Penggunaan A-Star pada Penentuan Pathing dalam Game

Nicolaus Boby – 13516077
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jl. Ganesha 10 Bandung 40132, Indonesia
nicolausboby@gmail.com

***Abstrak*—Pencarian *path* atau jalur adalah masalah klasik yang sering ditemui dalam berbagai persoalan, bahkan dalam *game* sekalipun. Game *real time strategy* yang memiliki intelijensi buatan sebagai *Non-Player Character* (NPC) sering kali menggunakan algoritma pencarian jalur untuk menggerakan bot-nya. Makalah ini akan menjelaskan pemanfaatan algoritma A-Star untuk pencarian jalur. Bot dan intelijensi buatan yang memilih jalur terpendek pada pergerakannya menjadi lebih menantang bagi para pemain.**

Kata Kunci*—Path-finding, RTS, A-Star, A

I. PENDAHULUAN

Algoritma A* (dibaca A-star) merupakan algoritma yang cukup sering digunakan untuk pencarian jalur terpendek dari sebuah graf atau lebih dikenal dengan istilah *path-finding*. *Path-finding* sering digunakan di dalam permainan komputer untuk menggerakan karakter-karakter dengan AI, terutama permainan-permainan strategi real-time yang memiliki bot sebagai musuhnya atau lebih dikenal dengan player-versus-enemy.

Algoritma A* akan memastikan bahwa gerakan yang dilakukan oleh bot permainan adalah gerakan yang paling efektif dan optimal agar permainan terasa lebih menantang, nyata dan sulit. A* juga memungkinkan bot bergerak dalam grid-map yang penuh dengan halangan. Meskipun seringkali halangan terhadap path ini seringkali muncul dan hilang seiring berjalannya permainan, kalkulasi dan penentuan jalur terpendek dapat terus dilakukan seiring pergerakan bot.

Permainan RTS merupakan salah satu genre yang paling terkenal dan diminati, karena dinamika permainannya yang lebih cepat, dimainkan oleh lebih banyak pemain, dan terasa lebih nyata interaksi antara permainan dan pemainnya.

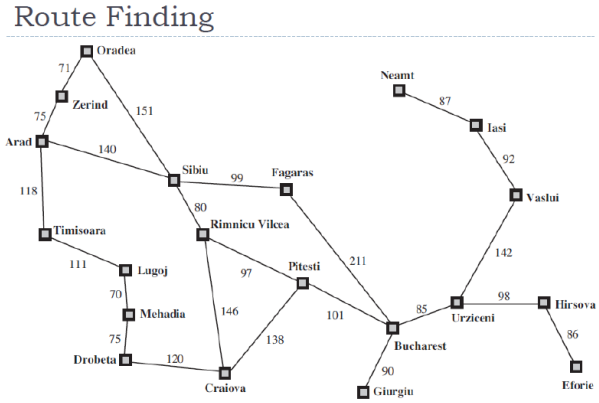
Artifisial intelijen di dalam video game merupakan hal penting. AI digunakan untuk menghasilkan perilaku yang responsif dan adaptif atau cerdas oleh NPC. Ungkapan "game AI" biasanya merujuk kepada algoritma yang juga mengandung teknik *control theory*, robotika, grafika computer dan teknik komputasi, yang mungkin tidak sepenuhnya menggambarkan AI yang sebenarnya. Banyak pakar ahli yang mengatakan bahwa AI dalam game bukanlah tentang intelijensi, sementara AI yang sesungguhnya fokus kepada *machine learning*, pengambilan keputusan dan interaksi.

Fokus dari makalah ini hanyalah AI bot yang dapat mencari jalur terpendek pergerakannya.

II. DASAR TEORI

Proses pencarian merupakan mekanisme pemecahan masalah yang paling umum di dalam kecerdasan buatan. Secara umum, proses pencarian dibagi menjadi 2, yaitu:

1. Uninformed Search
   Proses pencarian ini disebut juga blind search, yaitu pencarian tanpa informasi
2. Informed Search
   Proses pencarian dengan informasi mengenai jalur atau tujuan.



