Algoritma Pathfinding untuk Game

Riefky Amarullah Romadhoni - 13511038
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jl. Ganesha 10 Bandung 40132, Indonesia
riefkyamarullah@students.itb.ac.id

***Abstract*— *Game* atau permainan komputer merupakan suatu sarana hiburan yang digemari banyak orang pada masa kini. Banyaknya jenis dari permainan komputer tersebut menyebabkan banyaknya orang yang menjadikan permainan komputer sebagai sarana hiburan yang cocok untuk menghilangkan rasa jenuh dan bosan. Salah satu jenis permainan tersebut adalah jenis strategi, dimana dalam permainan tersebut biasanya ada unit yang melakukan pergerakan. Pergerakan unit ini dapat dilakukan dengan menggunakan algoritma pencarian graf A* dengan memodifikasi graf yang ada dan juga algoritma pencarian itu sendiri.**

Index Terms*—*Games, Grid Map, Navigation Meshes*, A* *Algorithm

I. PENDAHULUAN

Game atau permainan komputer merupakan suatu sarana hiburan yang digemari banyak orang pada masa kini. Banyaknya jenis dari permainan komputer tersebut menyebabkan banyaknya orang yang menjadikan permainan komputer sebagai sarana hiburan yang cocok untuk menghilangkan rasa jenuh dan bosan. Selain alasan tersebut, masih banyak lagi alasan lain sehingga orang-orang memainkan suatu *game.*

Salah satu jenis permainan tersebut adalah jenis strategi, dimana dalam permainan tersebut dibutuhkan semacam taktik agar dapat memenangkan atau mendapatkan efisiensi maksimum dalam memainkannya. Pada setiap *game* strategi pasti selalu ada unsur yang membedakan cara memainkannya, tetapi ada satu hal yang sama yaitu dalam setiap game tersebut pasti ada cara tertentu atau trik-trik khusus agar kita dapat memainkan *game* tersebut dengan baik dan maksimal. Semua game, dari *casual game* hingga *hardcore game* memerlukan strategi dalam memainkannya. Contohnya, lihat saja game yang sangat digemari sekarang, yaitu game sejenis *Angry Birds* dan *game* lain yang dibuat oleh developer *Angry Birds*. Meskipun *game* tersebut hanya membutuhkan satu jari untuk memainkannya, tetap saja harus menggunakan strategi agar dapat menyelesaikan level-levelnya.

Selain *game* tersebut, tentu saja *game* lain yang lebih kompleks juga membutuhkan strategi dalam memainkannya. Salah satu *game* yang membutuhkan strategi dalam memainkannya yaitu *game* dengan *genre* RTS atau *Real Time Strategy* dan juga TBS atau *Turn Based Strategy*. Seperti namanya, *game Real Time Strategy* merupakan game strategi dimana pemain harus memikirkan strategi dan bertindak dengan cepat dalam waktu *real time* atau langsung. Pemain lain dalam *game* tersebut juga melakukan hal yang serupa yaitu membuat strategi secara langsung. Sedangkan *game Turn Based Strategy* merupakan game strategi dimana pemain diberikan waktu, biasanya terbatas, untuk memikirkan strategi dan bertindak sesuai strategi yang telah ia buat dan bergantian dengan pemain lainnya. Contoh yang mudah untuk *game Turn Based Strategy* ini adalah permainan catur, dimana kedua orang diberikan waktu untuk berpikir dan menggerakan pionnya kemudian bergantian dengan lawannya.

Pada makalah ini akan dibahas mengenai perbedaan aplikasi pathfinding menggunakan algoritma A* pada *game Real Time Strategy* dan *game Turn Based Strategy*, yang selanjutnya akan disingkat oleh *game RTS* dan *game TBS*. *Game* yang akan dibahas adalah *game* Dota 2 yang merepresentasikan *game RTS* dan *game* Civilization V yang merepresentasikan *game TBS*.

II. LANDASAN TEORI

A. Algoritma A*

Algoritma A* merupakan algoritma komputer yang sangat umum digunakan dalam hal *pathfinding* (mencari jalur) dan *graph traversal* (menjelajahi graf) dengan proses merencanakan jalur yang efisien antar node yang ada. Pada umumnya, performansi dan akurasi dari algoritma ini adalah standar tetapi ada algoritma lain yang dapet memproses grafnya terlebih dahulu sehingga mendapatkan hasil performansi yang lebih baik.

