Penerapan Algoritma Greedy dalam Game Final Fantasy Tactics

Unggul Bhakti Muhammad/13509079
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jl. Ganesha 10 Bandung 40132, Indonesia
Unggul.bhakti@students.itb.ac.id

***Abstract*—Final Fantasy Tactics merupakan salah satu game yang dirilis oleh square(sekarang square enix) dengan platform Sony Playstation. Game ini berbasis RPG(Role Playing Game) dengan gameplay *turn base system*. Pada game ini *turn base system* dilakukan dengan memberi kotak-kotak dimana player dapat bergerak kesana maupun penentu jarak antara karakter dengan musuh. Karena dalam game ini dilakukan pemilihan aktivitas berdasarkan giliran, tentu berbagai aspek perhitungan akan menjadi pertimbangan player. Dalam makalah ini akan dibahas tentang bagaimana penerapan algoritma greedy untuk melakukan pemilihan aktivitas. Algoritma ini adalah algoritma populer yang disering digunakan untuk optimasi.**

***Kata Kunci*—Final Fantasy Tactics, RPG, Turn Base, Greedy**

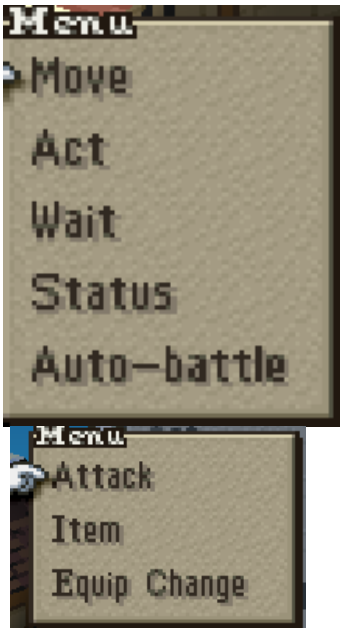
I. PENDAHULUAN

Final Fantasy Tactic adalah permaianan bertipe *role playing game* dengan gameplay *turn base system.* Permainan ini dibuat oleh squaresoft(square enix sekarang) yang dirilis pada tahun 1997 untuk platform Sony PlayStation dan merupakan permainan pertama dari seri Final Fantasy Tactics.

Gambar 1. Gameplay dalam final fantasy tactics

Gameplay yang digunakan dalam gam\e ini merupakan salah satu dari tipe *turn base*. Dalam game ini terdapat beberapa area berupa kotak-kotak yang merupakan intrepertasi dari tempat karakter berada, berbeda dengan pada seri-seir sebelumnya dimana karakter pemain selalu pada satu sisi, dan musuh pada sisi yang berlawanan. Dengan menggunakan dimensi tiga dengan menggunakan isometric karaketer menjadi dapat bergerak ke tempat lain berdasarkan petak dimana dia berada.

Dari gameplay yang ada maka selanjutnya player akan diminta untuk melakukan 2 hal pada gilirannya. Pertama player dapat melakukan move, yaitu melakukan perpindahan petak. Kedua player dapat memilih antara act, dan wait.



Gambar 2. Pilihan untuk player pada gilirannya

Pada tiap pilihan dari player akan tergantung dari status yang dimiliki baik mulai dari menyerang, skill, dan move. Untuk serangan akan dilihat dari beberapa aspek mulai dari jenis senjata, *brave*, job, zodiak, dll. Sedangankan untuk bagian skill akan berkait dengan job saat ini dan job pilihan yang digunakan. Dan untuk move akan dilihat dari status move masing-masing player. Tetapi tidak semua itu murni bergantung dari status karakter, karena ada beberapa skill yang membolehkan untuk mengabaikan

perhitungan-perhitungan dasar yang ada.

