisis Kebutuhan Perangkat Keras

Perangkat yang digunakan untuk memainkan game ini adalah smartphone. Berikut adalah spesifikasi smartphone yang diperlukan untuk memainkan game ini :

1. Sistem Operasi Android 2.3 keatas
2. Processor 800 MHz
3. Kapasitas RAM 128 MB
4. Kapasitas penyimpanan 20 MB

III.1.4.2. Analisis Kebutuhan Perangkat Lunak

Perangkat lunak yang dibutuhkan untuk mendukung pembangunan game ini adalah sebagai berikut:

1. Sistem Operasi Windows XP keatas
2. Eclipse
3. Phonegap Framework
4. Construct 2 R95
5. Android SDK
6. Android Virtual Device
7. Adobe Photoshop CS3

III.1.4.3. Analisis Kebutuhan Perangkat Pikir

Analisis kebutuhan perangkat pikir dilakukan untuk mengetahui kemampuan *player* (pemain) yang dapat memaikan game ini. Analisis kebutuhan perangkat pikir digambarkan dalam tabel kebutuhan *player*. Kebutuhan *player* dijabarkan pada Tabel III-16.

Tabel III-16 Tabel Kebutuhan *Player*

Karakteristik Player	Keterangan
Computer Literacy	Dapat menggunakan *smartphone* berbasis Android
Age	10 tahun keatas [HYPERLINK "" \l "ESR" **16**]
Application Experience	Tidak dibutuhkan
Gender	Pria dan Wanita
Disabilities	Buta total dan buta warna

III.1.7. Analisis Kebutuhan Fungsional

Analisis kebutuhan fungsional dilakukan untuk mengetahui gambaran sistem yang akan dibangun. Analisis kebutuhan fungsional terdiri dari pendefinisan fungsi game, penggambaran alir aktivitas pada game dan penggambaran interaksi antar objek pada game.

III.1.8. Analisis Kebutuhan Fungsional

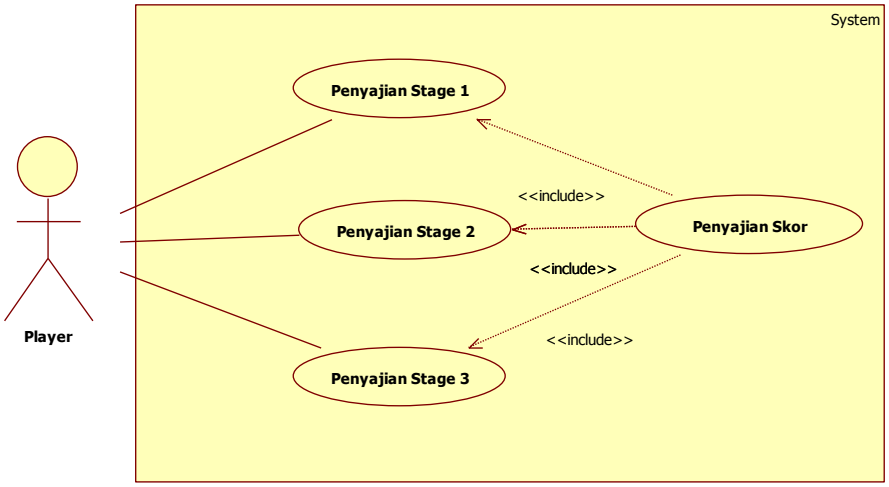
Analisis kebutuhan fungsional dilakukan untuk mengetahui gambaran sistem yang akan dibangun. Analisis kebutuhan fungsional terdiri dari pendefinisan fungsi game, penggambaran alir aktivitas pada game dan penggambaran interaksi antar objek pada game.

III.1.5.1. Pendefinisian Fungsi Game

Pendefinisian fungsi game digunakan untuk menggambarkan fungsi-fungsi pada game. Pendefinisian fungsi pada game ini digambarkan dengan menggunakan use case diagram, definisi aktor,definisi use case dan skenario use case.

1. Use Case Diagram

   Penggambaran fungsi game berdasarkan interaksi antar aktor dan objek pada sistem digambarkan dengan menggunakan diagram use case yang terdapat pada Gambar III-11.



Gambar III-11 *Use Case Diagram*

2. Definisi Aktor

Definisi aktor berfungsi menjelaskan peran setiap aktor yang terdapat pada use case. Definisi aktor ditunjukan pada Tabel III-14.

Tabel III-14 Definisi Aktor

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

3. Definisi Use Case

Definisi use case berfungsi menjelaskan tiap-tiap proses dan objek yang terdapat pada use case.

Tabel III-15 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	Penyajian petunjuk permainan	Proses menampilkan petunjuk permainan

5	Penyajian Skor	Proses menampilkan skor yang diperoleh pada permainan

4. Skenario Use Case

Skenario use case menunjukan proses apa yang terjadi pada setiap bagian di dalam use case tersebut, dimana user memberikan perintah pada bagian-bagian use case dan respon apa yang diberikan oleh sistem kepada user setelah user memberikan perintahnya. Skenario use case ditujukan pada Tabel III-16 sampai dengan Tabel III-19.

Tabel III-16 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 form *stage* 1
1. Memilih node milik musuh		2. Menjalankan proses menyerang *node*
3. Memilih node netral		4. Menjalankan proses menyerang *node*
5. Memilih node milik aktor		6. Menjalankan proses mempertahankan *node*
7. Menekan tombol produksi		8. Menjalankan proses memproduksi unit
1. Menekan tombol *end turn*		2. Menjalankan proses *end turn*
11. Menekan tombol keluar		12. Menampilkan pesan keluar dari *game*
Skenario Tambahan		
Kondisi Awal	Aktor berada di pesan keluar dari *game*	
1. Menekan tombol Ya		2. Menampilkan menu main
3. Menekan tombol Tidak		4. Menampilkan form *stage* 1

Tabel III-17 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 form *stage* 2
1. Memilih node milik musuh		2. Menjalankan proses menyerang *node*
3. Memilih node netral		4. Menjalankan proses menyerang *node*
5. Memilih node milik aktor		6. Menjalankan proses mempertahankan *node*
7. Menekan tombol produksi		8. Menjalankan proses memproduksi unit
1. Menekan tombol *end turn*		2. Menjalankan proses *end turn*
11. Menekan tombol keluar		12. Menampilkan pesan keluar dari *game*
Skenario Tambahan		
Kondisi Awal	Aktor berada di pesan keluar dari *game*	
1. Menekan tombol Ya		2. Menampilkan menu main
3. Menekan tombol Tidak		4. Menampilkan form *stage* 2

Tabel III-18 Skenario *Use Case* Penyajian *Stage* 3

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 form *stage* 3
1. Memilih node milik musuh		2. Menjalankan proses menyerang *node*

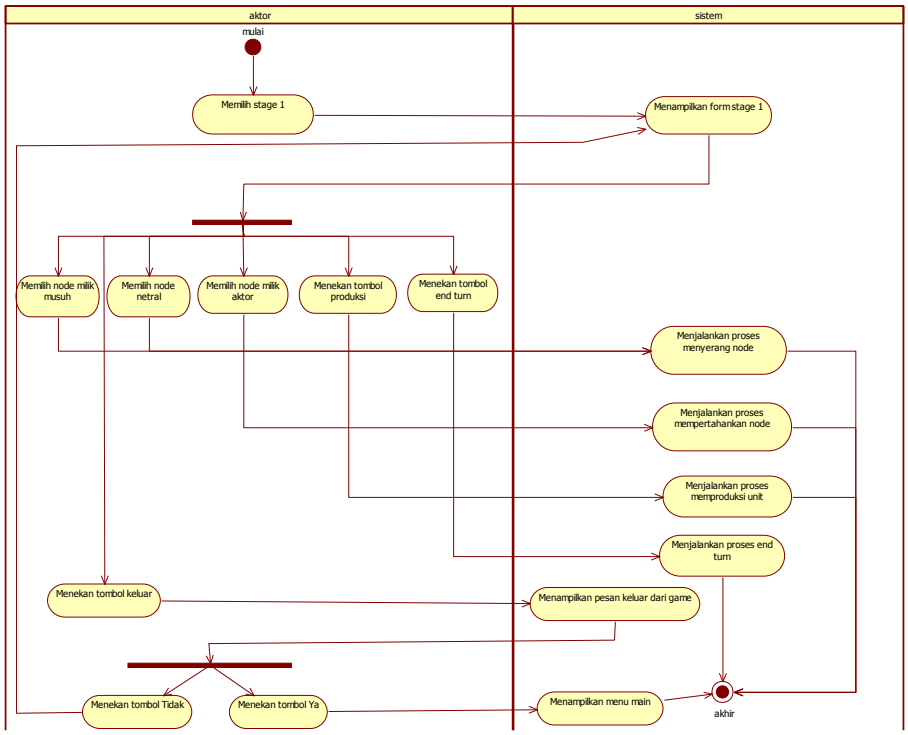
3. Memilih node netral	4. Menjalankan proses menyerang *node*
5. Memilih node milik aktor	6. Menjalankan proses mempertahankan *node*
7. Menekan tombol produksi	8. Menjalankan proses memproduksi unit
1. Menekan tombol *end turn*	2. Menjalankan proses *end turn*
11. Menekan tombol keluar	12. Menampilkan pesan keluar dari *game*
Skenario Tambahan	
Kondisi Awal	Aktor berada di pesan keluar dari *game*
1. Menekan tombol Ya	2. Menampilkan menu main
3. Menekan tombol Tidak	4. Menampilkan form *stage* 3

Tabel III-19 *Skenario Use Case* Penyajian Skor

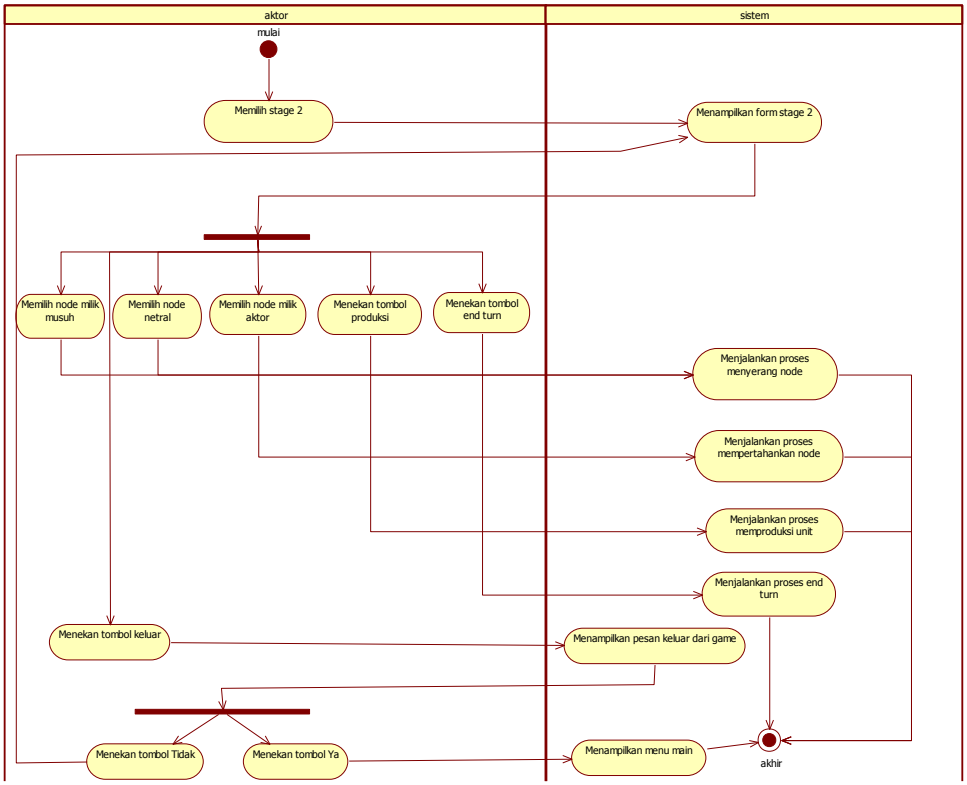
Identifikasi	
Nomor	5
Nama	Penyajian Skor
Tujuan	Menampilkan skor
Deskripsi	Proses menampilkan skor yang diperoleh
Aktor	Proses menampilkan skor yang diperoleh pada permainan
Skenario Utama	
Kondisi Awal	Aktor telah selesai menjalankan permainan
Aksi Aktor	Reaksi Sistem
1. Aktor selesai menjalankan permainan	2. Menjalankan proses perhitungan skor *infantry*
	3. Menampilkan skor *infantry*
	4. Menjalankan proses perhitungan skor *tank*
	5. Menampilkan skor *tank*
	6. Menjalankan proses perhitungan skor *artillery*
	7. Menampilkan skor *artillery*
	8. Menjalankan proses perhitungan skor *resource point*
	9. Menampilkan skor *resource point*
10. Menekan tombol keluar	11. Menampilkan form menu main

