Algoritma Pengambilan Skill pada Permainan Final Fantasy X

13510068 Frilla Amanda
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jl. Ganesha 10 Bandung 40132, Indonesia
13510068@students.stei.itb.ac.id

***Abstract*—Final Fantasy X adalah salah satu game konsol PS2 terpopuler yang dikembangkan oleh Square Enix. Seperti kebanyakan game RPG lain, salah satu bagian yang paling menentukan keberhasilan player adalah kustomisasi karakter. Kustomisasi ini bisa mengakibatkan karakter menjadi lemah atau menjadi sangat kuat dalam pertarungan. Karena tujuan dari game ini adalah untuk mengalahkan musuh-musuh yang ada, maka karakter yang dimainkan oleh player dioptimasi sehingga menjadi kuat.**

Dalam makalah ini akan dibahas optimasi kustomisasi karakter dengan menggunakan algoritma greedy. Dengan menggunakan algoritma greedy, skill/perubahan status yang akan diambil merupakan solusi optimal dari kandidat-kandidat solusi yang ada.

***Index Terms*—greedy algorithm, RPG, turn-based system,.**

I. PENDAHULUAN

1.1 Final Fantasy X

Final Fantasy X adalah salah satu seri dari Final Fantasy, Role-Playing Game yang dikembangkan oleh Square Enix pada tahun 2005. Berkisah mengenai seorang pemuda bernama Tidus yang secara tiba-tiba masuk ke dalam sebuah dunia lain bernama Spira. Di dunia ini tersebutlah sebuah entitas penghancur yang disebut dengan nama Sin, Tidus, setelah bertemu dengan seorang summoner bernama Yuna yang memiliki impian untuk membuat dunia yang damai, berusaha untuk mengalahkan Sin. Tujuan akhir dari permainan ini adalah untuk menyelamatkan Spira dari kehancuran yang diakibatkan oleh Sin.

1.2 Gameplay

Karena permainan ini bergenre RPG, di dalamnya ada banyak gameplay termasuk puzzle dan arcade. Namun dalam makalah ini yang akan dibahas adalah mengenai sistem kustomisasi karakternya dalam persiapan *battle*. Final Fantasy X menggunakan sistem *turn-based* dalam *battle*-nya. Pemain diperbolehkan memilih tiga karakter utama dalam tim yang akan dimainkan pada battle. Setiap bar aksi penuh, karakter akan mendapatkan giliran untuk melakukan aksinya.

(Gambar 1a, battle mode)

Terdapat lima jenis aksi yang bisa dijalankan oleh setiap karakter yaitu: Attack, Ability, Magic, Item, dan Run. Attack adalah serangan dengan menggunakan senjata yang sedang dipakai oleh karakter. Aksi attack, item dan run bisa digunakan oleh semua karakter meskipun tingkat keberhasilannya ditentukan oleh status karakter tersebut. Sedangkan aksi ability dan magic masing-masing karakter berbeda sesuai dengan profesi dan kemampuan yang diambil oleh karakter tersebut. Setiap karakter memiliki statusnya masing-masing yaitu Strength (STR), Magic(MAG), Agility (AGI), Defense (DEF), Magic Defense/Spirit (SPR), Accuration (ACC), Evasion (EVA), Luck (LUK), Magical Point (MP), dan Health Point (HP).

Terdapat tujuh macam profesi pada permainan ini yaitu swordman, broad-sword user, summoner, lancer, magician, thief, dan athlete. Masing-masing memiliki keunggulannya masing-masing. Swordman adalah profesi paling aman, ia menyerang menggunakan senjatanya (attack) dan ability yang dimilikinya adalah support seperti memperlambat gerakan musuh atau justru mempercepat gerakan timnya. Broad-sword user adalah profesi yang paling kuat daya serangnya, ia menyerang menggunakan senjatanya dan ability yang dimilikinya adalah menghancurkan pertahanan musuh dengan melemahkan defense-nya.

