Vol. 2, No. 2, Oktober 2013, ISSN : 2089-9033

TURN BASED STRATEGY GAME MENGGUNAKAN ALGORITMA RESOURCE ASSIGNMENT PADA PERANGKAT MOBILE BERBASIS ANDROID

Riandanu Madi Utomo, Nelly Indriani Widiastuti

Universitas Komputer Indonesia
Jl. Dipati Ukur 112 - 116 Telp. (022)2504119 Bandung 40132
Email : riandanumu@gmail.com, alifahth@yahoo.com

ABSTRAK

Turn-based strategy (TBS) game adalah turunan dari game dengan genre strategi dimana pemainnya saling bergiliran pada pengambilan keputusannya dalam bermain. Pada game jenis ini, player dapat memainkan oleh satu pemain saja (*single player*) sedangkan lawannya adalah komputer atau istilah lainnya disebut *Non Playable Character* (NPC). Berdasarkan kebutuhan tersebut, maka perlu diterapkan NPC atau program yang memiliki *Artificial Intelegence* (AI). Peran NPC tersebut berfungsi untuk menggantikan peran manusia dalam menjadi teman atau musuh dari pemain pada sebuah game. NPC tersebut pada game TBS harus memiliki kemampuan berfikir dan berstrategi secara logis. Untuk memunculkan kemampuan tersebut, NPC harus dapat memperhitungkan kondisi yang dialaminya sebagai dasar keputusan yang akan diambil. *Resource Assigment Algorithm* (RAA) dipilih untuk menyelesaikan masalah tersebut, karena algoritma RAA mempunyai kemampuan untuk menghasilkan sebuah nilai berdasarkan beberapa nilai yang ada dengan diurutkan menggunakan skala prioritas [**2**]. Game TBS pada perangkat mobile masih jarang, sehingga penempatan game ini pada perangkat mobile dipilih agar game ini dapat dimainkan dimana saja dan kapan saja.

Kata Kunci: *Turn-Based Strategy*, *Game*, Algoritma *Resource Assignment*, Perangkat *Mobile*

1. PENDAHULUAN

Turn-based strategy (TBS) game adalah turunan dari game dengan genre strategi dimana pemainnya saling bergiliran pada pengambilan keputusannya dalam bermain. Contohnya adalah permainan catur kuno yang merupakan cikal bakal game TBS moderen dimana bidak putih dapat dijalankan oleh pemainnya ketika pemain yang mengendalikan bidak hitam telah menjalankan bidaknya. Untuk menguasai game TBS, player memerlukan kemampuan berfikir dan berstrategi secara logis. Game ini tidak memerlukan kecepatan dan keakuratan dalam bermain tetapi memerlukan pengambilan keputusan yang tepat yang dapat membawa pemain pada kemenangan [**1**].

Varian game TBS semakin berkembang, tidak hanya dalam bentuk catur namun ada pula yang berbentuk simulasi peperangan dengan aturan yang lebih kompleks, pemainnya masih bergiliran dalam mengambil keputusan. Game TBS dapat dimainkan oleh hanya satu pemain saja (single player), sebagai lawan diperankan oleh Non Playable Character (NPC). Peran NPC adalah sebagai teman atau musuh dari pemain pada sebuah game. NPC pada game TBS memerlukan kemampuan berfikir dan berstrategi secara logis yaitu dengan memasukan kondisi yang sedang dialaminya pada penentuan keputusan. Berdasarkan masalah yang telah dijelaskan, maka NPC pada game TBS pun harus memiliki kemampuan yang sama agar pemain merasa lebih tertantang dan merasa seakan-akan bermain dengan pemain manusia.