Dalam permainan ini secara garis besar ada 3 jenis cara penyerangan, jarak dekat, jarak jauh, dan skill. Untuk itu meskipun ada banyak job yang tersedia dalam hal ini akan kita batasi. Jarak dekat berarti karakter bisa menggunakan senjata jarak dekat: hand, sword, spear. Jarak jauh berarti karakter mampu menyerang jarak jauh secara fisik dengan menggunakan senajata tertentu misal: bow, gun. Skill berarti karakter memiliki kemampuan menyerang khusus tergantung dari job yang dipilih.

Gambar 3. Status yang dimiliki player

Jika dilihat dari gambar 3 maka kita bisa menyimpulkan bahwa karakter dapat:

1. Berjalan sejauh 5 petak
2. Mampu loncat untuk ketinggian 3
3. Mempunyai serangan fisik 5
4. Mempunyai serangan skill 5
5. Memiliki job squire
6. Serangan jarak dekat

Gambar 4. Pergantian job

Sedangkan dapat kita lihat dari gambar 4 bahwa karakter bisa berubah job tergantung dari tim bagaimana yang kita inginkan.

II. METODE

Algoritma greedy adalah algoritma favorit yang diambil jika kita menemui suatu masalah tentang optimalisasi. Algoritma ini juga akan memberikan jawaban langkah perlangkah. Dengan algoritma ini kita akan menentukan apa yang sebaiknya dilakukan pada tiap gilirannya.

Pada algoritma in ada beberapa hal penting yang harus diperhatikan:

1. Pada setiap langkah kita akan membuat solusi optimum lokal.
2. Dengan harapan optimum lokal ini akan mengacu ke optimum global

Dengan kata lain pada tiap langkahnya kita akan mengambil pilihan yang terbaik pada saat itu mengacu pada greedy seperti apa yang akan kita pilih, dengan kata lain mempunyai prinsip “take what you can get now”.

Dalam permainan ini ada beberapa jenis greedy yang bisa diimplementasikan, tapi disini kita hanya akan membahas 2 dari banyaknya metode.

A. Greedy berdasarkan kemampuan

Pada tiap karakter kita sudah mengetahui seberapa besar dia mampu menyerang dengan serangan fisik dan seberapa besar dia mampu mengeluarkan skillnya berdasarkan gambar 3.

Untuk memilih langkah terbaik tentu saja kita harus memikirkan hal terbaik apa yang bisa dilakukan oleh karakter. Untuk itu kita akan melakukan penyerangan berdasarkan hal terbaik apa yang dimiliki oleh karakter.

Pertama kita akan membagi karakter-karakter kita menjadi 2 tipe berdasarkan kemampuan mereka jika status fisik mereka lebih baik maka kita akan menyerang secara fisik, sedangkan jika status mereka lebih baik melakuakan skill. Dengan adanya 2 kondisi tadi artinya kita sudah memiliki algoritma pemisah antara fisik dan non fisik. Tetapi kita dapat satu masalah lagi, bagaimana dengan karakter yang baik untuk serangan fisik maupun non fisik seperti gambar 3? Dalam hal ini kita dapat memberi asumsi bahwa kita lebih mengutamakan fisik daripada non fisik.

Kedua adalah memisahkan lagi mereka terhadap job yang dimiliki. Jika kita perhatikan dengan baik, maka kita dapat melihat bahwa ada dua tipe karakter berdasarkan masing-masing jobnya, pertama adalah tipe support dan kedua adalah tipe damager. Untuk tipe pertama bisa kita lihat dari job misal: priest. Sedangkan untuk yang kedua bisa kita lihat dari job misal:wizard.