Gambar 1. Contoh Route Finding

A. Algoritma A* Search

Algoritma A* atau dibaca A-Star adalah algoritma yang digunakan untuk pencarian jalur dan penelusuran graf. Pada tahun 1968, peneliti AI Nils Nilsson membuat sebuah algoritma path-finding yang kemudian dinamakan A1, yang merupakan versi lebih ampuh dalam pencarian jalur terpendek dibandingkan dengan algoritma Dijkstra. Kemudian dikembangkan oleh Bertram Raphael dan memunculkan versi selanjutnya yaitu A2. Pada tahun berikutnya, Peter E. Hart mengusulkan perubahan minor, yang kemudian ketiga peneliti diatas mengusulkan bahwa versi revisi A2 merupakan algoritma yang optimal untuk pencarian jalur terpendek pada suatu kondisi yang terdeskripsi dengan cukup jelas.

A* merupakan algoritma pencarian Informed Search, yang berarti memecahkan masalah pencarian dengan mencari jalur yang mungkin menuju solusi atau tujuan dengan *cost* terendah.

Setiap iterasi didalam pengulangannya, algoritma A* akan menentukan cabang yang memiliki *cost* terendah. Penentuan *cost* menggunakan formula:

$$f(n) = g(n) + h(n)$$

g(n) merupakan cost dari node yang telah dilalui oleh jalur yang sedang diperiksa dan h(n) adalah cost dari simpul yang sedang diperiksa ke tujuan. h(n) dalam hal ini sering kali ditentukan oleh pencari, sehingga akan berbeda-beda sesuai dengan kebutuhan pencarian.

Berikut merupakan algoritma A* secara umum

```
// A* Search Algorithm
1.  Initialize the open list
2.  Initialize the closed list
    put the starting node on the open
    list (you can leave its f at zero)

3.  while the open list is not empty
    a) find the node with the least f on
       the open list, call it "q"

    b) pop q off the open list

    c) generate q's 8 successors and set their
       parents to q


   d) for each successor
       i) if successor is the goal, stop search
         successor.g = q.g + distance between
                               successor and q
         successor.h = distance from goal to
         successor (This can be done using many
         ways, we will discuss three heuristics-
         Manhattan, Diagonal and Euclidean
         Heuristics)

         successor.f = successor.g + successor.h

       ii) if a node with the same position as
           successor is in the OPEN list which has a
          lower f than successor, skip this successor

       iii) if a node with the same position as
           successor  is in the CLOSED list which has
           a lower f than successor, skip this successor
           otherwise, add  the node to the open list
    end (for loop)

   e) push q on the closed list
   end (while loop)
```

B. Real-Time Strategy

Real-time strategy atau disingkat RTS, merupakan sub-genre dari video game startegi dimana laju permainan tidak ditentukan oleh *turn* atau giliran. Di dalam RTS, para pemain akan menentukan posisi dan pergerakan unit yang dimiliki dibawah control penuh pemain. Tujuan atau goal dari permainan RTS dapat bervariasi, namun pada umumnya pemain akan diminta untuk mengolah *resource* yang dimiliki untuk menguasai permainan.

Pada awalnya game RTS dimulai oleh pionir game *Utopia* (1989). Game tersebut memiliki elemen hybrid dari aksi giliran yang ditentukan oleh waktu nyata sebenarnya. Kemudian genre ini merambah ke dunia game 3D pada awal tahun 1995, dengan lebih menekankan unsur jumlah unit yang dimiliki, tipe unit, peta permainan yang lebih luas, dan sebagainya.

Gambar 2. Screenshot dari game 0 A.D.

C. Video Game Bot

Di dalam video game, bot adalah tipe dari AI *expert system software* yang memainkan peran pengganti manusia. Bot digunakan dalam berbagai macam genre permainan dengan berbagai macam tugas. Pemakaian bot atau script bot biasanya dilarang pada genre MMORPG (massively multiplayer online role-playing game) tetapi jumlah yang cukup signifikan digunakan dalam genre tesebut diluar permainan player-vs-player ataupun turnamen.