Solusi untuk memunculkan kemampuan berfikir dan berstrategi secara logis pada AI adalah menggunakan *Resource Assignment Algorithm* (RAA). RAA mempunyai kemampuan untuk menghasilkan sebuah nilai berdasarkan beberapa nilai yang ada dengan diurutkan menggunakan skala prioritas [**2**]. Dalam penerapannya pada game TBS maka nilai yang akan dihasilkan adalah nilai untuk mengeksekusi sebuah keputusan, dan beberapa nilai yang diurutkan dengan menggunakan skala prioritas adalah berbagai kemungkinan kondisi yang dihadapi sehingga AI akan terlihat memiliki kemampuan berfikir dan berstrategi secara logis karena faktor-faktor kondisi yang sedang dihadapi oleh AI ikut dimasukan dalam perhitungan.

Game TBS dapat ditempatkan pada berbagai platform, baik desktop, mobile, ataupun konsol. Game TBS tidak memerlukan banyak tombol kontrol pada perangkat yang menjalankannya, bahkan dapat dilakukan hanya dengan menggunakan

Vol. 2, No. 2, Oktober 2013, ISSN : 2089-9033

touch screen. Game TBS pada mobile platform dipilih agar game ini dapat dimainkan dimana saja dan kapan saja. Perkembangan game TBS pada platform mobile juga masih dinilai lambat. Menurut data dari www.metacritic.com, hanya terdapat 28 game TBS pada platform mobile untuk Android. Berdasarkan hal tersebut, maka peluang untuk mengembangkan game TBS pada platform mobile terutama Android masih sangat besar.

RUMUSAN MASALAH

Berdasarkan pendahuluan yang telah dijelaskan, maka rumusan masalah adalah bagaimana membuat NPC pada game TBS mobile yang memiliki kemampuan yang dapat memperhitungkan kondisi yang dialaminya sebagai dasar keputusan yang akan diambil menggunakan *Resource Assigment Algorithm.*

2. LANDASAN TEORI

2.1 *Turn-based Strategy Game*

Turn-based strategy (TBS) game adalah turunan dari game dengan genre strategi dimana pemainnya bergantian dalam bermain. Hal ini sangat bertolak belakang dengan game-game lain dimana para pemainnya bermain bersama secara simultan. Game TBS berawal dari permainan catur kuno dimana player saling bergiliran menjalankan bidak caturnya.Game TBS mempunyai *gameplay* yang menawarkan pemainnya memilih satu dari berbagai macam keputusan yang disediakan pada game.Saat ini game TBS banyak dijumpai dalam video game.Berbagai jenis gameplay turunan game TBS ini pun bermunculan seperti menghabisi musuh (*annihilate*), menguasai wilayah (*dominate*), bahkan dengan misi-misi lebih spesifik lagi [3].

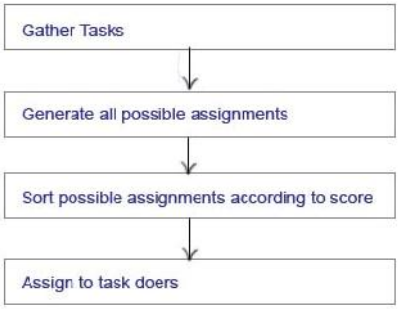
Pemain dalam game TBS tidak bergantung pada kecepatan, refleks, dan akurasi ketika bermain game lainnya karena sifat gameplay yang saling bergiliran dalam bertindak membuat pemainnya cenderung lebih mengandalkan logika dan intuisi dalam mengambil tindakan. Game ini pertama kali muncul dalam video game pada tahun 1977 dan sampai sekarang masih terus dikembangkan.

2.2 Resource Assigment Algorithm

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. Dengan kata lain, RAA adalah algoritma yang bergantung pada nilai resource yang dimiliki oleh objek-objek dalam menentukan sebuah nilai untuk sebuah objek [**4**]. Algoritma ini memiliki tiga parameter utama yaitu objects, resource dan modifier. 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* seperti yang ditunjukan pada Gambar 2.1.

1. *Gather Task*

Gather Task adalah proses pertama dari *Algoritma Resource Assignment* yang mana pada tahap ini adalah menentukan banyaknya objects yang terlibat.



