Penggunaan Algoritma Pathfinding pada Game

Ahmad Fauzan (13510004)

Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jl. Ganesha 10 Bandung 40132, Indonesia
13410004@std.stei.itb.ac.id

Game adalah sebuah permainan digital yang dapat dimainkan di berbagai perangkat yang mendukung I/O. Game masa kini lebih kaya, dapat juga merepresentasikan dunia nyata. Game yang akan dibahas merupakan game yang hampir mirip dunia nyata. Game ini memiliki peta.Karakter dikontrol dengan tetikus untuk diarahkan ke sebuah titik pada peta.Bagaimana sebuah komputer dapat mengkomputasi langkah-langkah apa saja yang akan dilakukan sang karakter untuk mencapai titik tersebut?.Makalah ini akan membahas cara-cara yang dilakukan game untuk menggerakkan karakter dari satu titik ke titik lain pada peta.

Indeks: **Game, Pathfinding, Map, DFS, BFS, Branch and Bound, A***

I. PENDAHULUAN

Game *(Video Game*) merupakan sebuah program/aplikasi yang berinteraksi dengan manusia layaknya sebuah program/aplikasi biasa. Perbedaannya adalah game lebih mengandung unsur kesenangan kepada pemainnya. Dampak dari game tidak langsung dapat dirasakan. Namun, dampaknya bisa saja berupa kepuasan pemain.

Game berkembang karena adanya mesin (*hardware*) yang dapat digunakan untuk mengimplementasi sebuah game. Pada awalnya, game hanya memiliki sumber daya yang cukup sedikit dan hardware yang tidaklah murah. Game generasi pertama antara lain : Pong, spacewar dan beberapa game kasual lainnya.

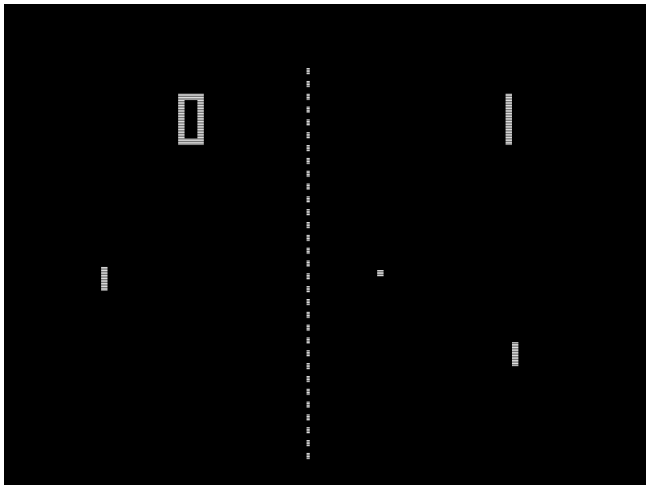


Fig 1. Pong

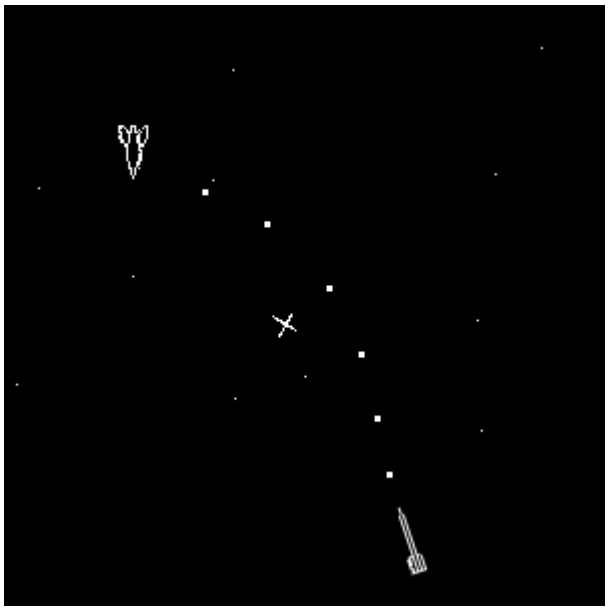


Fig 2. Spacewar

Pada perkambangannya, performansi dari hardware dikembangkan. Dari yang awal game hanya 2 dimensi menjadi 3 dimensi. Selain itu, game sekarang ini kaya akan sumber daya, bahkan hampir dapat menggambarkan dunia nyata seperti peta daerah.

Fig 3. Age Of Empire - Salah satu game yang menggunakan sistem map

Peta dalam game menggambarkan letak objek, jalur dan tanda-tanda lainnya. Dalam pengembangannya game yang

memuat peta dan memainkannya dengan cara mengklik daerah tertentu sehingga karakter bergerak ke daerah tersebut. Dalam pemrosesan gerakan karakter ini, kita membutuhkan sebuah algoritma *pathfinding*.