Setalah kita berhasil membagi kita akan menemukan satu masalah besar yaitu move. Setiap karakter memiliki perannya masing-masing, untuk itu kita harus mampu menempatkan masing-karakter pada tempatnya masing-masing. Untuk hal ini kita dapat membagi karakter berdasarkan jarak serang dan potensinya. Misalkan untuk job knight dengan kemampuan fisik 7 dan non fisik 3 dengan senjata pedang, maka kita secara otomatis akan bergerak menuju musuh dan mendekatinya. Hal ini akan

berbeda dengan job archer dengan kemampuan fisik dan non fisik yang sama hanya dengan senjata berbeda yaitu bow, disini karakter akan berusaha mencari jalan dimana serangan mereka masih dapat kena dengan jarak terjauh. Sedangkan untuk support hal yang harus dijaga adalah agar skill mereka mampu mengenai teman pada jarak yang terjauh. Untuk kasus move maka kita akan melakukan dengan target karakter musuh yang paling dekat dengan kita. Sehingga jika kita bisa menyerang maka kita akan menyerang.

Tentu saja khusus untuk yang akan melakukan skill maka kita juga harus memperhatikan skill apa yang harus dikeluarkan, untuk itu kita memerlukan sebuah list tentang urutan skill. Untuk penyerang tentu kita mengurutkan berdasarkan serangan terbesar yang kena terhadap musuh. Sedangkan untuk yang bertipe support maka kita akan utamakan menggunakan skill yang belum pernah kita berikan pada karakter tertentu, misal jika kita memiliki karakter priest yang telah memberikan skill reraise ke karakter knight maka kita tidak usah mengulangi hal itu. Dengan kata lain kita akan memiliki 2 pilihan yaitu memberikan skill itu pada orang lain yang belum memiliki atau memberikan skill lain yang belum didapat oleh orang tersebut, sehingga untuk supporter kita memiliki 2 list, yaitu list karakter dan satu lagi list skill.

Dari penjelasan diatas maka kita bisa melihat adanya percabangan pilihan tergantung dari karaketer terpilih. Jika hanya melalui tulisana maka kita akan kurang mencernanya. Untuk lebih jelasnya maka kita dapat melihat keseluaran dari algoritma ini. Pseudo code untuk algoritma ini adalah sebagai berikut:

```
if(fisik >= nonFisik){
    if(jarakDekat){
        moveToNearestEnemy();
        if(canAttack){
            attack();
        }
    }
    else{
        moveToFurthestAttackRange(enemy);
        if(canAttack){
            attack();
        }
    }
}
else{
    if(support){
        moveToFurthestSkillRange(friend);
        if(canSkill){
            Skill(listC,listS);
        }
    }
    else{
        moveToFurthestSkillRange(enemy);
        if(canSkill){
            Skill(list);
        }
    }
}
```

B. Greedy berdasarkan menyerang

Tentu kita semua pernah mendengar "pertahanan terbaik adalah menyerang". Dengan adanya satu kalimat itu maka algoritma ini tidak membutuhkan kita untuk berpikir tentang karakter, hanya cukup dengan menyerang, menyerang, dan menyerang.

Jika kita sudah memutuskan untuk menyerang, masalah kedua adalah siapa yang diserang? Untuk menjawab hal itu mari kita berpikir tentang cara penyerangan.

Cara penyerangan yang ada dalam jenis game ini ada 2 yaitu :

1. Serang satu orang secara fokus
2. Serang semua orang secara merata

Jika kita hanya melihat dari luar maka keliahatannya cara kedua adalah yang terbaik, karena serangan akan terkena secara penuh tidak dengan yang pertama. Misal kita mempunyai 2 karakter dengan daya serang sebanyak 30 untuk masing-masing karaker dan ada 2 musuh dengan darah masing-masing 40. Jika menggunakan cara partama maka 20 serangan yang terakhir akan sia-sia, sedangkan pada serangan kedua maka tidak ada yang sia-sia dan berharap dari karakter kita ada yang memiliki serangan area. Tetapi itu salah, karena itu akan membutuhkan giliran selanjutnya dan bisa saja 2 karakter itu akan memfokuskan serangan ke pemilik serangan area tersebut.