Gambar 2.1
Elemen Algoritma Resource Assignment [**1**]

2. *Generate All Possible Assignments*

Proses ini untuk penentuan nilai modifier yang mana nilai modifier ini akan digunakan untuk mengambil keputusan terhadap objects. Proses penentuan nilai modifier disini akan melibatkan resource yang terdapat pada objects. Nilai modifier terdapat pada setiap keputusan. Setiap keputusan tersebut akan diujikan pada setiap objects. Pengujian dilakukan berdasarkan aturan yang akan diterapkan pada sistem. Setiap terdapat aturan yang terpenuhi, maka nilai modifier untuk keputusan pada aturan tersebut akan ditambahkan.

3. *Sort Assignments According to Score*

Proses ini adalah proses pengurutan setiap nilai modifier pada objects. Proses pengurutan ini bertujuan menentukan nilai modifier terbesar.

4. *Assign to Task Doers*

Proses untuk menjalankan keputusan dengan memperhatikan nilai modifier terbesar yang didapatkan dari *tahap Sort Assignments According to Score.*

3. ANALISIS

3.1 Analisis Game

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.

1. *Storyline*

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 Serikat dan 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 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.

2. 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.

a. Deskripsi Alur Permainan

Pada game ini alur permainan sangat bergantung pada keputusan player sehingga alur permainan tidak dapat dipastikan. 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.

b. 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 :

1) Player membuat keputusan terhadap sumber daya yang dimilikinya yang berupa *unit* ,*resource*, dan *node*.
2) 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 :

1) Jenis keputusan yang dapat diambil adalah *pass, build unit, attack, assign unit*.
2) Pass adalah keputusan player untuk tidak melakukan apa-apa pada *turn*-nya.
3) Build unit adalah keputusan untuk membuat unit, unit dapat dibuat dengan uang yang dimiliki player.
4) Attack adalah keputusan menyerang *node* netral atau *node* milik musuh dengan mengirimkan beberapa unit yang ada ke *node* tersebut.
5) *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*.

c. Deskripsi Kondisi Pada Game

Kondisi pada game adalah keadaan yang mungkin terjadi dalam game. Berikut adalah kondisi tersebut.

1) Keadaan awal (*starting condition*)
   Pada saat memulai game, player dan AI diberikan kondisi sebagai berikut :
   (a) Terdapat 1 *node* yang telah dikuasai.
   (b) Diberikan sejumlah resource untuk memproduksi unit.
   (c) Diberikan sejumlah unit untuk mengusai atau mempertahankan *node*.
2) Keadaan dalam game (*in-game condition*)
   Pada saat bermain, player harus melakukan tugas-tugas sebagai berikut :
   (a) Mempertahankan *node* yang dimiliki.
   (b) Menyerang dan mengambil alih *node* musuh atau *node* netral.
   (c) Memproduksi unit.
   (d) Tidak melakukan apa-apa (*skip turn*)
3) Keadaan akhir (*final condition*)

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

3. Sistem Penilaian

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.

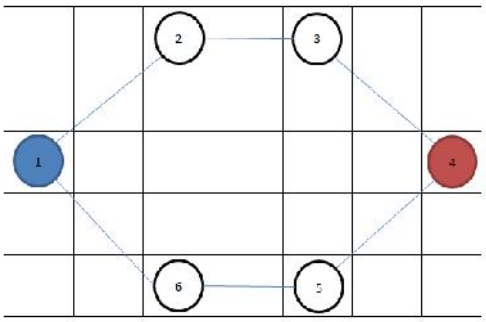
3.2 Analisis Kebutuhan Games

Pada *game* ini terdapat objek-objek dan *resource* yang akan berperan dalam jalannya permainan ini. Objek dan *resource* dapat dilihat pada Tabel 3.1.

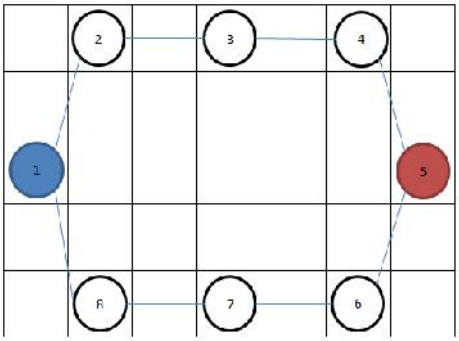
Tabel 3.1. Objek dan Resource Permainan

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 3.2. Penggambaran inisialisasi map pada permainan ini dapat ini dapat dilihat pada Gambar 3.2 dan Gambar 3.3.