II. PATHFINDING

Pathfinding merupakan cara untuk mendapatkan *route* antara 2 buah point. Pathfinding memiliki beberapa algoritma yang bisa diterapkan antara lain.

A. Brute Force

Algoritma ini merupakan algoritma yang paling mudah dimengerti. Cara kerjanya adalah membandingkan posisi sekarang dengan posisi tujuan dan menentukan langkah berikutnya.

B. BFS

Breadth-First Search merupakan algortima yang menyelesaikan masalah dengan memanfaatkan struktur pohon.

C. DFS

Deep-First Search merupakan algoritma yang menyeleseaikan masalah dengan memanfaatkan struktur pohon. DFS mencari solusi ke node yang paling dalam pada pohon.

D. Branch and Bound dan A*

Branch and Bound merupakan pengembangan dari BFS. Pada Branch and Bound, setiap node memiliki harga (dengan cara penghitungan harga yang bermacam-macam). Harga node menentukan kedekatan node dengan solusi.

Algoritma A* (Baca : A bintang) merupakan salah satu pengembangan dari Algoritma *Branch and Bound*. Perhitungan harga pada algoritma A* memanfaatkan unsur heuristik pada benda.

III. PENGGUNAAN ALGORITMA

Penerapan algoritma pada peta game.

A. Brute Force

Penyelesaian pada algoritma brute force terbilang cukup mudah dipahami. Perbandingan posisi dan menentukan langkah selanjutnya.

Algoritma :

1. Cek apakah tujuan ada disebelah kanan atau tidak,
2. Jika ya, bergerak ke kanan,
3. Jika tidak, bergerak ke kiri,
4. Cek apakah tujuan ada disebelah atas atau tidak,
5. jika ya, bergerak ke atas,
6. Jika tidak bergerak ke bawah.

Contoh penyelesaian dengan algoritma ini.

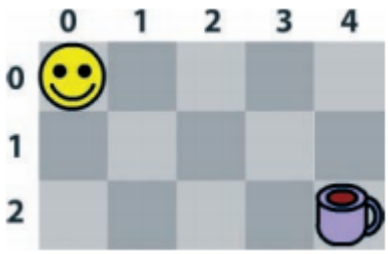


Fig 4. Contoh persoalan

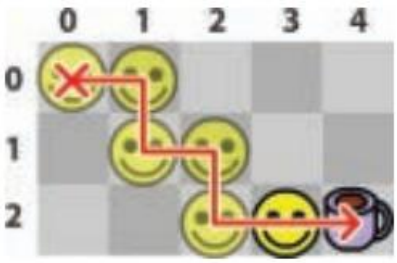


Fig 5. Penyelesaian dengan brute-force

Namun, tidak semua permasalahan pathfinding ini dapat diselesaikan dengan algortima brute force. Permasalahan itu muncul ketika adanya penghalang.

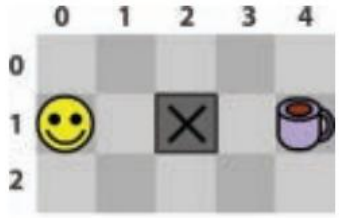


Fig 6. Permasalahan dengan penghalang

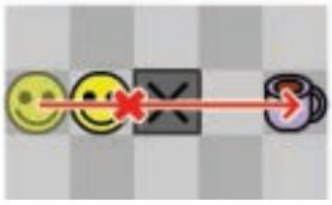


Fig 7. Algoritma yang dijelaskan gagal

Modifikasi dari algoritma ini diperlukan untuk menyelesaikan kegagalan di atas. Dalam pengembangannya, algoritma ini harus diperbaiki untuk kasus-kasus tertentu.

B. BFS

Breadth-First Search merupakan algortima yang menyelesaikan masalah dengan memanfaatkan struktur pohon. Algoritma ini cukup intuitif di kalangan developer.

Konsep algoritma ini pada pathfinding adalah menjelajahi 4 mata angin dari posisi sekarang yang belum pernah dikunjungi. Dalam meprosesannya, algoritma ini menggunakan Queue untuk memproses point sesuai urutan.

```
Buat Queue kosong
Tambahkan posisi awal pada Queue
While Queue belum kosong
        PULL Queue
                If Node adalah tujuan
                        Solusi ditemukan (keluar!)
                Else
                For each Node-Node tetangga
                        If Node tetangga belum dilewati
                                PUSH Node ke Queue
```

Pseudo-code 1. BFS pada pathfinding

BFS membentuk semua struktur pohon semu. Struktur pohon ini digunakan untuk menjelaskan bagaimana BFS bekerja. BFS akan mengeksekusi semua node pada 1 level, setelah itu pada level berikutnya.