Summoner tidak memiliki daya serang yang kuat, namun memiliki magic yang bisa menyembuhkan kawan dan ability summon untuk memanggil mahluk magis untuk menjadi pengganti tim dalam melawan musuh.

Lancer mirip seperti swordman, tetapi senjatanya adalah tombak yang kuat melawan beberapa tipe musuh tertentu. Magician memiliki skill magis elemental serta memiliki agility yang tinggi sehingga mudah menghindar dari serangan lawan.

Keunggulan dari profesi thief adalah abilitynya untuk mengambil barang, uang, maupun experience point dari musuh. Selain itu Thief juga memiliki abilty untuk menggunakan barang-barang yang tidak bisa digunakan oleh profesi lain. Yang terakhir adalah athlete yang menyerang dengan menggunakan bola. Keunggulannya adalah memiliki ability untuk menurunkan daya serang lawan.

Kustomisasi karakter yang dibahas pada makalah ini adalah pengambilan skill dan perubahan status setiap karakter. Terdapat sebuah pohon skill yang disebut Sphere Grid di mana tiap simpulnya memiliki kemampuan masing-masing. Sphere Grid inilah yang menentukan akan jadi seperti apa karakter yang ada.

II. Dasar Teori

Algoritma Greedy adalah algoritma yang paling populer digunakan untuk persoalan yang menyangkut optimasi dari suatu hal. Persoalan optimasi adalah persoalan yang bukan hanya mencari penyelesaiannya, melainkan mencari penyelesaian terbaik yang ada untuk mendapatkan hasil yang maksimal. Persoalan optimasi dibagi menjadi dua macam yaitu maksimasi dan minimasi.

Sesuai dengan namanya Greedy (tamak) adalah algoritma yang selalu mengambil pilihan yang paling bagus atau yang paling mahal.

Algoritma greedy membentuk solusi pertahap (langkah demi langkah). Langkah yang sudah diambil tidak bisa diubah lagi selanjutnya. Setiap langkah yang diambil adalah langkah optimal lokal, karena mungkin saja dengan mengambil solusi yang bukan paling mahal kemudian selanjutnya bisa mendapatkan solusi yang lebih baik. Meskipun belum tentu mendapatkan hasil yang paling baik, algoritma greedy dipakai dengan harapan bahwa solusi optimal lokal tersebut akan menjadi solusi optimal global.

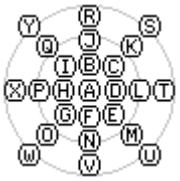
Skema umum algoritma greedy terdiri dari:

1) Himpunan Kandidat: kumpulan dari elemen-elemen yang membentuk solusi
2) Himpunan Solusi: kumpulan dari kandidat-kandidat yang terpilih menjadi solusi
3) Fungsi seleksi: fungsi yang memilih dari semua kandidat yang ada, elemen yang memungkin untuk mencapai solusi optimal
4) Fungsi kelayakan: fungsi yang memeriksa apakah solusi yang diambil memenuhi constraint atau tidak. Jika tidak, maka solusi akan dibuang.
5) Fungsi objektif: fungsi yang memaksimumkan atau meminimumkan nilai solusi.

III. Penerapan Algoritma

A. Sphere Grid

(Gambar 3a. menu sphere grid)



(Gambar 3b. sebagian kecil dari sphere grid)

Pada Final Fantasy X, ketika berhasil mengalahkan musuh maka karakter akan mendapatkan experience point (EXP). Ketika EXP ini cukup untuk membuat karakter tersebut naik level, maka karakter tersebut akan menerima AP. AP adalah point yang digunakan untuk bergerak pada sphere grid. Karena player hanya bisa mengaktifkan simpul yang sedang dilewati, AP adalah salah satu bagian terpenting dalam pengambilan skill.

Seperti yang terlihat pada gambar 3b, sphere grid terdiri dari dua bagian yaitu path (garis abu-abu) dan node. Setiap node memiliki pengaruhnya masing-masing kepada karakter. Secara garis besar terdapat empat jenis node yaitu empty node yang hanya bisa dilewati, locked node yang memerlukan sphere kunci untuk bisa dilewati, status node yang bisa mempengaruhi status karakter, dan ability node yang bisa membuat karakter mempelajari ability/skill/magic baru.