Gambar 3.2 *Map Stage 1 dan Stage 2*



Gambar 3.3 *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 3.2 terlihat bahwa setiap node memiliki tetangga yang ditunjukan pada tabel berikut.

Tabel 3.2 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 .33 terlihat bahwa setiap node memiliki tetangga yang ditunjukan pada tabel berikut.

Tabel 3.3 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

3.3 Analisis Algoritma

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.

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 :

1) Keputusan menyerang
Pada keputusan ini, dilakukan pemeriksaan kondisi pada *node* sebagai berikut :
(a) 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.
(b) 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.
(c) 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.
(d) Nilai *modifier* akhir untuk keputusan ini adalah jumlah dari nilai *modifier* pada kondisi yang cocok dibagi dengan jumlah seluruh kondisi yaitu 6.

2) Keputusan bertahan
Pada keputusan ini, dilakukan langkah sebagai berikut :
(a) 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.
(b) 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.

(c) 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.

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

3) Keputusan memproduksi unit

Pada keputusan ini, dilakukan langkah sebagai berikut :

(a) 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.

(b) 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.

(c) 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.

(d) 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.

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

4) Keputusan melewatkan giliran

Pada keputusan ini, dilakukan langkah sebagai berikut :

(a) 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.

(b) 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.

(c) 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.

(d) 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.

(e) 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.

4. IMPLEMENTASI & PENGUJIAN

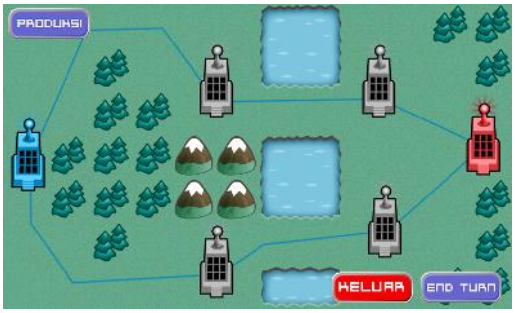
4.1 Implementasi Sistem

Game ini dibuat menggunakan Construct 2 R95 sehingga game ini merupakan game berbasiskan HTML 5. Implementasi game kedalam *smartphone* berbasiskan android menggunakan PhoneGap sebagai *tool*-nya. Perangkat keras yang digunakan untuk mengimplimentasikan dan menguji game ini memiliki spesifikasi sebagai berikut :

1. *Processor* dengan kecepatan 2,3GHz
2. Memori 2GB
3. *Harddisk* 250GB
4. *Video Card* dengan memori 256MB
5. Monitor
6. *Mouse* dan *Keyboard*
7. *Speaker*

4.2 Implementasi Antar Muka

Implementasi antar muka adalah bagian yang menunjukan bentuk tampilan setiap antarmuka pada aplikasi yang sudah dibangun. Implementasi antar muka pada game ini disesuaikan dengan kebutuhan sistem. Berikut adalah salah satu antar muka pada stage 1 ditunjukan pada Gambar 4.1



Gambar 4.1 Antarmuka Stage 1

Beberapa antarmuka selama permainan seperti menyerang, mempertahankan node dan produksi unit.

Gambar 4.2 Antarmuka Menyerang

Gambar 4.3Produksi unit

Gambar 4.4 Mempertahankan Node

4.3 Pengujian Sistem

Rencana pengujian perangkat lunak ini dibagi menjadi 2 bagian yaitu pengujian *black box* dan pengujian *white box*. Pengujian *white box* dilakukan untuk menguji kesalahan logik dan asumsi yang tidak tepat pada kemungkinan eksekusi. Pengembangan kode-kode program selalu memungkinkan terjadinya alur program yang tidak dan tereksekusi dan kesalahan *typography* yang sulit ditemukan kalau tidak dijalankan. Pada umumnya pengujian ini dilakukan hanya pada lingkungan developer.