Sedangkan jika kita lihat dari giliran terlihat bahwa cara 1 akan lebih efektif. Misal dengan contoh yang sama ketika musuh sudah mati 1 maka 1 orang musuh ini hanya bisa menyerang 1 kali sedangkan kita masih memiliki turn berikutnya untuk membunuh 1 orang yang tersisa ini. Untuk itu pada algoritma ini akan kita dahulukan untuk menyerang secara fokus.

Ketika melakukan penyerangan belum tentu serangan fisik merupakan yang terbaik maka dari itu untuk pemilik serangan non fisik seperti job wizard maka kita juga lebih memfokuskan ke bagian penyerangan menggunakan skill.

Setelah kita berbicara tentang cara menyerang sekarang kita akan memilih bagainama cara menuju target. Dalam game ini formasi awal kitalah yang menentukannya. Disini ada dua cara menyerang yaitu serangan jarak dekat dan serangan jarak jauh. Dengan adanya 2 kondisi ini maka bisa kita katakan bahwa kita harus mengoptimalisasikan serangan. Jika karakter kita merupakan karakter jarak dekat secara otomatis kita akan mencari jalan agar karakter kita bisa mendekat ke musuh dan menyerang musuh. Tetapi akan berbeda dengan karakter yang menggunakan skill atau bertipe jarak jauh. Untuk tipe ini maka kita membutuhkan mereka mampu mengoptimalisasikan kemampuan mereka, yaitu bisa menyerang dari jarak jauh. Berarti kita harus menempatkan mereka di tempat dimana sebisa mungkin merupakan jarak terjauh dimana mereka masih bisa menembak.

Untuk melakukan serangan secara fokus maka kita pertama harus menentukan target terlebih dahulu. Target yang kita ambil disini adalah musuh yang paling dekat

sehingga rencana ini bisa berjalan dengan cepat. Untuk lebih jelasnya maka kita bisa melihat dari pseudo code berikut :

```
if(dekat){
    moveToTarget();
    if(canAttack){
        attack();
    }
}
else{// untuk pemilik magic maupun ranged weapon
    moveToRange();
    if(fisik){
        attack();
    }
    else{
        skill(list);
    }
}
```

III. ANALISIS

Dari 2 algoritma diatas berarti kita mempunyai pilihan, tergantung dari kita ingin bermain seperti apa? Untuk algotritma pertama akan terlihat bahwa permainan akan cenderung lebih defensif karena karakter akan cenderung lebih mengutamakan apa yang terbaik yang mereka bisa lakukan. Sedangkan dengan algoritma kedua kita akan melihat karakter akan terus-menerus menyerang musuh, sehingga akan banyak terjadi pertumpahan karakter dari kedua belah pihak.

A. Kelebihan dan Kekurangan

Dua algoritma ini memiliki kelebihan dan kekurangan. Ketika kita berpikir tentang optimal lokal dengan berharap merupakan optimal global, berarti bisa terjadi kurang optimum.

Kelebihan pada algoritma pertama ada beberapa. Pertama algoritma ini akan berjalan bersama, sehingga jika karakter kita bekumpul maka kerjasama akan lebih terlihat. Kedua dengan algoritma ini maka kemungkinan kita akan kehilangan karakter lebih kecil sehingga kita tidak perlu untuk merekrut karakter baru dan memulai untuk menaikan level dari awal lagi.

Sedangkan kekurangan dari algoritma pertama ada beberapa juga. Pertama jika karakter kita terpisah, karena algoritma ini hanya akan memungkinkan untuk kerja sama maka akan sangat sulit jika karakter kita berada pada posisi yang terpisah-pisah. Kedua adalah jika kita memilki terlalu banyak karakter dengan job yang sama. Dengan begini maka pola serangan yang diberikan bisa sangat monoton, kondisi paling parahnya adalah jika kita hanya mempunyai 5 priest, dimana kelebihan priest hanya untuk support, sehingga ketika MP habis maka hanya tinggal menunggu mati saja.