III.1.5.2. Penggambaran Alir Aktivitas Pada Game

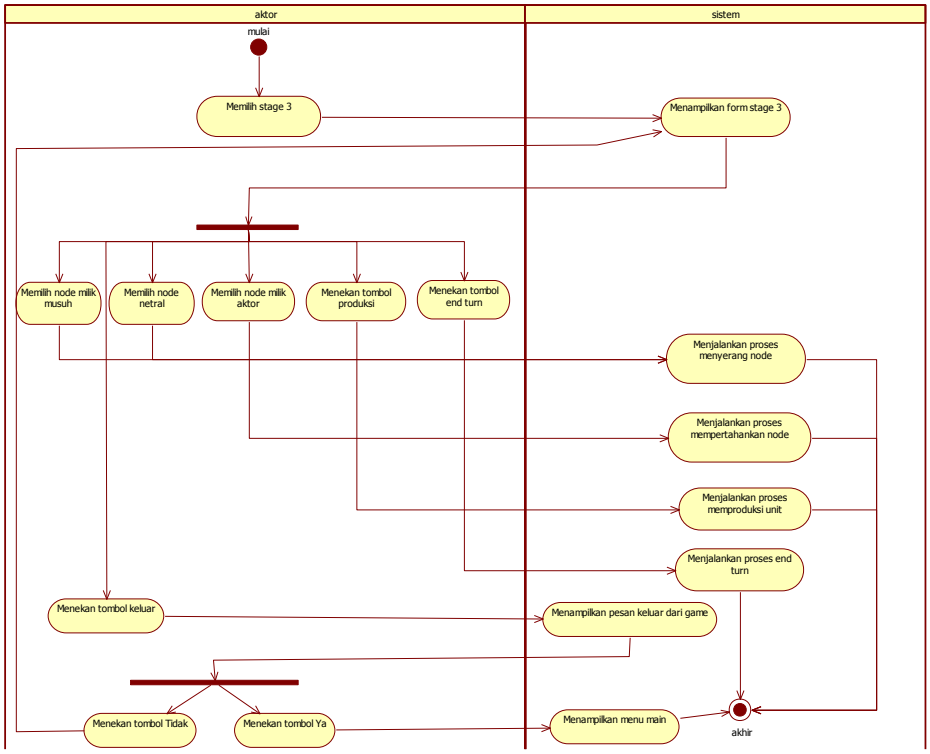
Penggambaran alir aktivitas pada game menggambarkan berbagai alir aktivitas dalam game yang sedang dirancang, bagaimana masing-masing alir berawal, *decision* yang mungkin terjadi, dan bagaimana mereka berakhir. Penggambaran alir aktivitas pada *game* digambarkan menggunakan *Activity Diagram*. *Activity 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. *Activity diagram* ditunjukan pada Gambar III-12 sampai dengan Gambar III. 15.



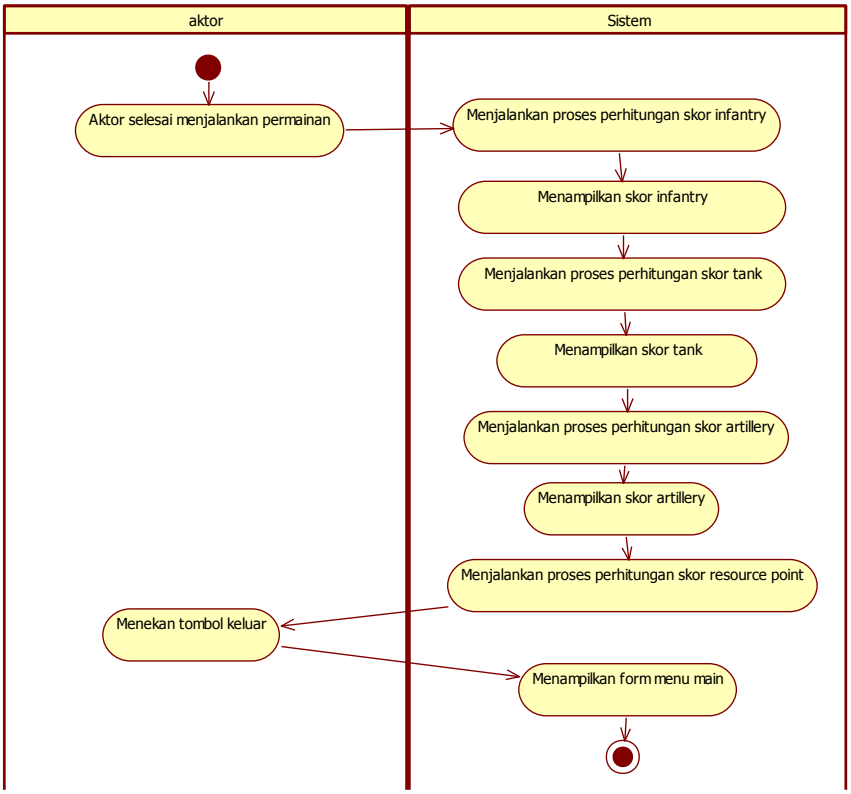
Gambar III-12 *Activity Diagram* Penyajian *Stage 1*



Gambar III-13 *Activity Diagram* Penyajian *Stage 2*



Gambar III-14 *Activity Diagram* Penyajian *Stage 3*



Gambar III. 15 *Activity Diagram* Penyajian Skor

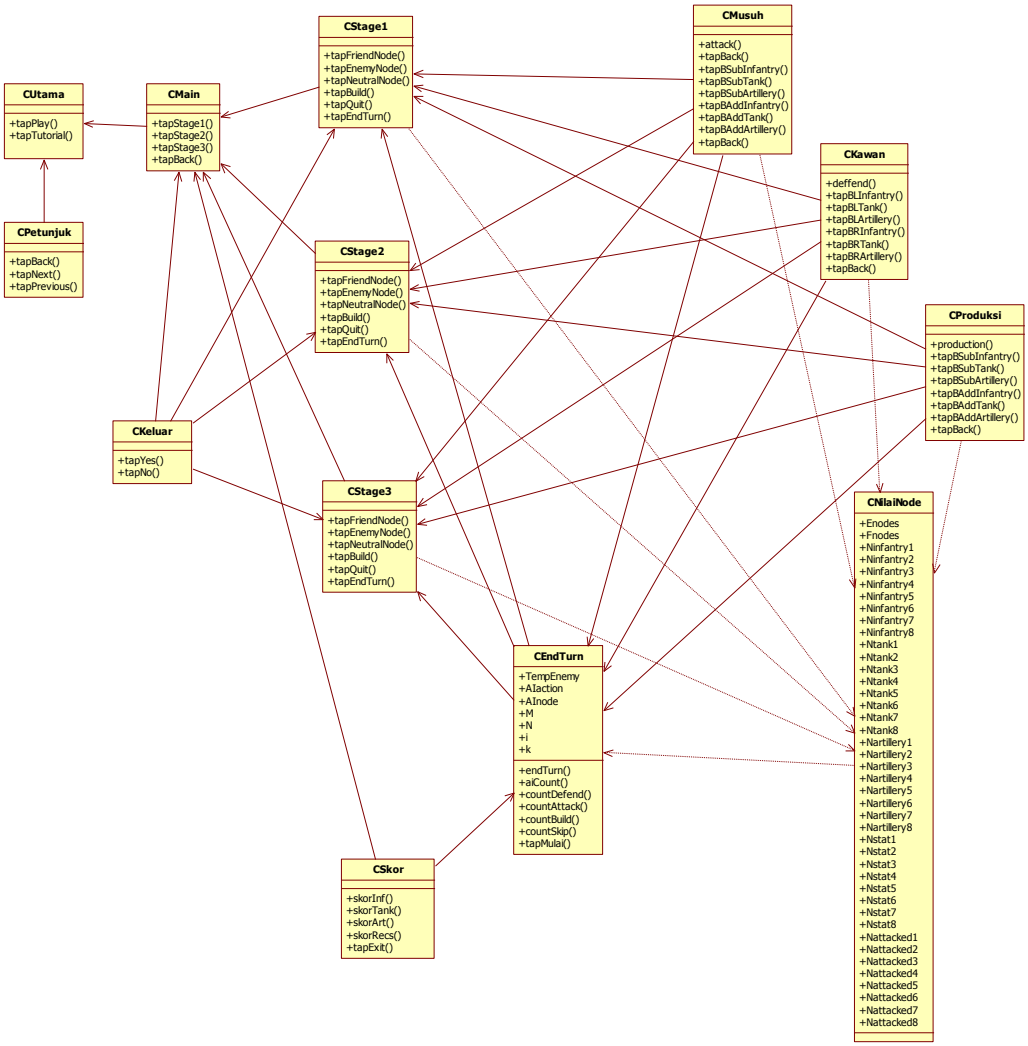
III.1.5.3. Penggambaran Kelas

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-20 Definisi Kelas

Jenis Kelas	Nama Kelas
Boundary	CUtama
	CPetunjuk
	CMain
	CStage1

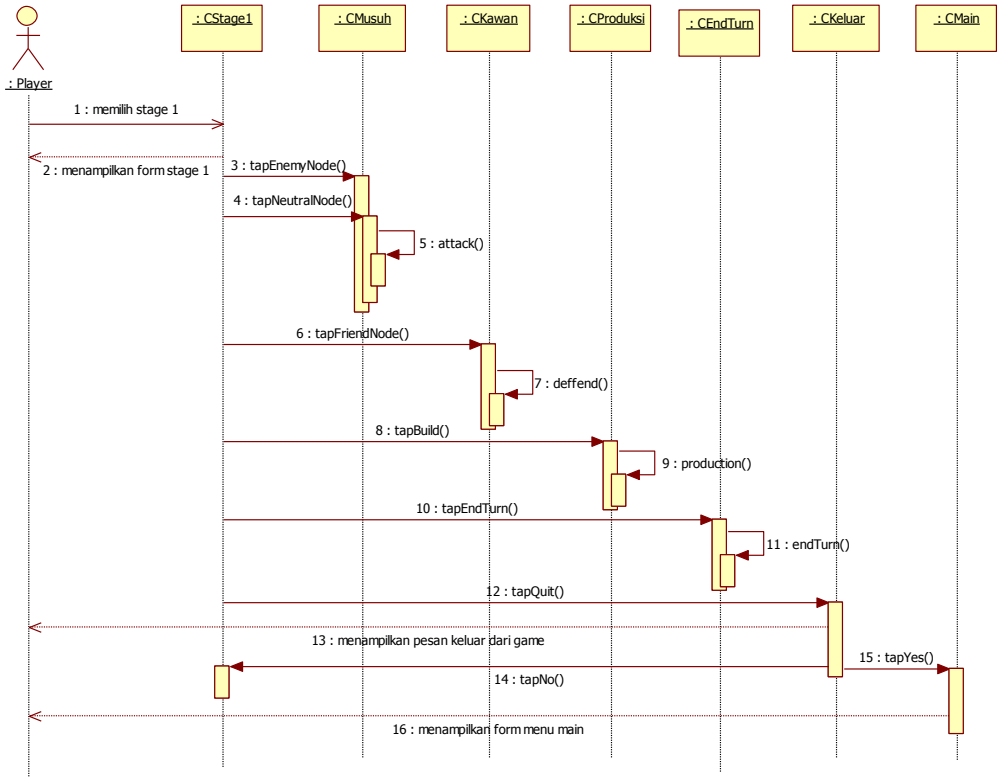
	CStage2
	CStage3
	CMusuh
	CKawan
	CProduksi
	CEndTurn
	CKeluar
	CSkor
Entity	CNilaiNode