Pengujian *black box* dilakukan untuk mencari fungsi-fungsi program yang tidak benar atau hilang. Kesalahan lain yang mungkin terjadi dalam pembuatan program adalah kesalahan dalam struktur data, kesalahan antar muka, inisialisasi dan akhir program dan kesalahan performansi.

Strategi pengujian akan dilakukan dengan menggunakan pengujian *Beta*. Pengujian *beta* adalah pengujian pada lingkungan pengguna. Pengujian ini dilakukan dengan menggunakan metode pengumpulan data melalui kuesioner. Pengguna akan mencoba game ini, kemudian pengguna akan memberikan *feedback* pada kuesioner yang telah disediakan . Deskripsi rencana pengujian dapat dilihat dalam Tabel 4.1.

Tabel 4.1 Rencana Pengujian

No	Kelas Uji	Detail Pengujian	Jenis Pengujian
1.	Uji algoritma *resource assignment*	Pengujian pada proses penentuan nilai keputusan pada algoritma	*White Box*
2.	Uji antarmuka Menu Utama	Menekan tombol main	*Black Box*
		Menekan tombol petunjuk main	*Black Box*
3.	Uji antarmuka Menu Main	Menekan tombol 1	*Black Box*
		Menekan tombol 2	*Black Box*
		Menekan tombol 3	*Black Box*
		Menekan tombol kembali	*Black Box*
4.	Uji antarmuka Menu Petunjuk Main	Menekan tombol ►	*Black Box*
		Menekan tombol ◄	*Black Box*
		Menekan tombol kembali	*Black Box*
5.	Uji antarmuka Stage	Menekan tombol produksi	*Black Box*
		Menekan tombol end turn	*Black Box*
		Menekan tombol Keluar	*Black Box*
		Memilih node netral	*Black Box*
		Memilih node musuh	*Black Box*
		Memilih node kawan	*Black Box*
6.	Uji antarmuka Menyerang Node	Menekan tombol tambah (+)	*Black Box*
		Menekan tombol kurang (-)	*Black Box*
		Menekan tombol end turn	*Black Box*
		Menekan tombol kembali	*Black Box*
7.	Uji antarmuka Mempertahank an Node	Menekan tombol ►	*Black Box*
		Menekan tombol ◄	*Black Box*
		Menekan tombol end turn	*Black Box*
		Menekan tombol kembali	*Black Box*
8.	Uji antarmuka Memproduksi Unit	Menekan tombol tambah (+)	*Black Box*
		Menekan tombol kurang (-)	*Black Box*
		Menekan tombol end turn	*Black Box*
		Menekan tombol kembali	*Black Box*

Pengujian dengan kuesioner juga dilakukan untuk mengetahui penilaian kualitatif pada game ini.

5. PENUTUP

Bagian ini merupakan kesimpulan dari keseluruhan penelitian yang dilakukan.

Berdasarkan hasil pengujian, maka kesimpulan pada penelitian ini adalah :

1. Penggunaan algoritma resource assignment dapat memunculkan kemampuan berfikir dan berstrategi pada AI yang terdapat di *game turn-based strategy*.
2. Game turn-based strategy cocok diimplementasikan pada *platform mobile* sehingga dapat dimainkan kapan saja dan dimana saja.

DAFTAR PUSTAKA

[1] Ed Welch. (2007, July) GAMASUTRA. [Online]. http://www.gamasutra.com/view/feature/1535/designing_ai_algorithms_for_.php

[2] Amnon Meisels and Ella Ovadia, "Assigning Resource to Constrained Activities".

[3] Scott Rigby and Richard M Ryan, *GLUED TO GAMES, How Video Games Draw Us In and Hold Us Spellbound*. California, United States of America: Greenwood, 2011.

[4] Rajendra Akerkar and Priti Sajja, *Knowledge-Based System*. United States of America: Jones and Barlett, 2010