Untuk mengaktifkan node, dibutuhkan item yang disebut dengan sphere. Sesuai dengan jenis node-nya, sphere juga memiliki beberapa tipe yaitu: sphere-key untuk membuka lock, attack-sphere untuk mengaktifkan node STR, HP, dan DEF, speed-sphere untuk mengaktifkan node EVA dan AGI, magic-sphere untuk mengaktifkan MAG, MP, dan SPR, luck-sphere untuk mengaktifkan LUC, dan ability-sphere untuk mengkatifkan node ability/skill/magic. Sphere, seperti AP, juga didapatkan dengan cara mengalahkan musuh.

Jumlah stiap jenis sphere terbatas sehingga tidak mungkin semua karakter (player memiliki total 7 karakter dalam permainan ini) bisa mengaktifkan semua simpul yang ada.

Dengan asumsi bahwa semakin bagus suatu node maka jumlahnya akan semakin sedikit, maka rumus untuk menentukan nilai dari sebuah node adalah:

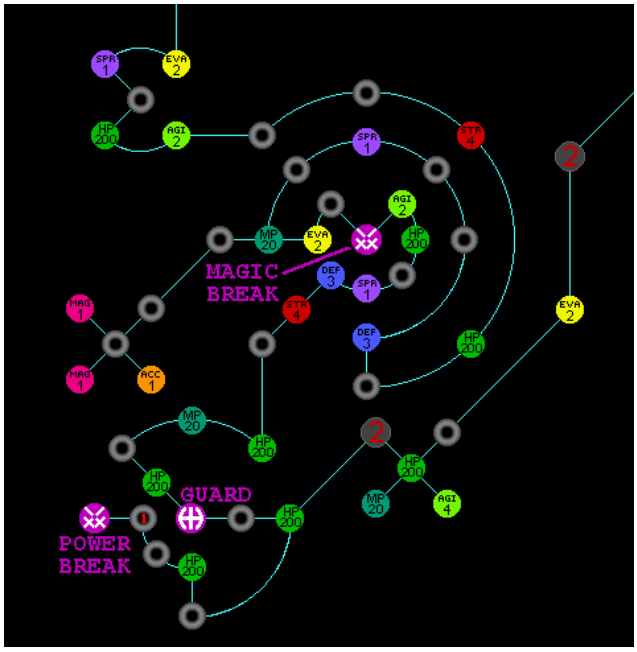
$$N/n$$

di mana n adalah jumlah dari node jenis tersebut dan N adalah jumlah seluruh node tidak kosong yang ada pada grid. Nilai dari N adalah 563.

Node yang ada pada grid tersebut:

Jenis Node	Jumlah	Nilai
STR4	23	24.47
STR3	5	112.6
STR2	26	21.65
STR1	11	51.18
MAG4	17	33.11
MAG3	14	40.21
MAG2	12	46.91
MAG1	2	281.5
DEF4	12	46.91
DEF3	19	29.63
DEF2	21	26.80
DEF1	1	563
SPR4	16	35.18
SPR3	6	93.8
SPR2	16	35.18
SPR1	6	93.8
AGI4	21	26.80
AGI3	15	37.53
AGI2	20	28.15
AGI1	4	140.75
EVA4	19	29.63
EVA3	4	140.75
EVA2	17	33.11
EVA1	2	281.5
ACC4	4	140.75
ACC3	8	70.375
ACC2	13	43.3
ACC1	4	140.75
LUK4	2	281.5
LUK2	1	563
LUK1	6	93.8
HP	97	5.8
MP40	7	80.42
MP20	48	11.7
MP10	1	563
LOCK4	11	51.18
LOCK3	11	51.18
LOCK2	17	33.11
LOCK1	24	23.4
Magic/Ability	1 (unique)	563

B. Algoritma Greedy I



(Gambar 3c. sphere grid)

Pada algoritma greedy yang pertama ini berfokus pada sphere mana yang terlebih dahulu diaktifkan, namun pada akhirnya karakter akan mengaktifkan semua simpul dengan tidak mempertimbangkan *job*-nya. Semua simpul yang dilewati akan diaktifkan tanpa mempertimbangkan resource yang harus digunakan.