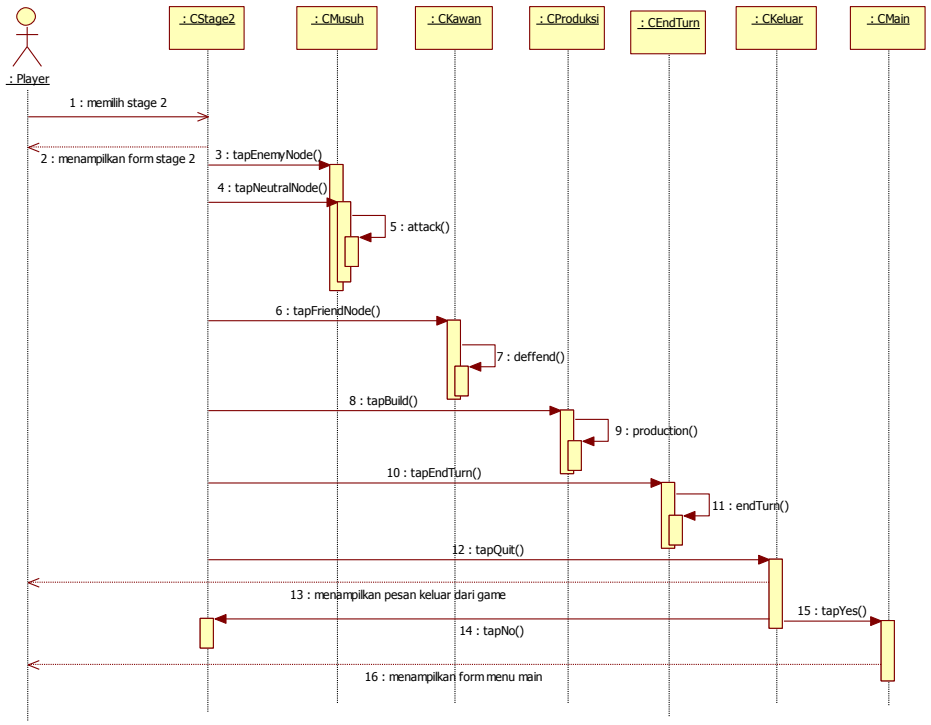
Gambar III-16 Class Diagram

III.1.5.4. Penggambaran Interaksi Antar Objek Pada Game

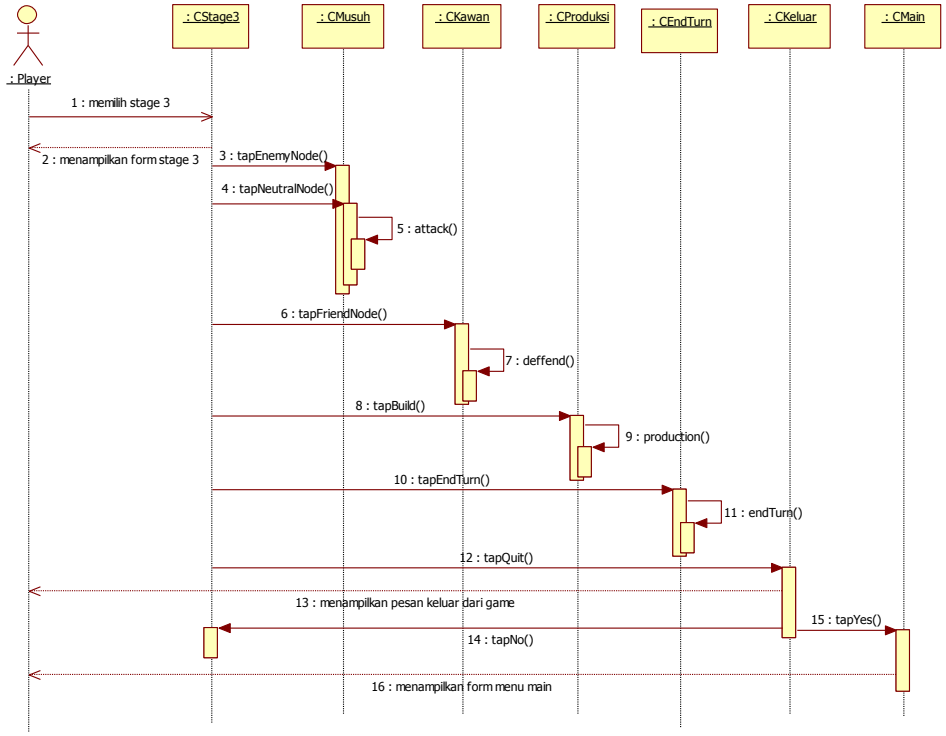
Penggambaran interaksi antar objek menggambarkan interaksi antar objek yang terjadi di dalam dan di luar sistem pada game (termasuk *player*, *display*, dan sebagainya). Penggambaran interaksi antar objek pada game menggunakan *Sequence Diagram*. *Sequence diagram* terdiri atas dimensi *vertical* (waktu) 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. *Sequence diagram* ditunjukan pada Gambar III-17 sampai dengan Gambar III-20.



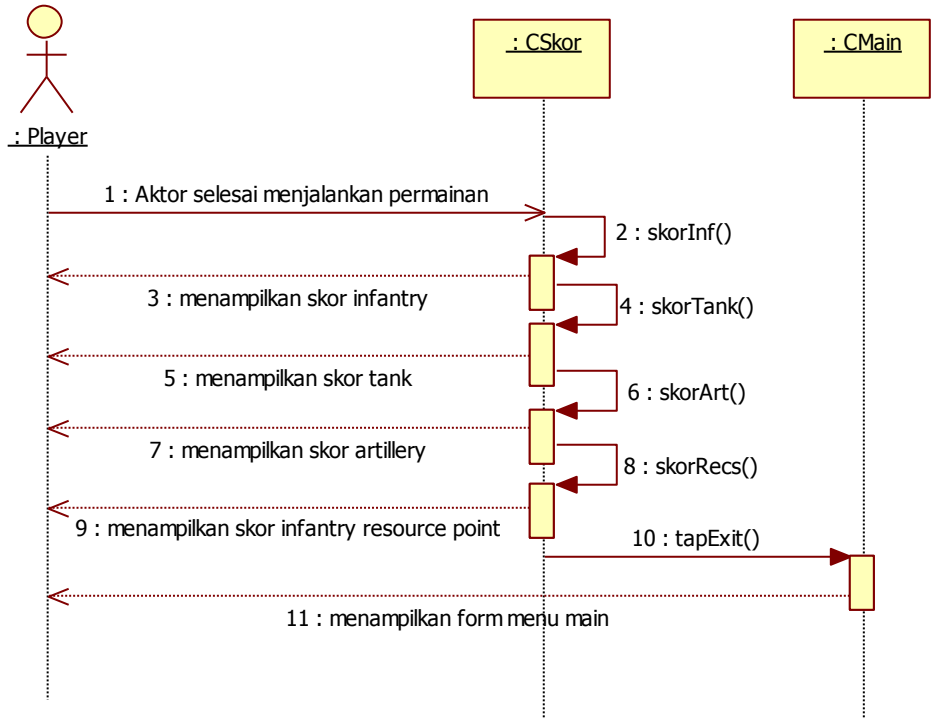
Gambar III-17 *Sequence Diagram* Penyajian *Stage 1*



Gambar III-18 *Sequence Diagram* Penyajian *Stage 2*



Gambar III-19 *Sequence Diagram* Penyajian *Stage 3*



Gambar III-20 *Sequence Diagram* Penyajian Skor

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. Perancangan Komponen Permainan

Perancangan komponen permainan merupakan bagian pendeskripsian setiap komponen yang ada pada game baik karakter, bangunan, dan *story board*. Perancangan komponen permainan terdiri dari perancangan karakter dan perancangan *story board*.

III.2.1.1. Perancangan 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 unit pada setiap fraksi. Perancangan karakter pada game ini dijabarkan pada Tabel III-17**}**.

Tabel III-17 Perancangan Karakter

No.	Nama Karakter	Desain Karakter	Keterangan
1.	GER Infantry		(a) Merupakan kelas unit infantry pada fraksi GER (b) Dikendalikan oleh player (c) Power 10 (d) Harga 30
2.	GER Tank		(a) Merupakan kelas unit tank pada fraksi GER (b) Dikendalikan oleh player (c) Power 20 (d) Harga 60
3.	GER Artillery		(a) Merupakan kelas unit artillery pada fraksi GER (b) Dikendalikan oleh player (c) Power 30 (d) Harga 120
4.	UNION Infantry		(a) Merupakan kelas unit infantry pada fraksi UNION (b) Dikendalikan oleh AI (NPC) (c) Power 10 (d) Harga 30

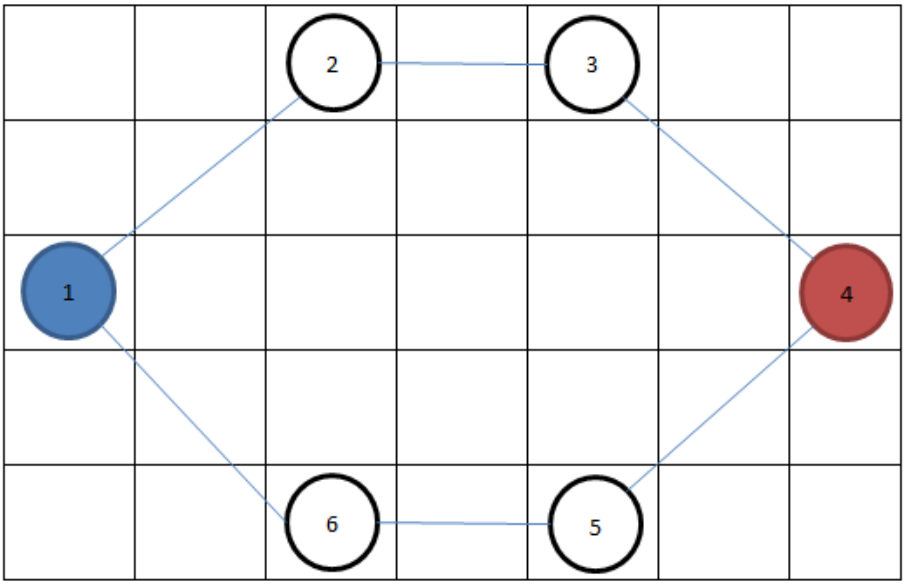
5.	UNION Tank		(a) Merupakan kelas unit tank pada fraksi UNION (b) Dikendalikan oleh AI (NPC) (c) Power 20 (d) Harga 60
6.	UNION Artillery		(a) Merupakan kelas unit artillery pada fraksi UNION (b) Dikendalikan oleh AI (NPC) (c) Power 30 (d) Harga 120

III.2.1.2. Perancangan Story Board

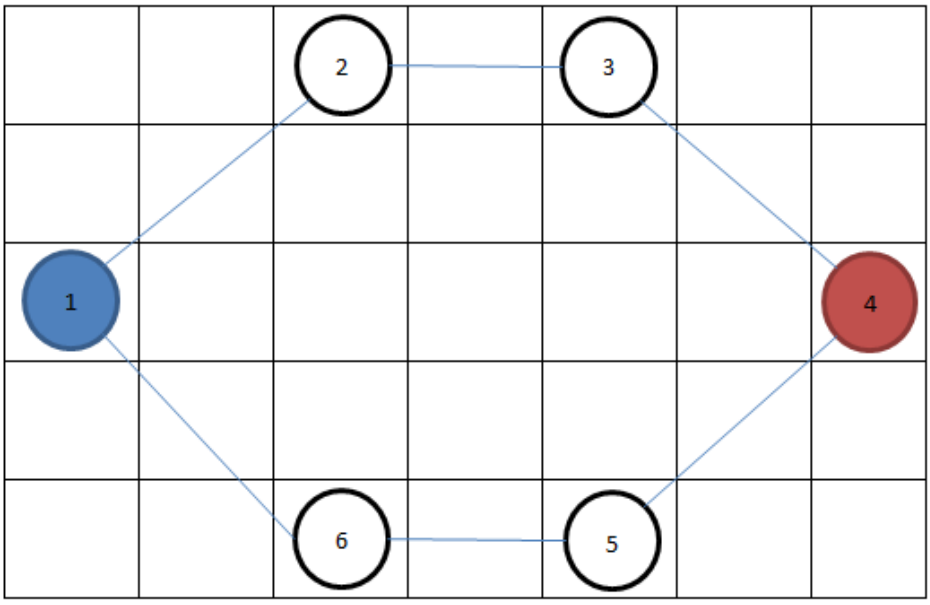
Storyboard merupakan sketsa gambar yang disusun secara berurutan berdasarkan naskah untuk menggambarkan lingkungan pada game. Pada game ini, storyboard merupakan sebuah *map*(peta) yang akan digunakan untuk bertempur dan diterapkan pada sebuah stage. *Map* pada game ini digambarkan pada Gambar III-9, Gambar III-10 dan Gambar III-11 . Pada map tersebut terdapat komponen *terrain* yang dijabarkan pada Terrain Tabel III-18 dan *node* yang dijabarkan pada Tabel III-19.