Cara dan Conoth BFS bekerja:

Fig 8. contoh permasalahan

Proses yang bekerja pad BFS untuk permasalahan di atas, digambarkan dengan pohon.

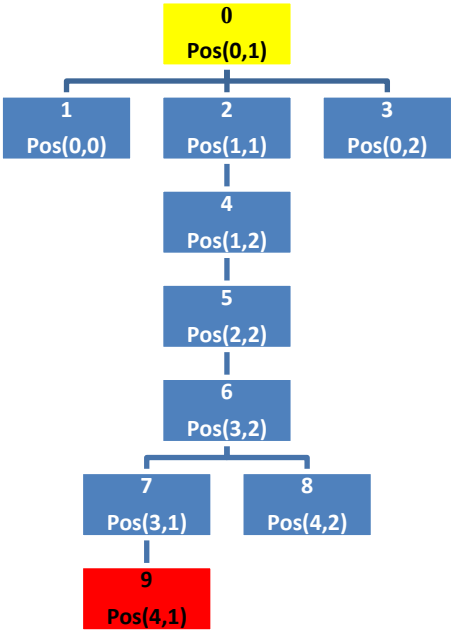


Fig 9. Pohon hasil BFS

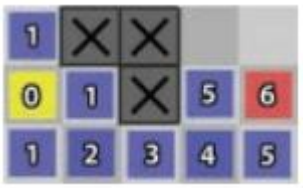


Fig 10. Hasil solusi BFS

C. *DFS*

Berbeda dengan BFS, DFS memproses Node lebih dalam sampai solusi ditemukan atau tidak ditemukan.

```
Function DFS(Node)

if Node adalah tujuan
        Hasil ditemukan
else
        For each Node-Node tetangga
                If Node tetangga belum dilewati
                        DFS(Node-tetangga)
```

Pseudo-code 2. DFS pada pathfinding

Untuk permasalahan yang sama dengan BFS, hasil pohon DFS adalah :

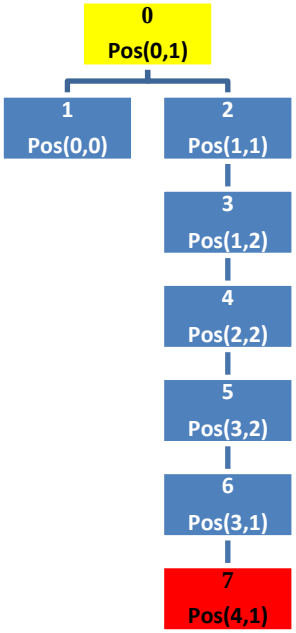


Fig 11. Pohon hasil DFS

D. *Branch and Bound dan A**

Brach and Bound merupakan pengembangan dari BFS. Penambahan harga dari tiap node. Node dengan harga terkecil atau yang terdekat dengan solusi merupakan node yang akan diproses terlebih dahulu.

```
Buat PriorityQueue kosong
Tambahkan posisi awal pada PriorityQueue
While PriorityQueue belum kosong
PULL PriorityQueue
        If Node adalah tujuan
                        Solusi ditemukan (keluar!)
        Else
        For each Node-Node tetangga
                If Node tetangga belum dilewati
                        Hitung Cost Node
                        PUSH Node ke PriorityQueue
```

Pseudo-code 3. Branch and Bounds

Branch and Bounds tidak baik digunakan pada peta biasa, sebaiknya menggunakan BFS saja. B&B lebih baik digunakan pada map yang setiap poin nya mempunyai nilai (Contoh: skor, ketinggian, dll).

A* (Baca: A*) merupakan salah satu B&B. A* banyak digunakan untuk pathfinding. Fungsi untuk menghitung cost setiap node adalah :

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

g(x) merupakan jarak yang telah dilalui sebelumnya (dari posisi awal ke posisi sekarang). h(x) adalah sebuah fungsi heuristik.

Fungsi heuristik yang paling mudah untuk digunakan adalah jarak dari posisi yang di proses ke proses tujuan (jarak dapat dihitung dengan garis lurus dari a ke b).

$$f(x) = g(x) + distance(pos, goal)$$

```
Hitung Heuristic(start,goal)
Buat PriorityQueue kosong
Tambahkan posisi awal pada PriorityQueue
While PriorityQueue belum kosong
PULL PriorityQueue
        If Node adalah tujuan
                        Solusi ditemukan (keluar!)
        Else
        For each Node-Node tetangga
                If Node tetangga belum dilewati
                        F(NodeT) = level + distance(NodeT,goal)
                        PUSH Node ke PriorityQueue
```

Fig 12. A* sederhana

Contoh penyelesaian pathfinding untuk fig. 8.