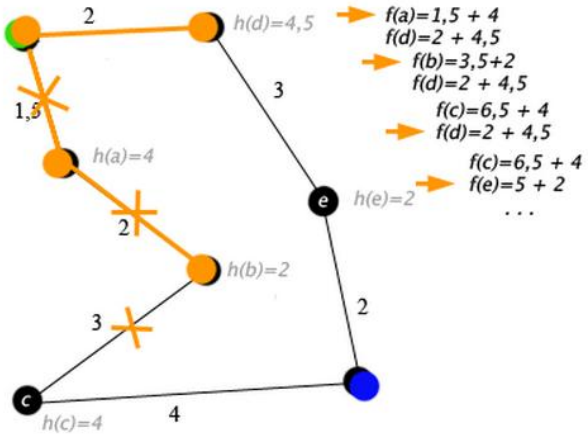
Algoritma A* menggunakan algoritma *Best First Search* dan menemukan cost yang paling kecil yang ditemukan selama masa pencarian dari node awal ke node akhir. Selama algoritma menelusuri graf, ia menelusuri jalur yang memiliki ekspektasi cost atau panjang yang paling kecil, dan menyimpan *queue* atau antrian prioritas dari jalur lainnya sepanjang perjalanan.

Ia menggunakan fungsi *knowledge-plus-heuristic cost* dari node x untuk menentukan urutan pencarian node di dalam pohon. Fungsi tersebut adalah penjumlahan dari 2 fungsi, yaitu :

- *Past path-cost function*, yaitu jarak yang diketahui dari node awal ke node x.
- *Future path-cost function*, yaitu estimasi heuristic jarak dari node x ke node tujuan.

Algoritma A* mempunyai pseudocode sebagai berikut :

```
function A*(start,goal)
// Set dari node yang sudah dievaluasi
closedset := set kosong
// Set dari node yang akan dievaluasi, dimulai
dari node awal
openset := {start}
// Map dari node yang akan dinavigasi
came_from := map kosong

// Cost dari awal dengan path terbaik yang sudah
diketahui
g score[start] := 0

// Estimasi total cost dari awal ke akhir
melalui y
f_score[start] := g_score[start] +
heuristic cost estimate(start, goal)

while openset is not empty
    current := node di openset yang memiliki
               f_score[] terkecil
    if current = goal
        return reconstruct_path(came_from, goal)

    remove current from openset
    add current to closedset
    for each neighbor in neighbor nodes(current)
        tentative_g_score :=
          g_score[current] +
          dist between(current,neighbor)
        tentative_f_score := tentative_g_score +
          heuristic cost estimate(neighbor, goal)
        if neighbor in closedset and
          tentative_f_score >= f_score[neighbor]
              continue
      if neighbor not in openset or
        tentative_f_score < f_score[neighbor]
                came_from[neighbor] :=
current
                g_score[neighbor] :=
                f score[neighbor] :=
                if neighbor not in openset
                    add neighbor to openset

    return failure

function reconstruct path(came from,
current_node)
    if current node in came from
        p := reconstruct_path(came_from,
                came from[current node])
        return (p + current node)
    else
        return current_node
```



Graf 1. Contoh Pohon Pencarian A*

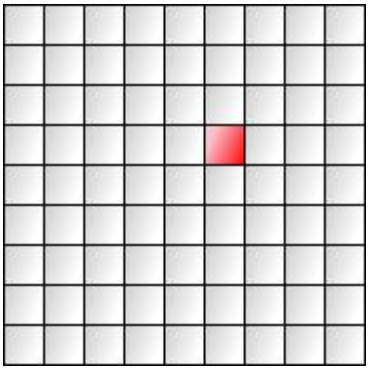
B. Representasi Map

Ada banyak sekali representasi map yang digunakan untuk merancang suatu *game*, dalam makalah ini akan dibahas 2 representasi map yang berkaitan dengan *game* yang akan dibahas yaitu *Grid Map* dan *Navigation Meshes*.

B.1. *Grid Map*

Grid Map merupakan representasi map yang paling umum digunakan oleh *game TBS*. *Grid Map* ini merupakan kumpulan tile atau kotak yang saring berhubungan satu

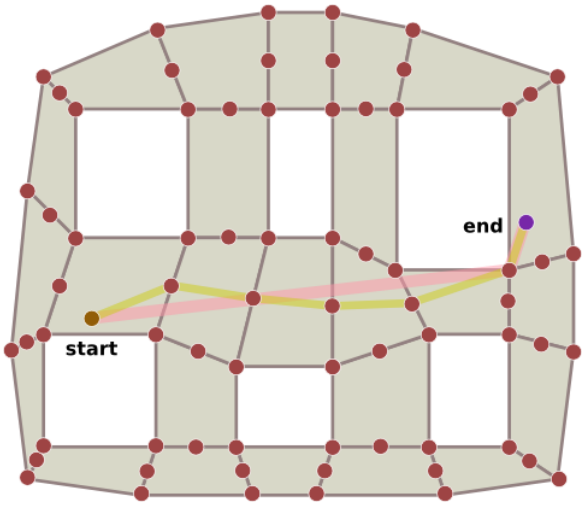
sama lain. Setiap kotak dapat diberikan informasi cost yang diperlukan dan juga dapat diberikan aturan tambahan mengenai pergerakan dari unit yang ada di peta. *Grid Map* ini diterapkan pada *game* Civilization V yang akan kita bahas.