1. *Map*

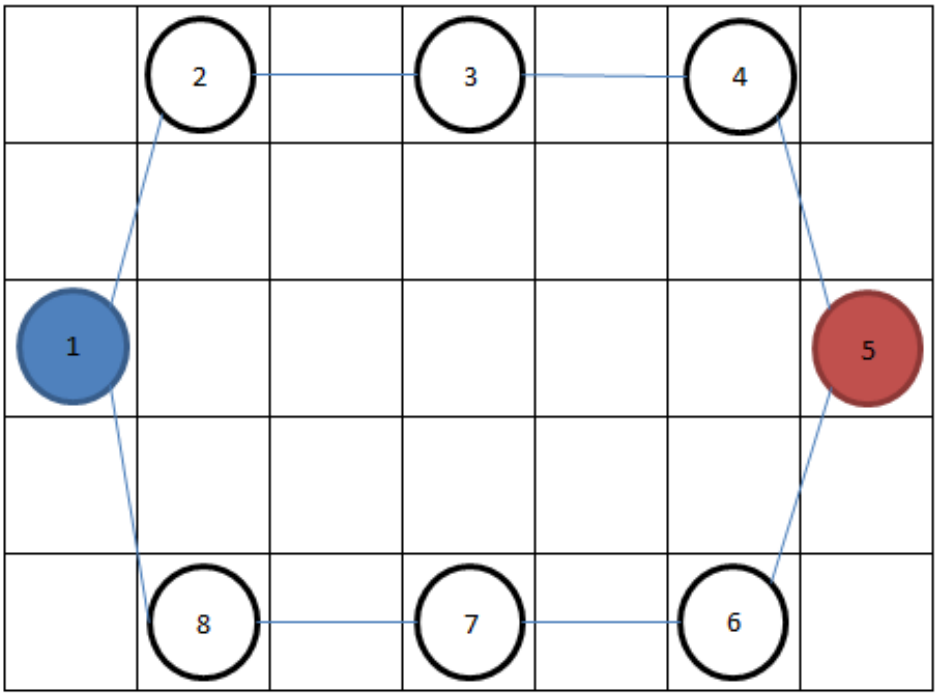
   Map pada *game* ini direpresentasikan menggunakan graph tertutup dan tidak berarah. Didalam *map* terdapat *node*, yaitu simpul-simpul yang harus dikuasai oleh *player*. Pada bagian perancangan ini, *node* digambarkan dengan bulatan berwarna. Angka dalam *node* merepresentasikan urutan *node*. Warna pada *node* merepresentasikan status kepemilikan *node* tersebut. Warna merah untuk *node* milik musuh (AI), warna biru untuk *node* milik *player* dan warna putih untuk *node* netral. Setiap *node* dihubungkan dengan menggunakan garis penghubung yang merepresentasikan hubungan tetangga dari sebuah *node*, contohnya adalah *node* 1 memiliki dua tetangga yaitu *node* 2 dan *node 6*.



Gambar III-9 *Map Stage 1*



Gambar III-10 *Map Stage 2*



Gambar III-11 *Map Stage 3*

2. *Terrain*

Tabel III-18 Perancangan *Terrain*

No.	Nama *Terrain*	Desain *Terrain*	Keterangan
1.	*Mountain*		Terrain yang merupakan gunung
2.	*Open Plain*		Terrain yang merupakan lapangan terbuka
3.	*Forest*		Terrain yang merupakan hutan

4.	*Water*		Terrain yang merupakan perairan
5.	*Link*		Merupakan garis penghubung antar *node*.

3. *Node*

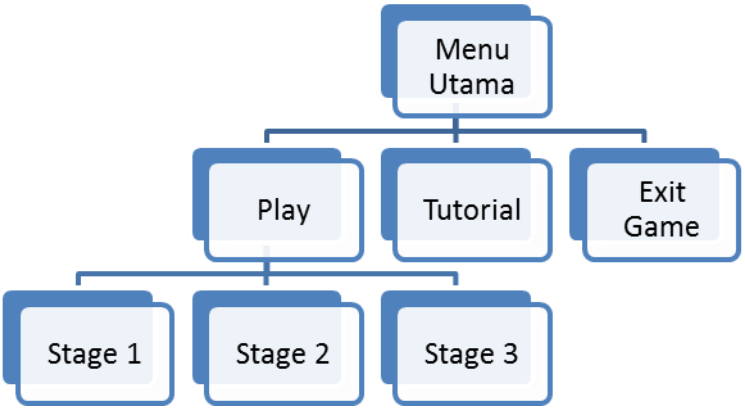
Tabel III-19 Perancangan *Node*

No.	Nama *Node*	Desain *Node*	Keterangan
1.	*Neutral Node*		(a) Node yang tidak dimiliki oleh pihak manapun. (b) Tidak memiliki pertahanan.
2.	*GER Node*		(a) Node yang tidak dimiliki oleh GER. (b) Dapat memiliki pertahanan. (c) Dikendalikan oleh Player. (d) Menghasilkan 10 point uang per *turn* kepada player.
3.	*UNION Node*		(a) Node yang tidak dimiliki oleh UNION. (b) Dapat memiliki pertahanan. (c) Dikendalikan oleh AI. (d) Menghasilkan 10 point uang per *turn* kepada AI.

III.2.2. Perancangan Struktur Menu

Dalam pembangunan aplikasi diperlukan sebuah struktur menu untuk menunjukan keterkaitan dan tingkatan antara menu utama dan sub menu, sehingga

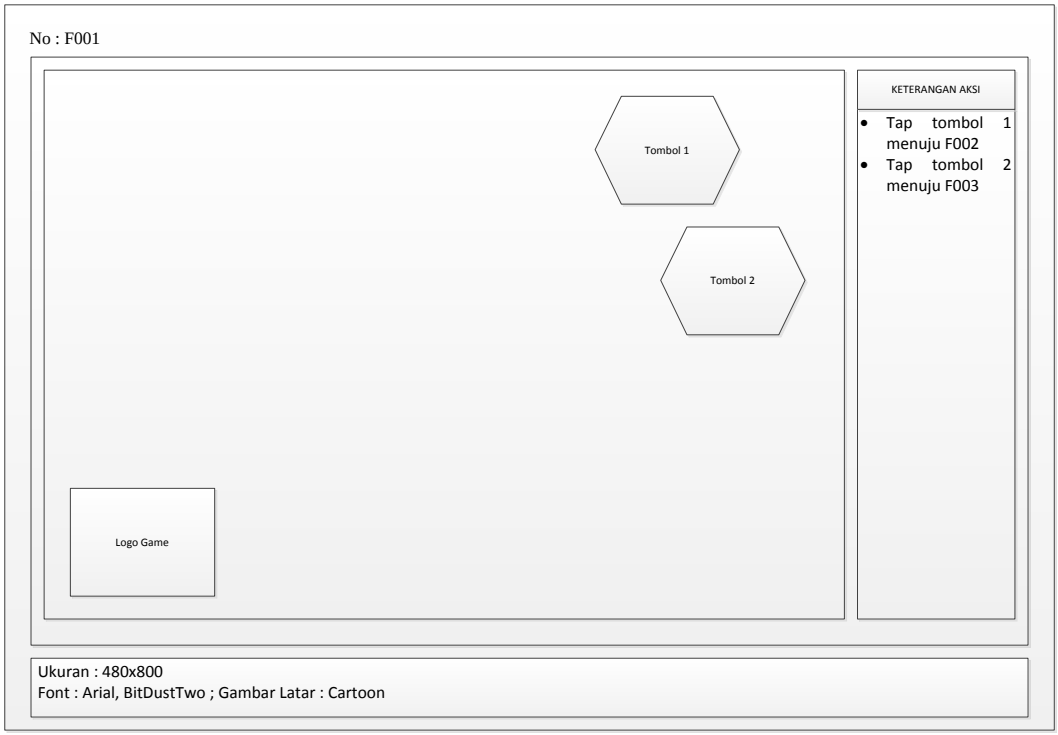
dapat lebih mudah di mengerti. Perancangan trusktur menu digambarkan pada Gambar III-12.



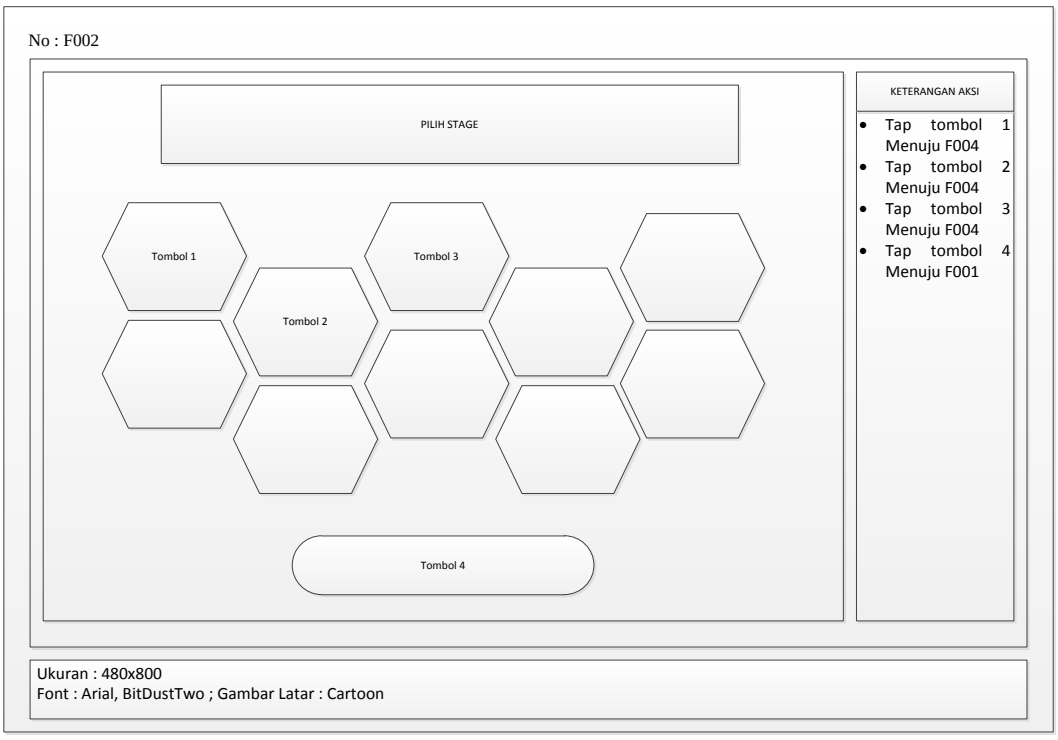
Gambar III-12 Struktur Menu

III.2.3. 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 pada game ini digambarkan pada Gambar III-35 sampai dengan Gambar III-41.