Simpul awal adalah simpul dengan angka satu bewarna merah. Dari simpul awal terdapat dua simpul yang bisa dikunjungi yaitu simpul ability Power Break dan yang satunya adalah simpul kosong. Menggunakan algoritma greedy, maka simpul yang akan dikunjungi adalah simpul abilty yang memiliki nilai lebih besar. Simpul berikutnya adalah simpul awal, kemudian simpul kosong, HP200, simpul kosong, kemudian HP200. Kemudian kandidat untuk simpul berikutnya adalah simpul kosong atau LOCK2. Karena simpul kosong nilainya lebih besar maka simpul yang dipilih adalah simpul kosong, kemudian ability Guard, HP200, dan seterusnya.

Adapun algoritma greedy yang dipakai adalah:

```
*
function pilihSimpul(input C:
himpunan_kandidat)→ himpunan_solusi

Deklarasi:
    x : kandidat
    S : himpunan_solusi

Algoritma:
    S ← {}
    while (not SOLUSI(S)) and (C ≠ {} ) do
       x ← SELEKSI(C)
       C ← C - {x}

       if LAYAK(S ∪ {x}) then
           S ← S ∪ {x}
       endif
    endwhile { solusi(S) atau C={} }
```

```
    if SOLUSI(S) then
       return S
    else
       { do nothing }
    endif
```

Himpunan kandidatnya adalah simpul-simpul yang ada pada grid. Fungsi seleksi kandidat dari himpunannya adalah dengan mengambil simpul yang berada di sebelah simpul awal yang memiliki nilai paling besar (solusi optimal lokal).

```
function pilihSimpul(input C:
himpunan_kandidat)→ kandidat

Deklarasi:
    X : kandidat
    Max : nilai

Algoritma:
    Max ← 0
    X ← FirstElement(C)
    while ( C ≠ {}) do
       if (Max < X) and (hasSphere) then
              Max ← X
       else
              X ← Next(C)
       endif
    endwhile

    return X
```

Sedangkan fungsi kelayakannya adalah apakah langkah yang diambil pada solusi ini sebanyak AP yang dimiliki oleh karakter.

```
function solusiLayak(input S: himpunan_solusi
AP: integer)→ boolean

Algoritma:
    if (element(S) =< AP) then
       return true
    else
       return false
    endif
```

Kemudian fungsi objektifnya adalah dengan membandingkan apakah X memiliki cost paling besar daripada kandidat lain (solusi optimal lokal).

C.Algoritma Greedy II

Algoritma greedy yang kedua ini tidak akan mengaktifkan semua simpul yang ada melainkan terlebih dahulu melihat apakah simpul itu diperlukan oleh tipe karakter. Misalkan seorang magician tidak membutuhkan simpul STR untuk aktif karena ia menyerang dengan menggunakan magic yang tidak dipengaruhi oleh status STR sehingga attack-sphere yang dibutuhkan untuk mengaktifkan simpul ini bisa digunakan oleh karakter lain. Begitu juga untuk swordman yang tidak memiliki serangan magic, tidak perlu mengaktifkan simpul MAG-nya karena meskipun dinaikan tidak akan berpengaruh pada daya serangnya.