Graf 2. Contoh *Grid Map* sederhana

B.2. *Navigation Meshes*

Navigation Meshes merupakan kumpulan node yang ada di dalam kumpulan poligon. Kumpulan poligon yang saling overlap atau tumpang tindih merupakan area yang tidak dapat dilalui oleh unit. Biasanya pada *game RTS* setiap node memiliki bobot yang sama tetapi ada unit tertentu yang memiliki kelakuan khusus, contohnya *Flying Unit*, merupakan unit yang terbang sehingga dapat melewati area yang tidak dapat dilalui oleh unit biasa. *Navigation Meshes* ini diterapkan pada *game* Dota 2 yang akan kita bahas di makalah ini.



Graf 3. Contoh *Navigation Meshes* sederhana

III. APLIKASI PADA *GAME*

A. Dota 2

Dota 2 adalah game sebuah permainan *desktop* bergenre strategi, yaitu *genre Real Time Strategy*, atau sering disebut *genre* MOBA (*Massive Online Battle Arena)* untuk platform Windows, Linux, dan Mac OS yang dikembangkan oleh Valve. Dalam Dota 2, pemain berperan sebagai seorang hero yang harus bekerjasama dengan 4 hero lainnya untuk dapat mempertahankan *ancient* mereka dan dapat menghancurkan *ancient* musuh. Musuh tersebut juga merupakan kumpulan dari 5 hero. Tiap hero memiliki status atribut dan kemampuan yang unik. Untuk dapat menghancurkan *ancient* musuh pemain harus bekerjasama dengan 4 pemain lainnya dalam menentukan strategi dan juga mengalahkan musuh yang mencoba mempertahankan *ancient* dan juga ingin menyerang *ancient* pemain.

Dota 2 menerapkan sistem *Real Time Strategy* yang artinya semua pemain dapat bergerak secara bersamaan dan pemain yang dapat menentukan taktik dan bertindak paling cepat dan tepat akan dapat mengalahkan musuhnya.

Gambar 1. *Game* Dota 2

Pada setiap sesi permainan selalu dimulai dengan peta yang sama sehingga memudahkan algoritma untuk mengatur jalur yang dilalui oleh unit. Dota 2 menggunakan *navigation meshes* dimana satu unit bisa memiliki banyak node untuk bergerak, contohnya unit hero pada umumnya berukuran 100 unit, sehingga dibutuhkan 100 node pada satu tempat agar hero tersebut dapat bergerak ke tempat tersebut. Seperti dapat terlihat di gambar 1, ada lingkaran dibawah hero, lingkaran tersebut adalah node yang dipakai oleh hero tersebut yang berukuran 100 unit.

Informasi tambahan lainnya yang dimasukan pada *navigation meshes* adalah peta yang berubah-ubah sesuai dengan kondisi yang ada, yaitu pohon yang dapat ditebang dan juga unit yang dapat berpindah tempat. Algoritma A* dapat dipanggil setiap kali pemain menginstruksikan suatu aksi.Aturan pergerakan pada *game* ini simpel yaitu dua unit tidak boleh berada pada node atau unit yang sama, jika ada unit yang menghalangi maka akan dicari jalur yang lain.

Aplikasi algoritma A* dimulai saat map diload pertama kali dan diulang secara terus menerus ketika ada unit yang melakukan aksi. Algoritma A* membentuk sebuah pohon yang memetakan node yang akan dicapai kemudian dicek apakah node tersebut memenuhi syarat, yaitu ukuran node sama atau lebih besar dari ukuran node unit.

Pada game ini juga kita dapat mengendalikan banyak unit sekaligus, dalam hal itu pathfinding akan memberikan jalur yang sama untuk setiap unit dan memberikan *queue* untuk setiap unit berdasarkan urutan posisi yang paling depan.

Gambar 2. Jalur untuk unit biasa pada peta Dota 2

Pada gambar 2 dapat dilihat bahwa untuk mencapai goal maka unit akan melewati jalur berwarna merah. Jalur berwarna hitam tersebut merupakan jalur yang dihitung juga oleh A* dan dieliminasi karena memiliki cost yang lebih besar.