Gambar III-13 Perancangan Antarmuka Menu Utama



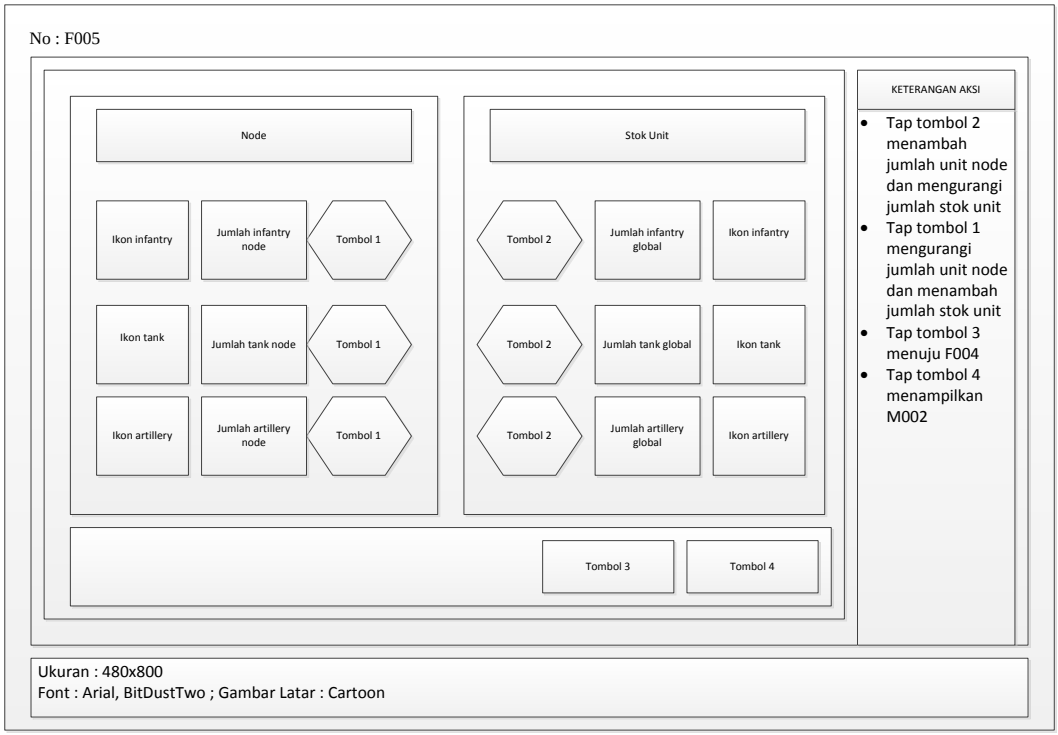
Gambar III-14 Perancangan Antarmuka Menu Main



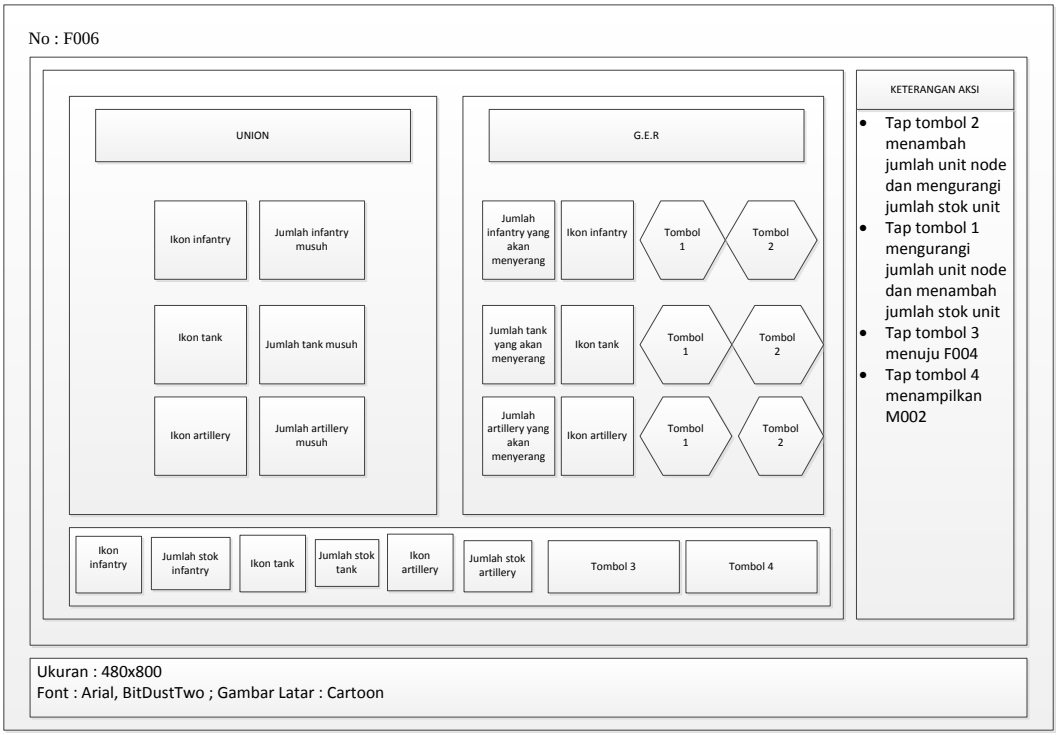
Gambar III-15 Perancangan Antarmuka Menu Petunjuk Main



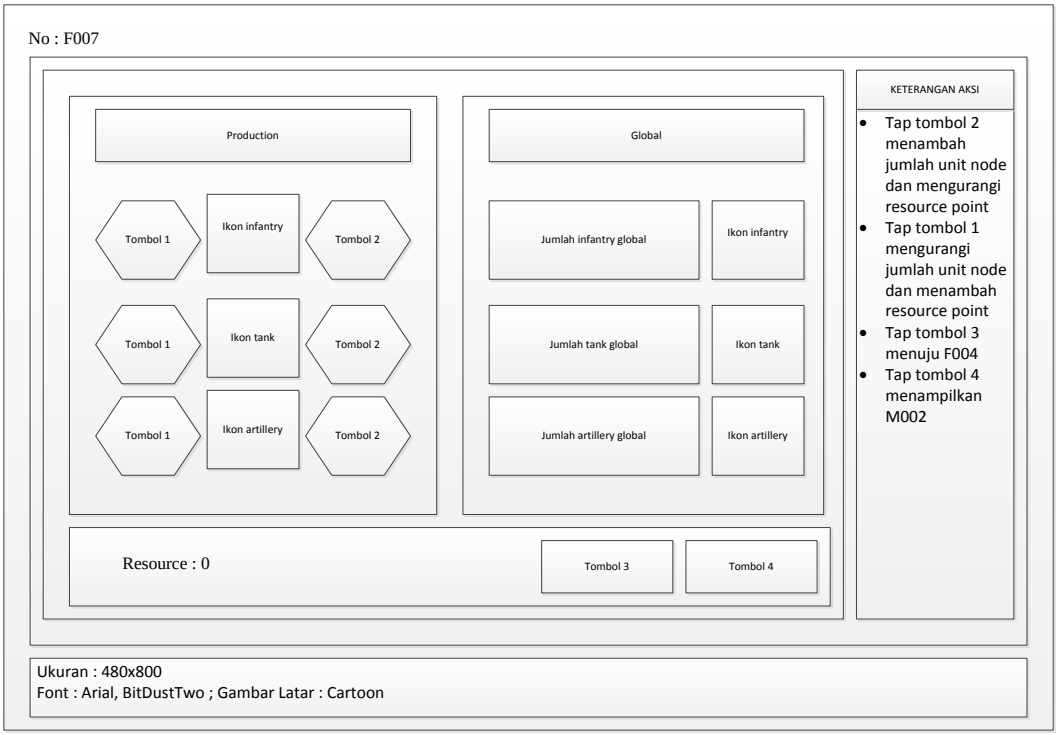
Gambar III-16 Perancangan Antarmuka Setiap Stage



Gambar III-17 Perancangan Antarmuka Mempertahankan *Node*



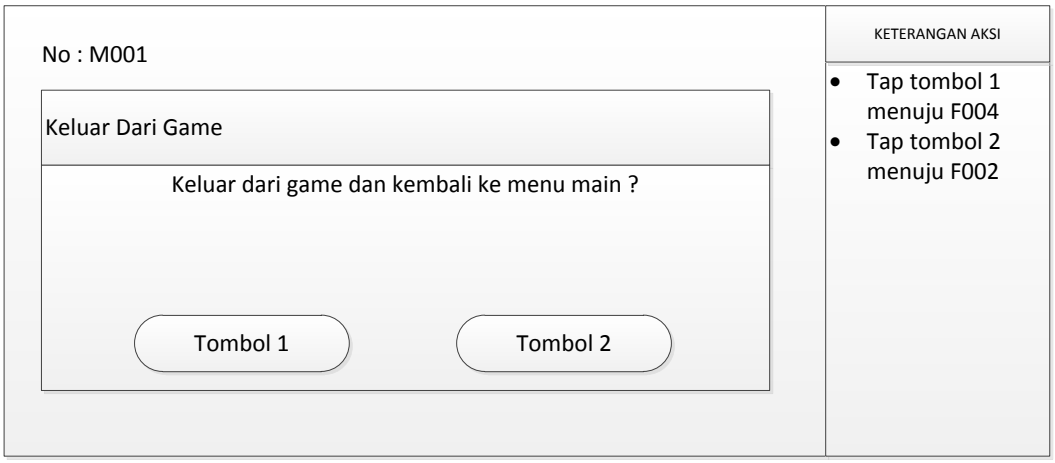
Gambar III. 18 Perancangan Antarmuka Menyerang *Node*



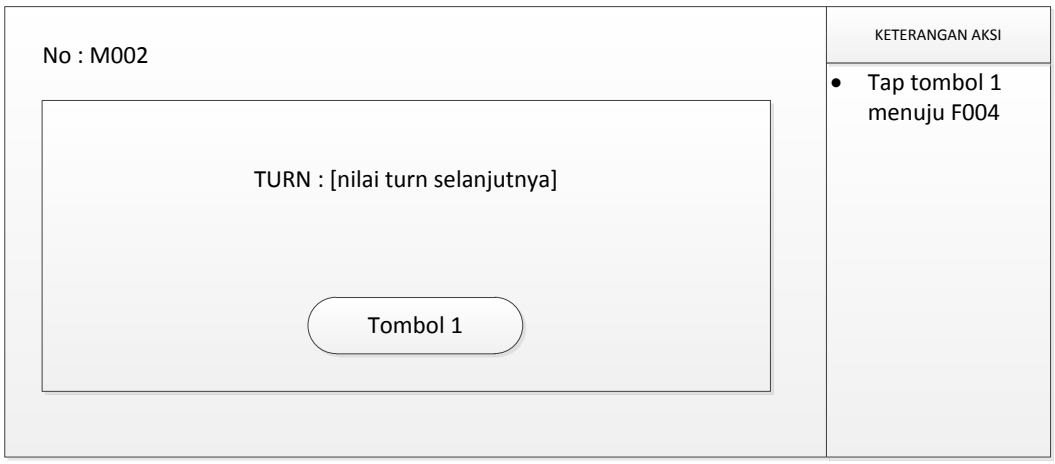
Gambar III. 19 Perancangan Antarmuka Memproduksi Unit