Ada 2 jenis bot dalam video game, yaitu:

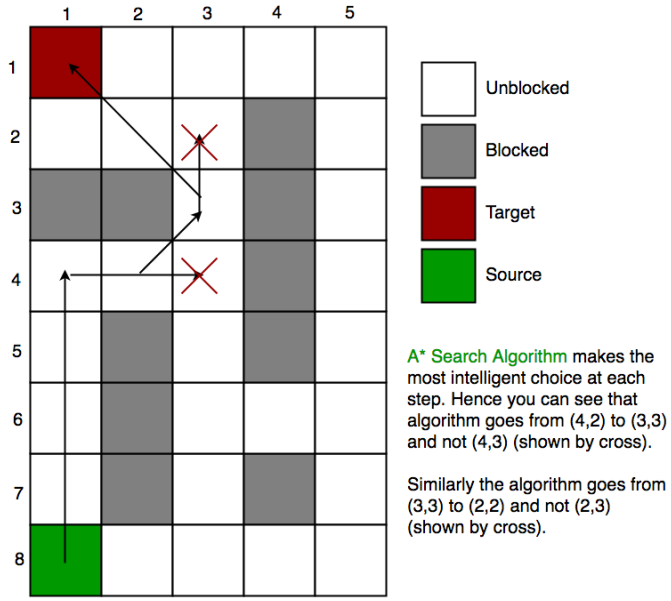
1. *Static bot*
   Bot statik didesain untuk melakukan tindakan-tindakan yang telah dibuat sebelumnya atau sudah memiliki scenario tertentu. Bot ini cenderung memilih tindakan berdasarkan posisi dalam map dan tindakan pemain, atau dapat dikatakan reaktif.
2. *Dynamic bot*
   Bot dinamik beradaptasi sesuai lokasi peta, tindakan pemain, dan perlakuan sekitar. Sehingga bot ini akan belajar untuk memberikan respon yang sesuai.

III. IMPLEMENTASI

Secara umum, path-finding dalam sebuah game bekerja untuk mencari rute atau jalur terdekat untuk mencapai tujuan. Pada makalah ini, akan digunakan sample game dari permainan dota 2.

Dota 2 adalah game multi-player battle online arena (MOBA) yang berasal dari game dota (defense of the ancient) yang merupakan mod dari game Warcraft III. Pengembangan bot dari Dota sebenarnya sudah aja sejak dota 1, yaitu versi permainan dota yang memiliki judul dota AI, yang pengembangannya sendiri dimulai pada tahun 2009 oleh IceFrog.

Secara garis besar, algoritma A* pada bot ini akan bekerja sebagai berikut



Ada 2 fase dalam proses path-finding di dalam algoritma ini, yaitu:

1. Kasar
   Fase proses dengan posisi sel grid memiliki batasan di tengah map, menggunakan algoritma A*
2. Presisi
   Fase ini menggunakan bantuan wall-tracing sebagai batuan heuristicnya. Halangan lebih diperhitungkan di fase ini.

A. Fase Kasar

Seperti yang disebutkan sebelumnya, pada fase ini akan digunakan algoritma A* dengan heuristic jarak Euclidean posisi sel dengan posisi tujuan. Implementasi pada Bahasa C++ nya seperti berikut

```
void PathFinder::findRough(vec2 start, vec2 end, Array<vec2>& path) {
    Point2 s = map()->getCoord(start);
    Point2 e = map()->getCoord(end);
    s.x = clamp(s.x, 1, m_size.width - 2);
    s.y = clamp(s.y, 1, m_size.height - 2);
    m_jpsPlus->find(Coord(s.x, s.y), Coord(e.x, e.y), path);
}
```

1. Point s merupakan posisi mulai
2. Point e merupakan posisi tujuan
3. Fungsi clamp menghitung posisi titik mulai dengan titik tengah
4. Dicari path yang terpendek