Kasus khusus diterapkan untuk unit terbang, yaitu tidak memperdulikan adanya halangan atau jalur yang tidak bisa dilalui dalam map sehingga untuk kasus diatas, jalur untuk unit terbang sangat sederhana, yaitu menarik garis lurus antar node awal dengan node akhir. Untuk unit terbang ini algoritma A* yang dipakai akan sama dengan algoritma greedy.

Gambar 3. Jalur untuk unit terbang pada peta Dota 2

B. Civilization V

Civilization V adalah sebuah permainan *desktop* bergenre strategi untuk platform Windows dan Mac OS X uang dikembangkan oleh Firaxis Games, dan dipublikasikan oleh 2K Games & Aspyr. Dalam Civilization V, pemain berperan sebagai seorang pemimpin peradaban, dimana sang pemimpin dan peradaban yang dipimpinnya didasarkan pada peradaban asli dunia. Tiap peradaban memiliki keunggulan dan kekurangan yang unik, yang menentukan cara memainkan peradaban tersebut dengan efektif. Sebagai pemimpin, sang pemain harus bisa membawa peradabannya melewati transisi zaman, bersaing dan berdiplomasi dengan peradaban pemain lain, serta memenuhi salah satu dari berberapa kondisi spesifik yang diperlukan untuk memenangkan sebuah sesi permainan.

Civilization V menerapkan sistem *turn-based*, yakni semua pemain (baik itu manusia maupun komputer) menjalankan gilirannya secara bergantian. Pemain pertama melaksanakan gilirannya, diikuti pemain kedua hingga pemain terakhir, setelah itu kembali giliran pemain pertama lagi dan seterusnya. Dalam setiap gilirannya, pemain dapat bergerak untuk mengelola peradabannya agar dapat memenangkan permainan dengan berbagai cara, contohnya membunuh negara lainnya, menjadi pemimpin teknologi, dan lain-lain.

Gambar 4. *Game* Civilization V dengan strategic view

Dapat dilihat dari gambar 2, Civilization V menerapkan sistem *Grid Map* disertai dengan tambahan informasi cost untuk setiap tile yang ada. Tambahan informasi itu adalah sebagai berikut :

Nama Tile	Jenis	Bobot
Hill	Darat	2
Marsh	Darat	2
Jungle	Darat	2
Forest	Darat	2
Grassland	Darat	1
Plains	Darat	1
Desert	Darat	1
Tundra	Darat	1
Ocean	Laut	1
Coast	Laut	1

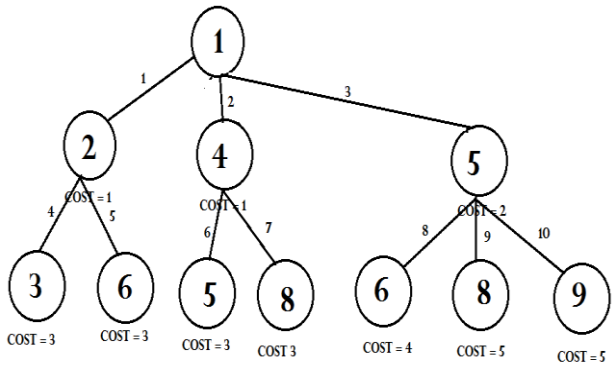
Tabel 1. List Tile dan Costnya

Setiap unit mempunyai move point yang akan dikonsumsi saat bergerak ke tempat lain, jumlah konsumsi tersebut bergantung pada jenis tile yang dilewati dan juga hanya unit-unit tertentu yang dapat melewati tile jenis Darat atau Laut.

Modifikasi dari algoritma A* disini adalah dengan mengecek cost. Pengecekan pertama dilakukan dengan cara mengecek apakah cost ke node tersebut kurang dari movement point dari unit, jika kurang maka pencarian dapat diteruskan jika tidak maka unit tidak dapat bergerak ke tempat tujuan tersebut. Pengecekan kedua saat cost dari node sama dengan movement point dari node, akan dicek apakah node tersebut adalah node tujuan, jika node tujuan berarti jalur ditemukan dan akan diberikan pergerakan unit, jika tidak maka unit tidak bisa bergerak ke tempat tersebut.