III.2.4. Perancangan Pesan

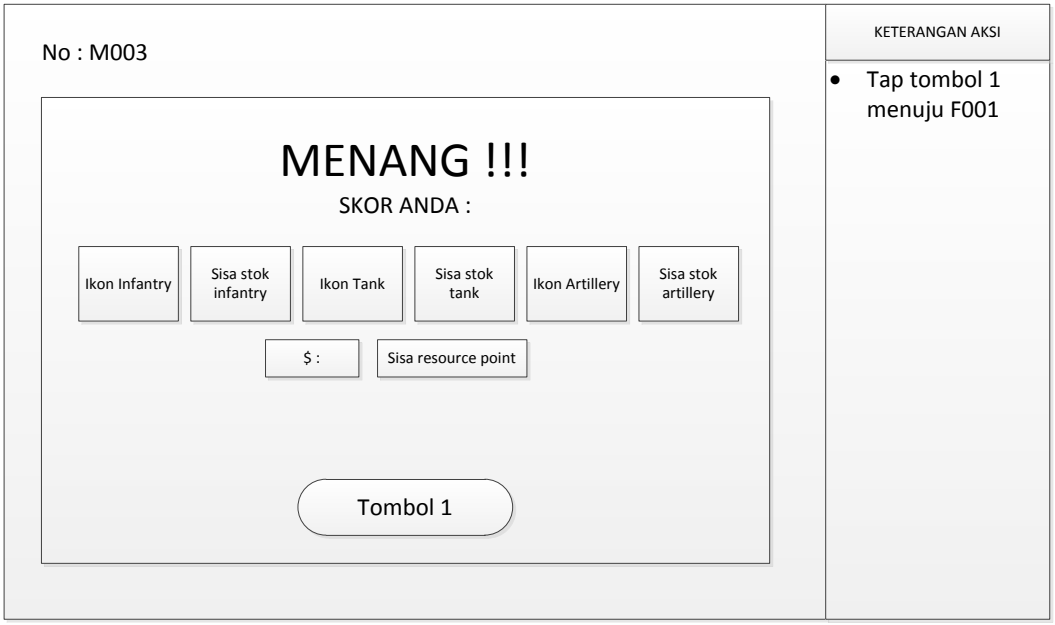
Perancangan pesan adalah rancangan antar muka dari pesan yang akan ditampilkan apabila terjadi kesalahan dan pemberitahuan dalam proses yang dilakukan. Rancangan pesan dari gameini digambarkan pada Gambar III-20 sampai dengan Gambar III. 23.



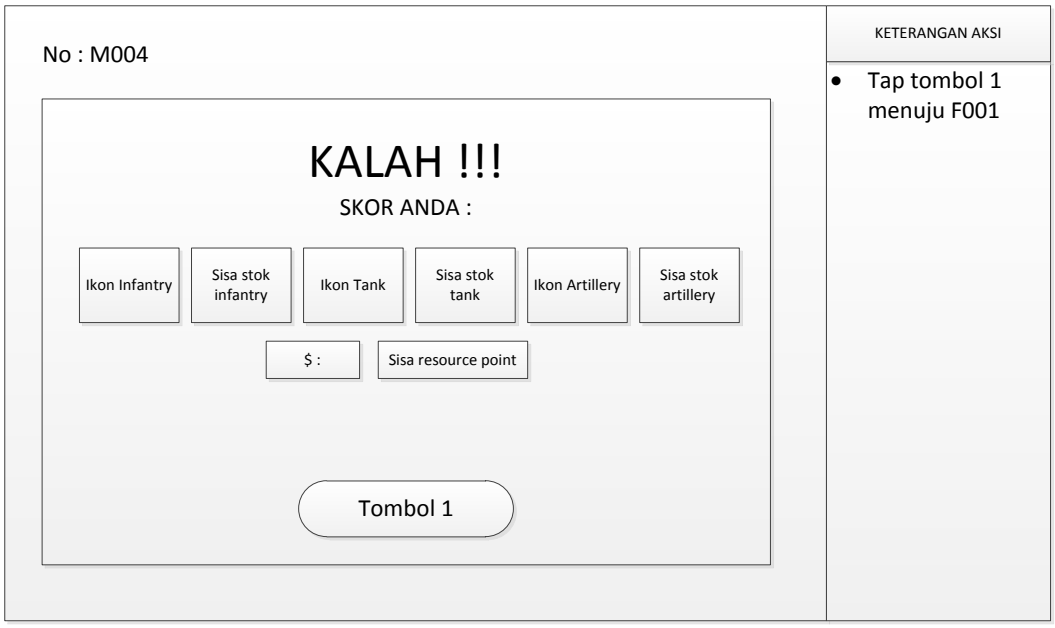
Gambar III-20 Perancangan Pesan Keluar Dari Permainan



Gambar III-21 Perancangan Pesan *End Turn*



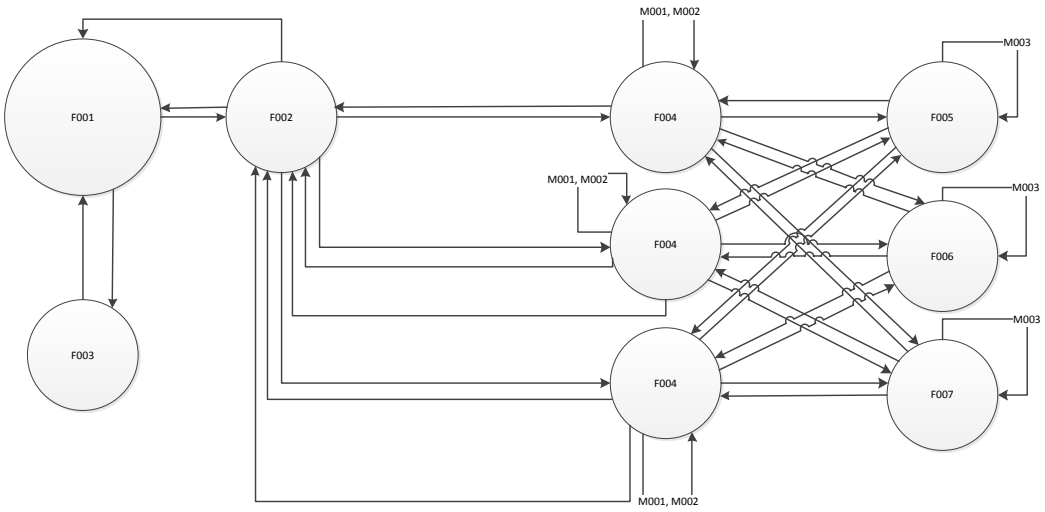
Gambar III-22 Perancangan Pesan Menang



Gambar III. 23 Perancangan Pesan Menang

III.2.5. Jaringan Semantik

Jaringan semantik merupakan alat yang mangkus untuk merepresentasikan pemetaan data, yang bertujuan mencegah terjadinya duplikasi data.Jaringan semantik pada game ini digambarkan pada Gambar III-24.



Gambar III-24 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 CEndTurn.aiCount()

   Method CEndTurn.aiCount(i,M) merupakan method pengecekan setiap *node* yang ada. Dari pengecekan setiap node, akan menghasilkan satu nilai keputusan pada setiap *node*, yang nantinya akan dipilih satu nilai keputusan berdasarkan nilai *modifier* yang terbesar dari seluruh *node*. Method ini digambarkan pada Gambar III-25.

b. Method CEndTurn.countAttack(i,M)

   Method CEndTurn.countAttack(i,M) adalah method perhitungan untuk keputusan menyerang pada AI. Pada method ini, beberapa nilai diperhatikan seperti nilai nilai status kepemilikan apakah 2, kekuatan *node* dan jumlah resource yang tersedia dibandingkan dengan resource yang ada pada *node* yang diperiksa. Setelah itu, bagi nilai jumlah keadaan yang cocok dengan jumlah kondisi yang ada, dimana pada kondisi ini jumlah kondisi yang ada adalah enam. Method ini digambarkan pada Gambar III. 26.

c. Method CEndTurn.countDefense(i,M)

   Method countDefense(i,M) adalah method perhitungan untuk keputusan bertahan pada AI. Pada method ini, beberapa nilai diperhatikan seperti nilai nilai status kepemilikan apakah 2, kekuatan *node* dan jumlah resource yang tersedia dibandingkan dengan resource yang ada pada *node* yang diperiksa, bandingkan nilai dana apakah lebih besar dari 30 atau tidak karena 30 merupakan nilai minimum untuk membuat pasukan. Setelah itu, bagi nilai jumlah keadaan yang cocok dengan jumlah kondisi yang ada, dimana pada kondisi ini jumlah kondisi yang ada adalah enam. Method ini digambarkan pada Gambar III-27.

d. Method CEndTurn.countBuild(i,M)

   Method CEndTurn.countBuild(i,M) adalah method perhitungan untuk keputusan memproduksi pasukan pada AI. Pada method ini, beberapa nilai

diperhatikan seperti nilai nilai status kepemilikan apakah 2, kekuatan *node* dan jumlah resource yang tersedia dibandingkan dengan resource yang ada pada *node* yang diperiksa, bandingkan nilai dana pada resource yang tesedia, apakah lebih besar dari 30 atau tidak karena 30 merupakan nilai minimum untuk membuat pasukan. Setelah itu, bagi nilai jumlah keadaan yang cocok dengan jumlah kondisi yang ada, dimana pada kondisi ini jumlah kondisi yang ada adalah enam. Method ini digambarkan pada Gambar III-28.

e. Method CEndTurn.countSkip(i,M)

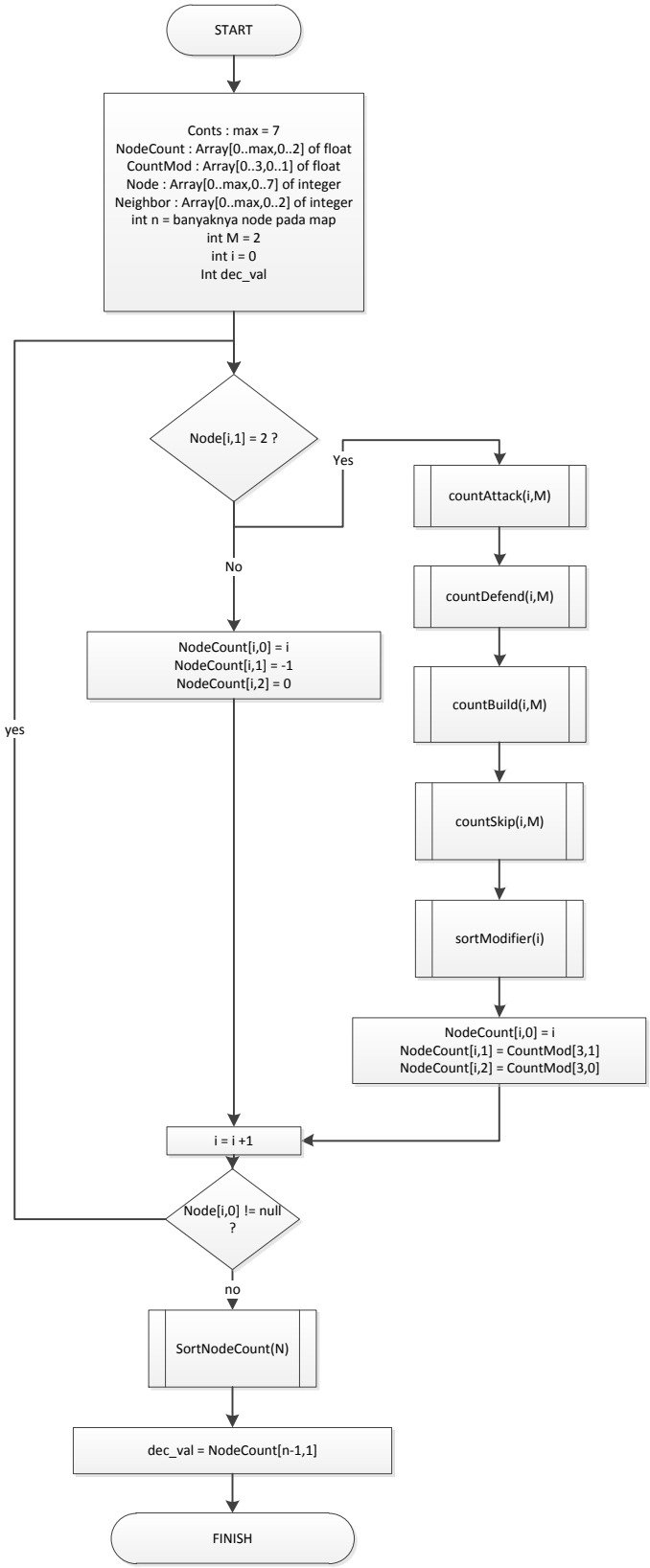
   Method CEndTurn.countSkip(i,M) adalah method perhitungan untuk keputusan memproduksi pasukan pada AI. Pada method ini, beberapa nilai diperhatikan seperti nilai nilai status kepemilikan apakah 2, kekuatan *node* dan jumlah resource yang tersedia dibandingkan dengan resource yang ada pada *node* yang diperiksa, bandingkan nilai dana pada resource yang tesedia, apakah lebih kecil dari 30 atau tidak karena 30 merupakan nilai minimum untuk membuat pasukan. Setelah itu, bagi nilai jumlah keadaan yang cocok dengan jumlah kondisi yang ada, dimana pada kondisi ini jumlah kondisi yang ada adalah enam. Method ini digambarkan pada Gambar III-29.