B. Fase Presisi

Pada fase ini akan diperhitungkan *obstacle* atau halangan yang berada diantara point mulai dan point tujuan dengan lebih cermat menggunakan bantuan wall-tracing. Wall-tracing akan menyimpan posisi halangan yang berada diantara point mulai dan point tujuan dan menyimpannya. Kemudian akan dibuat seolah-olah jalur diantara halangan dan point yang sedang dilalui tidak dapat dilalui. Lalu algoritma A* akan menentukan jalur terpendek yang dapat dilalui memutari obstacle yang ada tanpa mencoba berulang-ulang "menabrak" obstacle dan memeriksa point yang tidak dapat dilalui. Kode tidak akan ditampilkan disini karena terlalu panjang. Tetapi secara garis besar prosesnya sebagai berikut:

1. Wall tracing mencari *obstacle* diantara point s dan point e
2. Menandai semua point yang kemungkinan "end of path" dari grid map
3. Mencari kemungkinan pojok atau *corner* yang dimiliki obstacle
4. Algoritma A* akan mulai lagi mencari path yang sesuai

Algoritma ini hanya dapat melakukan path-finding pada map berukuran 256x256 dengan efektif.
Wall tracing sendiri merupakan algoritma yang digunakan hanya ketika jarak kedua titik terlalu jauh atau terlalu banyak obstacle. Algoritma ini memiliki kekurangan pada saat obstacle yang berada pada gridmap merupakan maze, karena algoritma ini akan menggunakan sangat banyak memori untuk menyimpan dan mencatat setiap obstacle yang tidak dapat dilalui.

C. Eksperimen

Selanjutnya akan dilakukan eksperimen program dengan menggunakan custom lobby di Dota 2. Perlu diketahui, map Dota 2 yang digunakan disini bukanlah map permainan yang biasa digunakan oleh para pemain. Map yang biasa digunakan berukuran 25.000 x 25.000 pixel, terlalu besar untuk keperluan observasi, dan menjadi tidak optimal jika menggunakan algoritma ini. Map yang digunakan disini berukuran 5.000 x 5.000 pixel, dengan grid yang berukuran 500 x 500.

1. Percobaan 1, tanpa halangan

Posisi mulai	200,200
Posisi tujuan	400,400

Hasil yang didapat:

Posisi	400,400
Total jarak tempuh	282.8427
Waktu kalkulasi	~2.5 ms
Waktu tempuh	5619 ms

2. Percobaan 2, tanpa halangan

Posisi mulai	400,100
Posisi tujuan	500,200

Hasil yang didapat :

Posisi	500,200
Total jarak tempuh	141.4213
Waktu kalkulasi	~2.2 ms
Waktu tempuh	2801 ms

3. Percobaan 3, menggunakan halangan

Posisi mulai	350,90
Posisi tujuan	500,200

Hasil yang didapat:

Posisi	500,200
Total jarak tempuh	201.0108
Waktu Kalkulasi	~19.8 ms
Waktu tempuh	3403 ms

4. Percobaan 4, menggunakan halangan

Posisi mulai	200,200
Posisi tujuan	400,400

Hasil yang didapat:

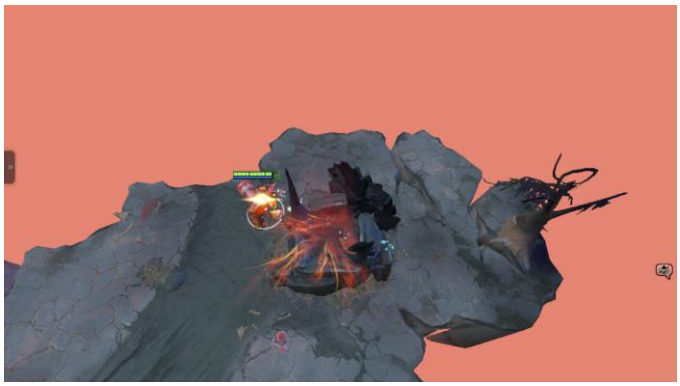
Posisi	400,400
Total jarak tempuh	391.7813
Waktu kalkulasi	~78.7 ms
Waktu tempuh	7291 ms