Selanjutnya untuk algoritma kedua kelebihannya ada beberapa juga. Pertama dengan cara menyerang secara fokus terus-menerus maka waktu yang terpakai untuk satu pertarungan akan semakin sedikit. Kedua adalah ketika satu orang mati dari musuh pada awal maka itu akan menjadi satu keuntungan besar bagi kita karena kondisi akan selalu membuat kita minimal menang 1 karakter.

Tetapi algoritma kedua ini juga memiliki kelemahan. Pertama adalah dengan algoritma ini jika musuh terdekat dari masing masing karakter berbeda yang mengakibatkan akan sulit untuk melakukan penyerangan yang fokus. Kedua adalah dengan menggunakan algoritma ini maka jumlah prajurit yang akan mati dari kita akan banyak, sehingga kita nantinya harus terus menerus melakukan pencarian prajurit sewaan baru dan melakukan leveling dari awal lagi untuk tiap prajurit baru.

B. Analisis Penggunaan Algoritma Greedy

Dengan penggunaan algoritma greedy berarti kita telah membuat suatu kebiasaan yang akan diikuti dengan mempercayai bahwa kebiasaan ini adalah yang terbaik untuk setiap langkahnya. Dibanding dengan algoritma lain tentu algoritma greedy bisa memberikan banyak kemungkinan greedy yang lain untuk satu contoh kasus tergantung dari situasi bagaimana yang diharapkan. Dengan adanya algoritam greedy ini juga maka kita jadi bisa lebih berpikir dalam membentuk tim yang bagaimana untuk menjalani permainan ini. Jika tim yang terbentuk adalah worst case bagi algoritma ini sendiri maka akan sia-sia pula.

Untuk itu pula akan lebih baik jika kita dapat menentukan algoritma mana yang akan kita pakai untuk tiap kasus berdasarakan tim seperti apa yang akan kita bentuk dan formasi awal. Dengan begitu maka algoritma ini bisa menjadi optimal.

IV. KESALAHAN YANG SERING TERJADI

Pada penggunaan algoritma greedy kesalahan yang sering terjadi adalah mengira bahwa greedy dan brute force mirip, padahal tidak. Pada algoritma brute force jalan yang didapat pasti selalu yang terbaik sedangkan pada algoritma greedy hanya akan mencari jalan sesuai dengan kondisi yang diingkan untuk dipenuhi untuk tiap langkahnya.

Tidak ada algoritma greedy yang sempurna untuk semua posisi, pasti ada worst case bagi masing-masing contoh kasus. Jangan sampai kita salah memilih algoritma greedynya sehingga tidak terjadi kesinkronan antara algoritma dan formasi yang dibentuk.

V. KESIMPULAN

Kesimpulan yang didapat dari pembahasan ini adalah bahwa algoritma greedy akan berharap optimum lokal adalah optimum global. Dengan begitu berarti ada worst case dan optimum case bagi setiap algoritma greedy yang di berikan.

Seperti pada kata “greedy” yang berarti serakah, jika kita menemukan adanya kasus dimana penggunakan greedy tidak optimal, maka justru kata greedy ini sendiri menjadi pedang bermata dua bagi kita.

REFERENCES

[1] http://id.wikipedia.org/wiki/Final_Fantasy_Tactics
[2] http://en.wikipedia.org/wiki/Final_Fantasy_Tactics
[3] http://www.gamefaqs.com/ps/197339-final-fantasy-tactics/faqs
[4] http://www.ffcompendium.com/h/ffthub.shtml
[5] http://www.informatika.org/~rinaldi/Stmik/2011-2012/Algoritma%20Greedy.ppt

PERNYATAAN

Dengan ini saya menyatakan bahwa makalah yang saya tulis ini adalah tulisan saya sendiri, bukan saduran, atau terjemahan dari makalah orang lain, dan bukan plagiasi.

Bandung, 9 Desember 2011

Nama dn NIM