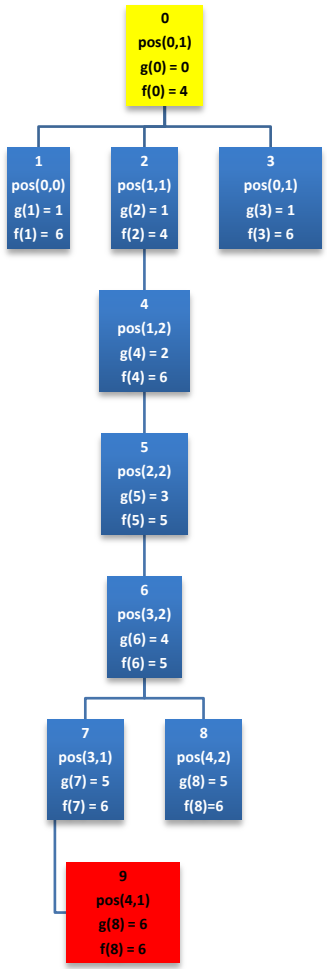


Fig 13. Aksi A*

Terlihat, solusi A* dan BFS terhilat sama. Namun, untuk lingkup yang lebih luas lagi A* dapat berjalan lebih optimal dibanding BFS.

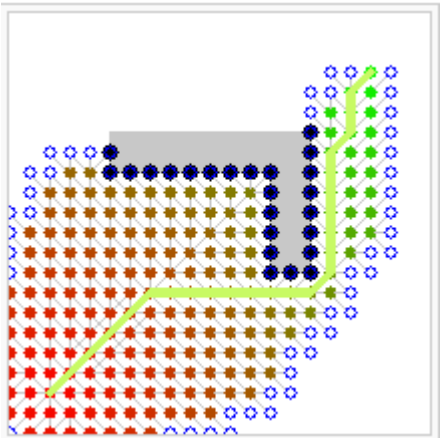


Fig 14. Solusi dengan A*[3]

Fig. 14. dapat dilihat bahwa proses A* lebih baik dari BFS. Jika BFS, maka jumlah node yang ditelusuri akan lebih banyak (bagian sebelah kiri tujuan akan ditelusuri juga).

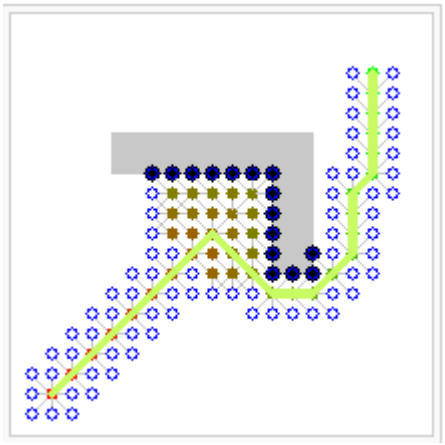


Fig 15. A* dengan perhitungan Heuristik yang lain[3]

Dengan memodifikasi perhitungan heuristik, kita dapat memperoleh A* yang lebih baik dan cepat. (fig. 15).

IV. KESIMPULAN

Algoritma brute force harus dimodifikasi untuk setiap kasus yang unik. Algoritma DFS, BFS dan B&B dapat digunakan untuk pathfinding pada game.

Algoritma yang paling ampuh untuk persoalan pathfinding adalah algoritma A*. Jika dibandingkan dengan BFS dan DFS, Algoritma A* lebih sedikit memproses node.

REFERENSI

[1] Munir, Rinaldi. *Diktat Kuliah IF3051 Strategi Algoritma*. Program Studi Teknik Informatika. 2009.

[2] Game Career Guide. 2011."*Getting there from here*". p.13-16

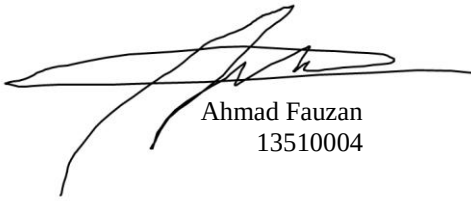
[3] http://en.wikipedia.org/wiki/A*_search_algorithm tanggal akses : 19 Desember 2012.

[4] http://theory.stanford.edu/~amitp/GameProgramming/Heuristics.html tanggal akses : 21 Desember 2012.

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, 21 Desember 2012



Ahmad Fauzan
13510004