5. Percobaan 5, jarak terjauh tanpa halangan

Posisi mulai	20,35
Posisi tujuan	480, 465

Hasil yang didapat :

Posisi	480,465
Total jarak tempuh	629.6825
Waktu kalkulasi	~101 ms
Waktu tempuh	15833 ms



6. Percobaan 6, jarak terjauh menggunakan halangan

Posisi mulai	20,35
Posisi tujuan	480, 465

Hasil yang didapat:

Posisi	245,231
Total jarak tempuh	311.6723
Waktu kalkulasi	>150 ms
Waktu tempuh	7858 ms

Pada Percobaan 6, tidak diperlihatkan gambar karna waktu kalkulasi terlalu lama, percobaan dihentikan ditengah saat unit berusaha menghindari halangan.

IV. ANALISIS

Pada percobaan 1 dan percobaan 2, digunakan jarak yang relatif dekat. Terlihat pada percobaan 1, waktu kalkulasi yang didapat cukup rendah.dan jarak yang ditempuh yaitu jarak terdekat (jarak Euclidean) titik asal ke titik tujuan.

Pada percobaan kedua jarak yang digunakan lebih jauh daripada percobaan pertama, hasil yang didapat tetap optimal, walaupun waktu yang dibutuhkan lebih dari 2 kali lipat percobaan pertama.

Pada percobaan ketiga dan keempat, terdapat *obstacle* atau halangan diantara titik mulai dan titik akhir. Algoritma mampu menemukan jalur terpendek yang dapat dilalui oleh unit. Akan

tetapi, waktu penghitungan algoritma melonjak sangat tinggi, karena pada percobaan menggunakan halangan, algoritma akan masuk ke fase wall tracing, sehingga membutuhkan waktu yang relative jauh lebih lama dibandingkan tanpa halangan.

Pada percobaan kelima dan keenam, digunakan jarak yang sangat jauh, jarak maksimum yang terdapat di dalam map, yaitu dari ujung awal map sampai ujung akhir map. Terlihat pada percobaan kelima, walaupun jarak sangat jauh dan melebihi kemampuan optimal algoritma ini (lebih dari 256x256), algoritma tetap dapat menghasilkan jalur yang minimum. Akan tetapi pada percobaan keenam, algoritma menjadi sangat tidak efektif dan bahkan berhenti berjalannya program saat dieksekusi. Hal ini dikarenakan adanya halangan diantara titik awal dan titik akhir dan jarak antara kedua titik tersebut sangat jauh dan diluar batas.

V. KESIMPULAN

Algoritma A* dapat digunakan untuk penentuan jalur terpendek di dalam game. Algoritma A* yang dibantu dengan wall tracing akan menjadi sangat tidak efektif ketika digunakan pada map yang berukuran sangat besar (lebih dari 256 x 256). Sehingga walaupun algoritma A* ini dapat menghasilkan hasil yang optimum, tetapi tidak cocok digunakan pada game-game rilisan baru yang biasanya memiliki ukuran map yang sangat besar.

REFERENCES

[1] Munir, Rinaldi. Diktat Strategi Algoritma.Informatika, Penerbit Informatika :Bandung, 2006.

[2] https://developer.valvesoftware.com/wiki/Dota_Bot_Scripting. Diakses 11 Mei 2018

[3] https://github.com/guinzoo/rts-path-finding/. Diakses 11 Mei 2018

[4] https://www.gamasutra.com/blogs/BenWeber/20161106/284970/A_Brief_History_of_RTS_AI_Research.php. Diakses 12 Mei 2018

[5] Dechter, Rina; Judea Pearl (1985). "Generalized best-first search strategies and the optimality of A*"

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, 13 Mei 2018

Nicolaus Boby - 13516077