f. Method CEndTurn.sortModifier(i)

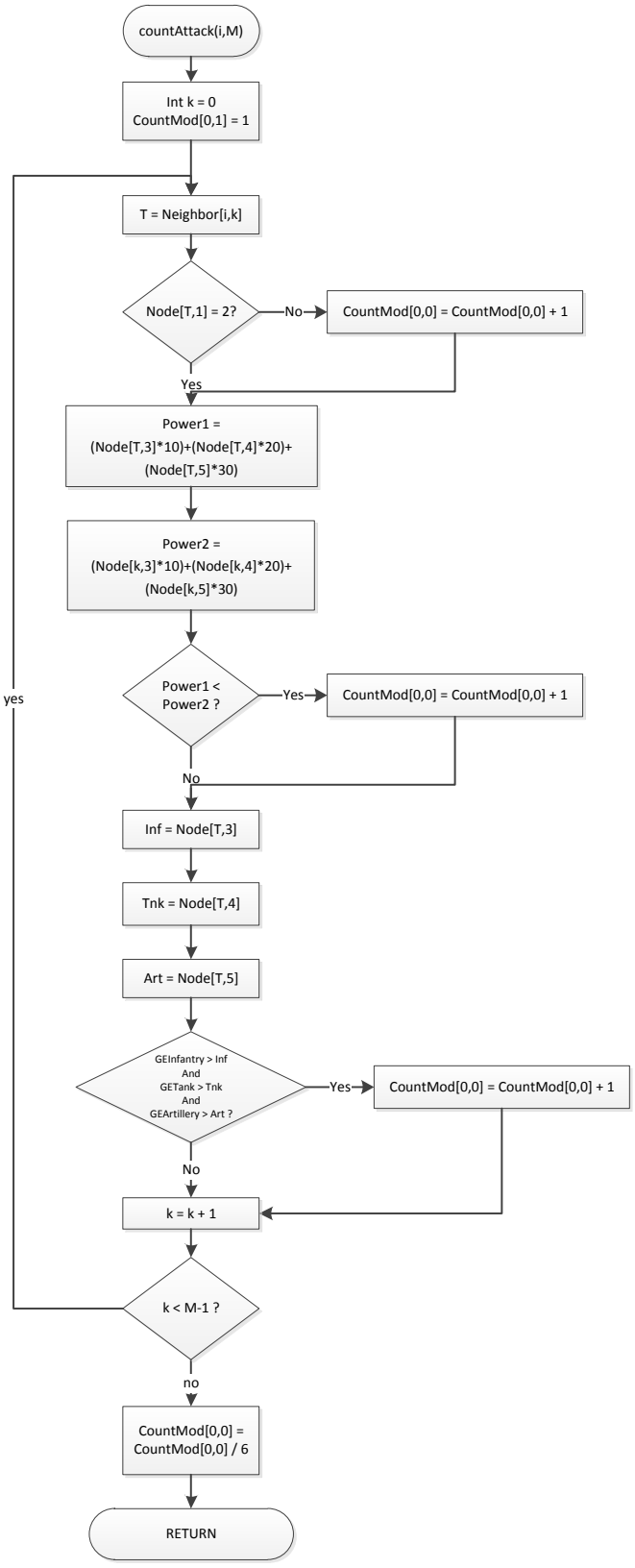
   Method CEndTurn.sortModifier(i) adalah method yang merupakan proses mengurutkan nilai modifier yang didaptkan dari method CEndTurn.countAttack, method CEndTurn.countDefense(i,M), method CEndTurn.countBuild(i,M) dan method CEndTurn.countSkip(i,M). Pada method ini, nilai terbesar akan diambil dijadikan keputusan pada sebuah node. Method ini digambarkan pada Gambar III-30.

g. Method CEndTurn.sortScore(n)

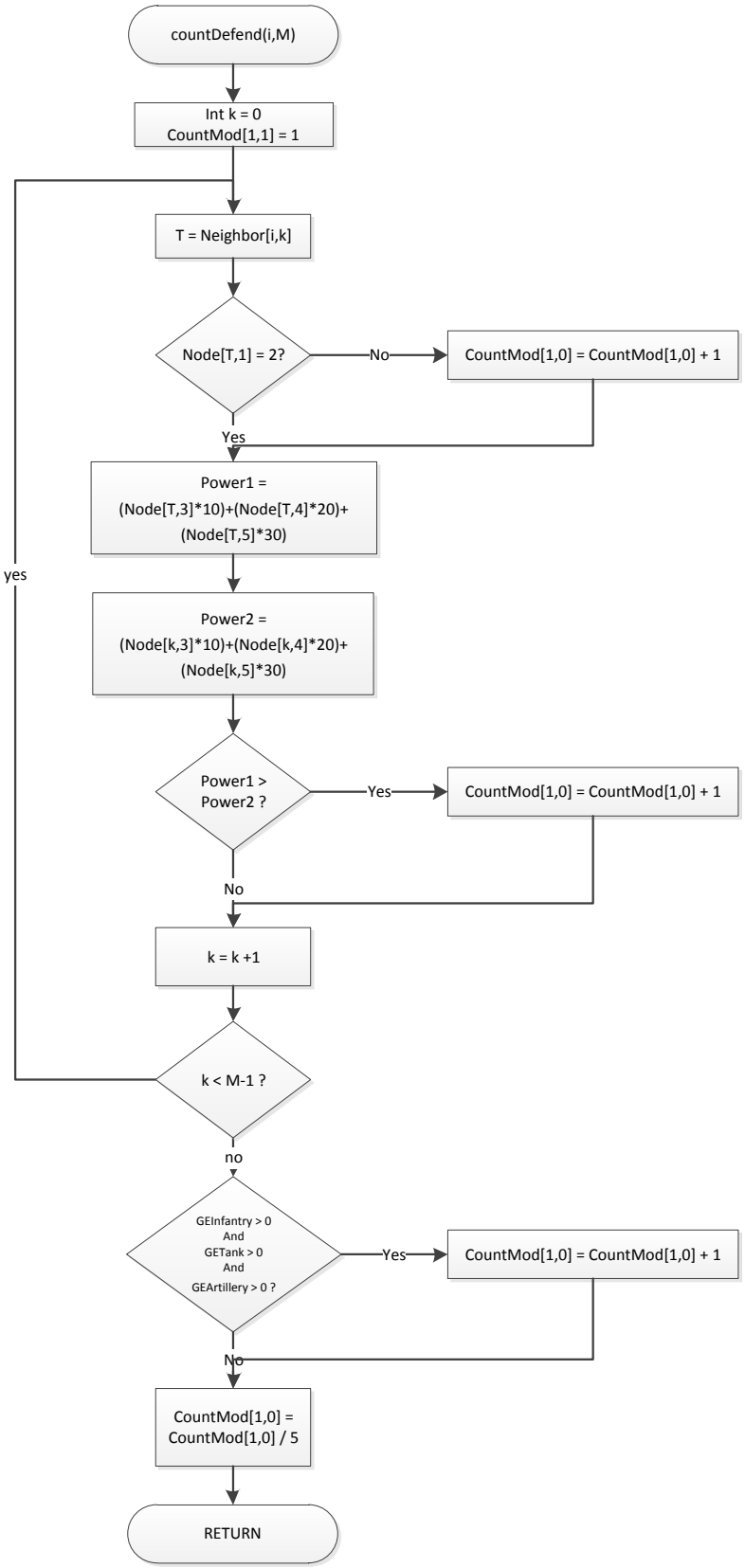
   Method CEndTurn.sortScore(n) adalah method yang merupakan proses pengurutan nilai *modifier* yang telah didapatkan dari perhitungan nilai modifier setiap *node* untuk mendapatkan keputusan. Method ini digambarkan pada Gambar III. 31.



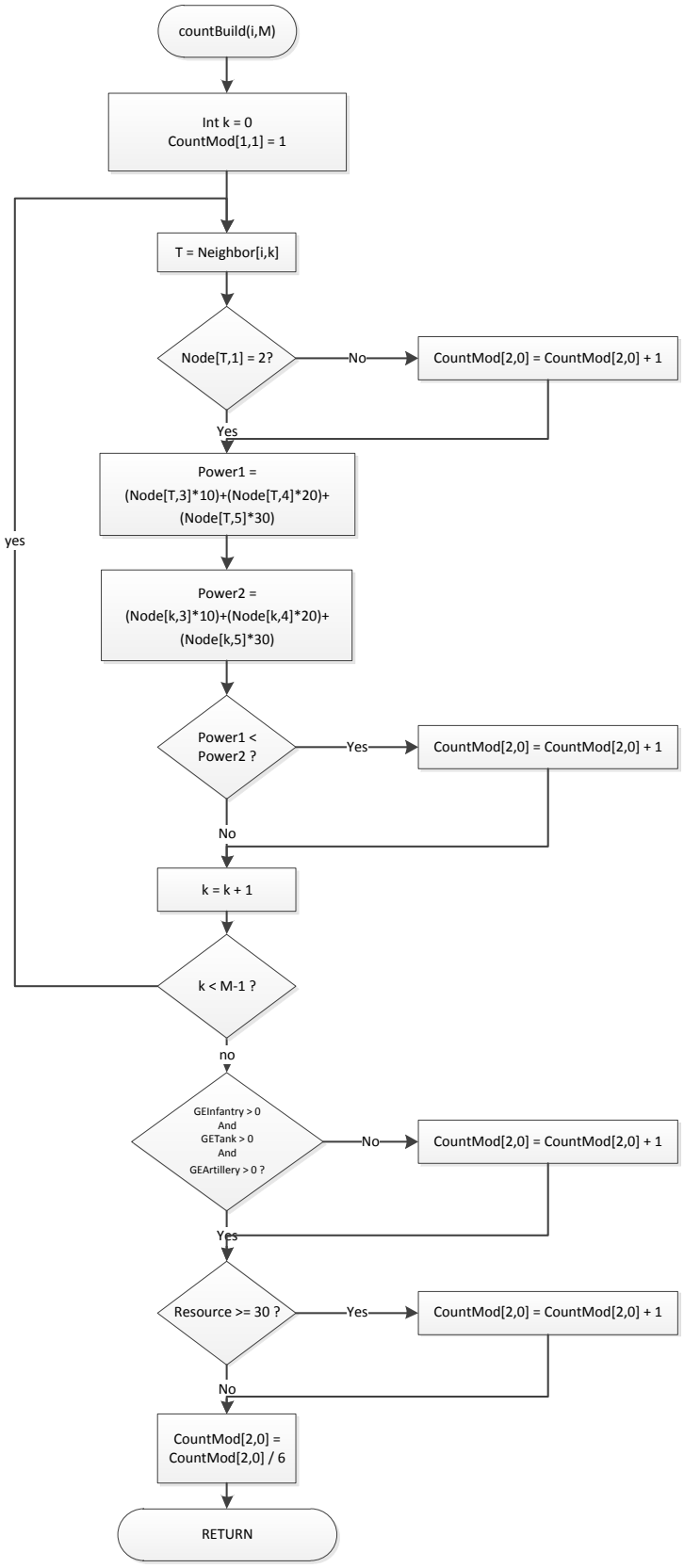
Gambar III-25 Method CEndTurn.aiCount()