Gambar 5. Contoh Tile pada Civilization V

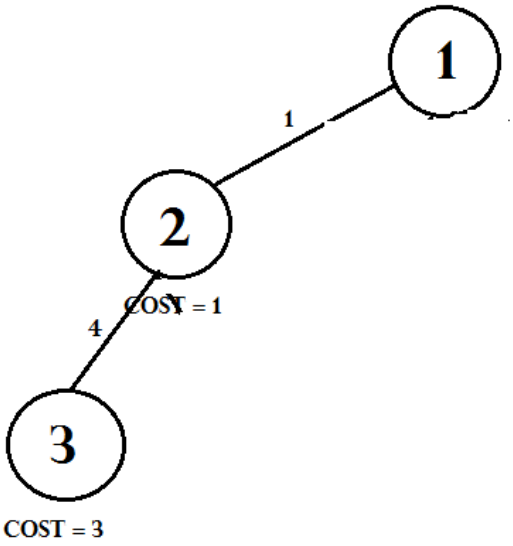
Dari gambar 3 diatas, dapat dibuat menjadi gambar 5 dengan penomoran tile dari nomor 1 sampai 15. Tile [1,2,4] adalah *Plains*, tile 14 adalah grassland, tile [3,8,9,11] adalah hill, dan sisanya adalah forest. Misalnya suatu unit pada posisi 1 yang dapat melewati tile darat, memiliki movement point 3, dan pemain ingin bergerak ke tile 15 maka akan diberlakukan algoritma A* dengan start node 1 dan goal node 15, akan dibentuk tree sebagai berikut



Graf 4. Pohon pencarian dengan start node 1 dan goal node 15

Dapat dilihat dari pencarian pohon diatas, bahwa semua node memiliki cost yang >= movement point dan node tersebut bukan merupakan goal node. Karena costnya sudah lebih besar dari movement point dan belum mencapai tujuan maka pergerakan unit tersebut tidak diperbolehkan.

Contoh lain adalah pergerakan unit yang diperbolehkan yaitu dari node awal 1 menuju ke node akhir 3. Maka akan dibentuk pohon seperti berikut :



Graf 5. Pohon pencarian dengan start node 1 dan goal node 3

Dapat dilihat dari pohon pencarian diatas, pencarian sampai ke node terakhir dengan cost = 3 = movement point, sehingga gerakan tersebut diperbolehkan dan unit dapat bergerak ke tempat tujuan

C. Pros and Cons

Kelebihan menggunakan strategi ini adalah nilai dan jalur yang ditemukan selalu merupakan jalur yang akurat dan paling efisien karena *game* strategi biasanya terdiri dari tile atau unit-unit yang dapat diselesaikan dengan pencarian graf.

Kekurangan dari strategi ini adalah : Pertama, A* tidak terlalu cepat dalam menentukan jalur, hal ini tidak boleh ada dalam game *Real Time Strategy* karena *game* tersebut membutuhkan respon cepat dalam mengambil tindakan.Hal ini dapat diatasi dengan cara mengkombinasikan A* dengan *navigation mesh* sehingga node-node yang berupa unit pada *game* Dota 2 dikelompokan terlebih dahulu sehingga node menjadi sedikit. Kedua, A* merupakan pencarian statik, sehingga dapat dilihat dikedua game bahwa algoritma A* dipanggil setiap kali unit melakukan gerakan karena kondisi map yang berbeda-beda setiap adanya pergerakan. Ketiga, A* memiliki informasi lengkap, terjadi masalah ketika adanya *fog of war* atau area di map yang belum dieksplorasi sehingga tidak tahunya kondisi di bagian tersebut. Dapat diatasi dengan cara menbamhkan bobot yang ada pada area *fog of war* tersebut sehingga unit akan lebih sering melewati area yang sudah tereksplorasi.

IV. KESIMPULAN

Pada pengerjaan makalah ini, penulis mendapatkan beberapa kesimpulan, diantaranya :

- Graf dapat dimanipulasi untuk dapat menyesuaikan dengan keadaan permainan
- Pada kedua game dapat diterapkan algoritma pencarian algoritma A* dengan modifikasi dan jenis graf yang berbeda
- Algoritma permainan A* kurang baik dalam hal performansi untuk kedua *game* yang dibahas karena dipanggil berkali-kali sehingga membutuhkan kemampuan komputasi yang cukup berat

V. REFERENCES

1. http://theory.stanford.edu/~amitp/GameProgramming/ diakses tanggal 19/12/2013
2. http://stackoverflow.com/questions/4059403/how-does-pathfinding-in-rts-video-games-work diakses tanggal 19/12/2013
3. http://www.policyalmanac.org/games/aStarTutorial.htm diakses tanggal 19/12/2013
4. Gambar didapat dari screenshot game

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, 19 Desember 2013

Riefky Amarullah Romadhoni - 13511038