Himpunan kandidatnya adalah seluruh simpul yang ada pada grid. Fungsi seleksi kandidat dari himpunannya adalah dengan mengambil simpul yang berada di sebelah simpul awal yang memiliki nilai paling besar (solusi optimal lokal) dan kesesuaiannya dengan job dari karakter tersebut (dijelaskan pada bagian 1.2).

```
function pilihSimpul(input C:
himpunan_kandidat)→ kandidat

Deklarasi:
    X : kandidat
    Max : nilai

Algoritma:
    Max ← 0
    X ← FirstElement(C)
    while ( C ≠ {}) do
       if (Max < X) and (sesuaiJob) then
              Max ← X
       else
              X ← Next(C)
       endif
    endwhile

    return X
```

Fungsi kelayakannya sama dengan fungsi kelayakan pada algoritma greedy yang sebelumnya.

```
function solusiLayak(input S: himpunan_solusi,
AP: integer)→ boolean

Algoritma:
    if (element(S) =< AP) then
       return true
    else
       return false
    endif
```

Dengan melihat gambar 3c, andaikan untuk karakter dengan job Magician, maka ability Power Break tidak akan diambil dan pergerakan justru ke simpul kosong. Setelah mengaktifkan dua buah simpul HP200, simpul Guard akan dilewati tanpa diaktifkan. Tetapi untuk karakter dengan job broad-sword user, maka ability Power Break akan diaktifkan begitu juga simpul Guard. Sedangkan simpul MAG (warna magenta) tidak akan diaktifkan karena tidak akan mempengaruhi daya serang karakter.

D. Analisis

Ada banyak cara kustomisasi karakter pada permainan ini. Pemain bebas memilih apakah dia akan menggunakan semua sphere yang dimilikinya dan mengambil semua kemampuan tanpa mempertimbangkan job dari karakter, atau apakah player akan menghemat sphere-nya dan hanya mengambil simpul yang sesuai dengan job karakter. Selain itu masih ada solusi lain misalnya dengan membuat karakter yang focus pada serangan sehingga hanya akan mengaktifkan simpul yang memiliki efek serangan atau sebaliknya.

1) Kelebihan dan Kekurangan

Algoritma yang pertama dibuat dengan pertimbangan semua sphere hanya akan digunakan pada karakter utama (sekitar 3 dari 7) dan karakter lainnya yang tidak dipakai tidak memerlukan sphere-sphere yang ada.

Sedangkan algoritma yang kedua dibuat dengan mempertimbangkan keseimbangan kekuatan tim secara keseluruhan sehingga jumlah sphere yang ada harus dibagikan kepada masing-masing anggota tim karena itu jumlah yang dapat digunakan menjadi terbatas.

2) Penggunaan Algoritma Greedy

Solusi Greedy pada permasalahan ini tidak optimal karena selain tidak memperhitungkan banyaknya pergerakan simpul, solusi greedy akan hampir selalu melewati simpul LOCK yang bernilai minus. Padahal setelah simpul LOCK mungkin saja ada simpul-simpul yang bernilai besar setelahnya.

IV. KESIMPULAN

Kesimpulan yang bisa didapatkan dari pembahasan mengenai algoritma greedy ini adalah bahwa algoritma greedy tidak selalu menghasilkan solusi yang terbaik. Karena algoritma greedy hanya bisa mencari solusi optimal lokal meskipun bukan berarti tidak mungkin bahwa solusi optimal tersebut juga merupakan solusi optimal global. Pada algoritma greedy terdapat worst case yaitu semua simpul bernilai tinggi tersembunyi oleh simpul LOCK yang hampir tidak akan pernah dipilih karena nilainya yang minus. Best case dari algoritma ini adalah jika di path yang ditutupi oleh simpul LOCK adalah simpul-simpul yang bernilai kecil sedangkan simpul yang berada di jalur tanpa LOCK memiliki nilai yang besar/maksimal.

Meskipun algoritma greedy tidak selalu mendapatkan solusi optimal global, namun algoritma ini adalah cara paling sederhana untuk mendekati hasil optimal global.

REFERENCES

www.squareenix.com
www.ffx-europe.com
www.gameinternals.com/post/3364162387/
www.finalfantasykingdom.net/finalfantasyxspheregrid.php

Munir, Rinaldi, "Diktat Kuliah IF3051 Strategi Algoritma", Program Studi Teknik Informatika, 2009.

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, 29 April 2010

Frilla Amanda (13510068)