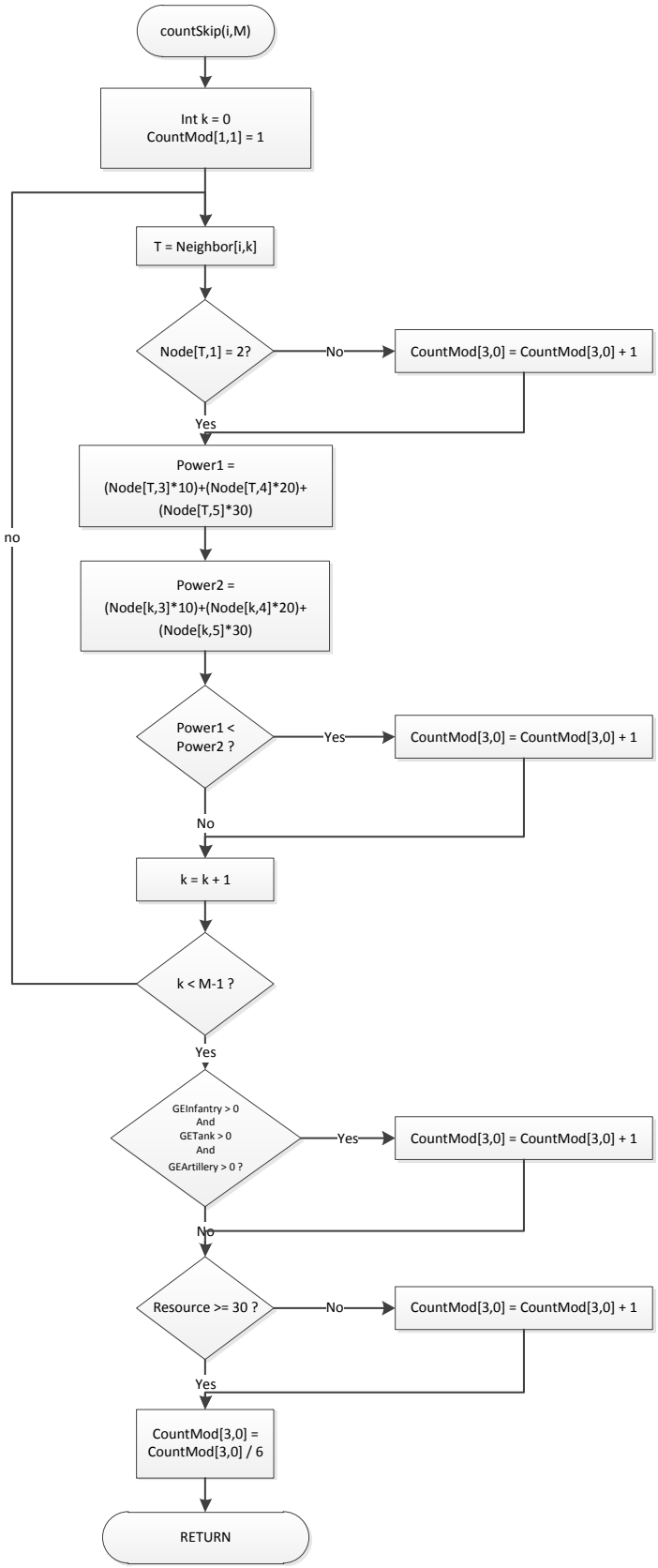
Gambar III. 26 method CEndTurn.countAttack(i,M)



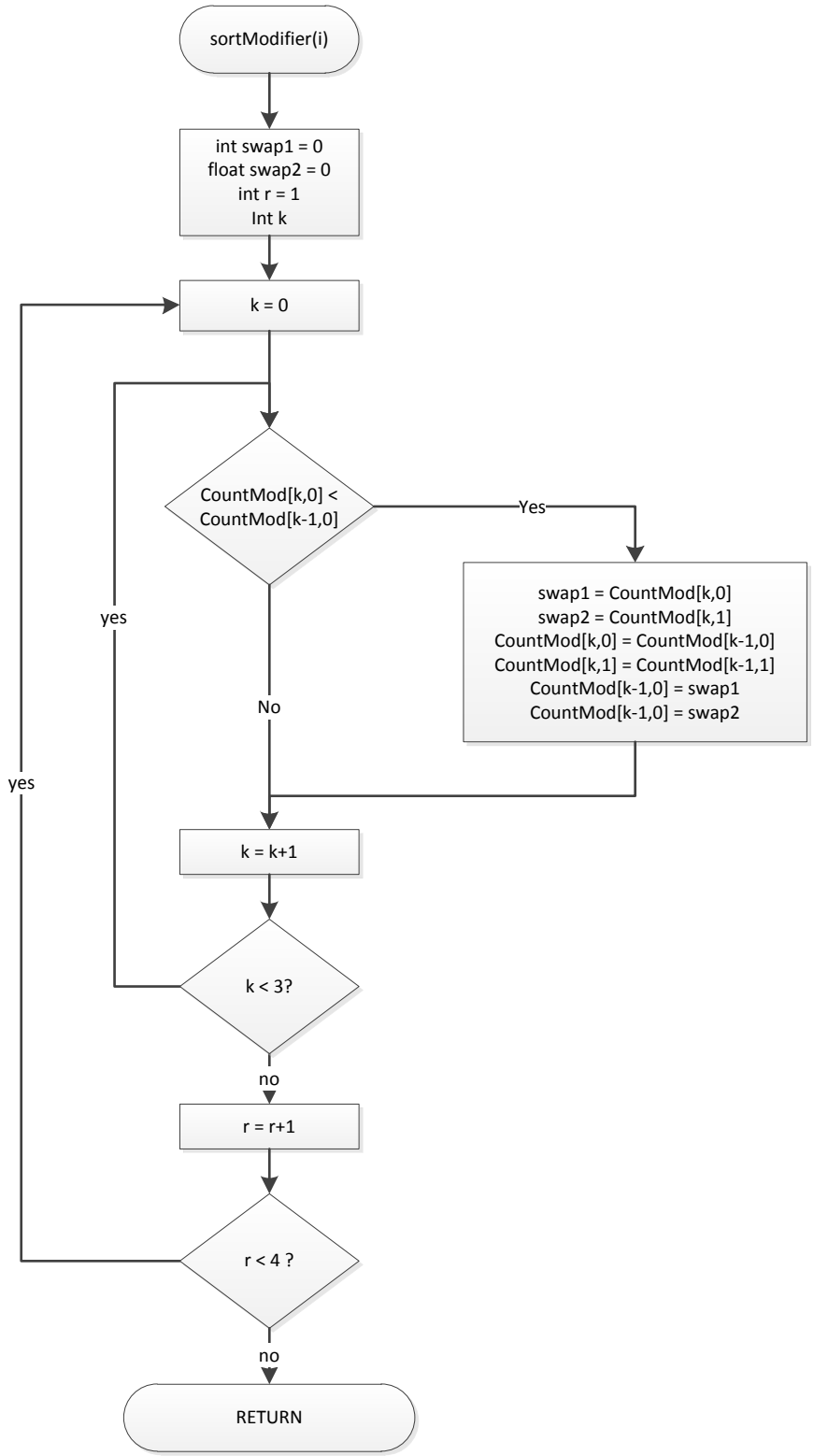
Gambar III-27 method CEndTurn.countDefend(i,M)



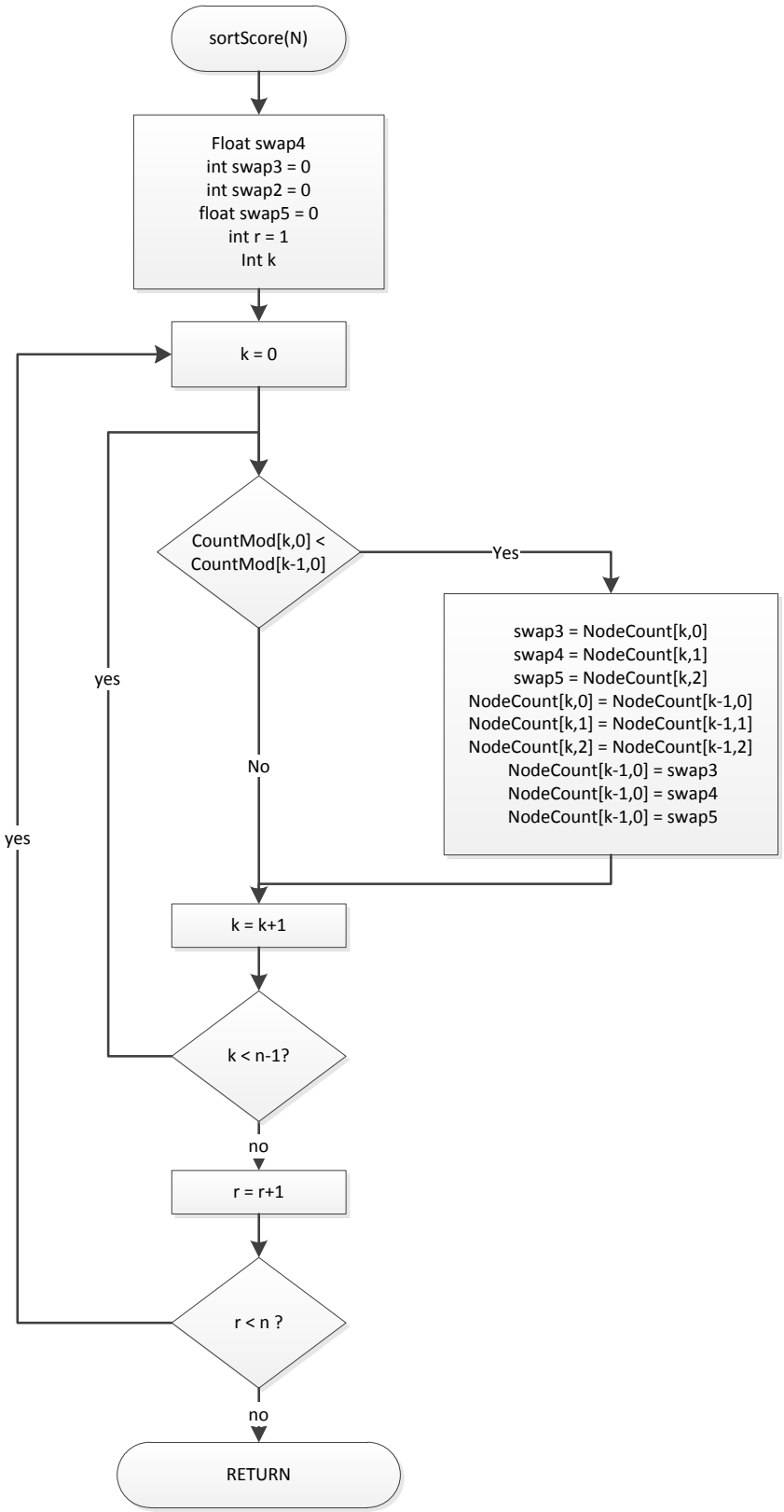
Gambar III-28 method CEndTurn.countBuild(i,M)



Gambar III-29 method CEndTurn.countSkip(i,M)



Gambar III-30 Method CEndTurn.sortModifier(i)



Gambar III. 31 Method CEndTurn.sortScore(N)