Penerapan Teori Graf dalam Game Bertipe *Real Time Strategy* (RTS)

Yudha Okky Pratama/13509005
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jl. Ganesha 10 Bandung 40132, Indonesia
13509005@std.stei.itb.ac.id

***Abstrak*—Makalah Struktur Diskrit ini akan membahas mengenai penerapan beberapa teori graf yang diterapkan dalam pengembangan game* Real Time Strategy (RTS)*, pada makalah ini akan banyak diambil contoh dari game Age of Empire II. Beberapa teori graf yang akan dibahas disini antara lain teori* dependency graph *untuk game* RTS *yang pada umumnya bertipe* resource management*, teori* pathfinding *dalam system navigasi* agent*, dan* state graph *untuk pencarian pemecahan masalah*.**

Kata Kunci : Real Time Strategy, *kecerdasan buatan, graf,* dependency, pathfinding, state.

I. PENDAHULUAN

A. Pengertian Game Bertipe Real Time Strategy

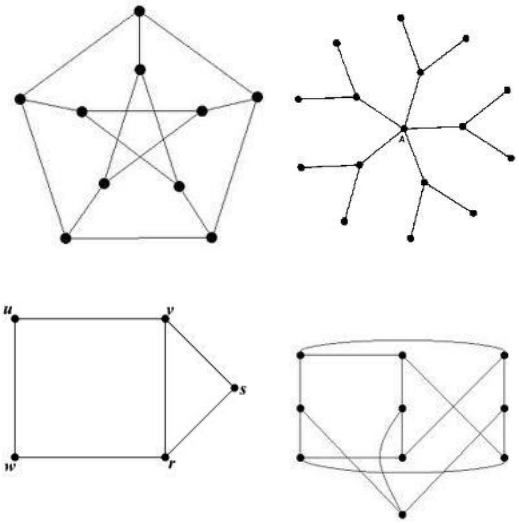
Real Time Strategy adalah istilah bahasa Inggris yang mengacu kepada salah satu genre dalam permainan komputer yang memiliki ciri khas berupa permainan perang yang terdiri atas pembangunan sebuah kekuatan atau negara dengan mengumpulkan sumberdaya dan pengaturan pasukan perang. Jenis permainan ini melibatkan pengaturan perang tingkat strategis contohnya pasukan, peperangan, dan diplomasi. Aspek yang paling dominan dalam RTS tentu saja adalah peperangan, tetapi selain itu RTS juga melibatkan beberapa aspek lain seperti pengelolaan sumberdaya dan pembangunan.

Dalam RTS, tidak dikenal system *turn* atau giliran bermain, game berjalan dalam waktu yang bersamaan sehingga ketika seorang pemain membuat keputusan, secara bersamaan pihak lawan juga berkasi sehingga menimbulkan serangkaian kejadian dalam waktu yang sebenarnya. Tema permainan dapat berupa sejarah (misalnya seri *Age of Empire*), fantasi (misalnya Warcraft), dan fiksi ilmiah (misalnya Sins of a Solar Empire).

B. Pengenalan Graf

Graf G didefinisikan sebagai pasangan himpunan (V,E), ditulis dengan notasi G = (V,E), yang dalam hal ini V adalah himpunan tidak kosong dari simpul-simpul dan E adalah himpunan sisi yang menghubungkan sepasang simpul.

Secara geometri, graf digambarkan sebagai sekumpulan noktah (simpul) di dalam bidang dwimatra yang dihubungkan dengan sekumpulan garis (sisi).



Gambar 1 : contoh-contoh graf

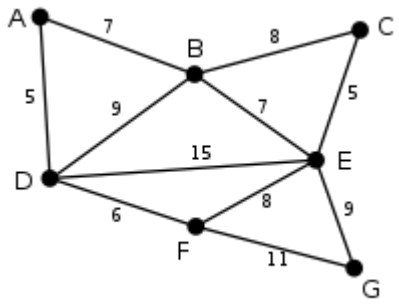
Terminologi Dasar

Beberapa terminologi dasar yang sering dipakai

1. Bertetangga
   Dua buah simpul pada graf tak berarah G dikatakan bertetangga bila keduanya terhubung langsung dengan sebuah sisi.
2. Bersisian
   Untuk sembarang sisi e = (u,v), sisi e dikatakan bersisian dengan simpul u dan simpul v
3. Simpul terpencil
   Simpul terpencil ialah simpul yang tidak mempunyai sisi yang bersisian dengannya. Atau, dapat juga simpul terpencil adalah simpul yang tidak satupun bertetangga dengan simpul-simpul lainnya.
4. Graf kosong
   Graf kosong adalah graf yang himpunan sisinya merupakan himpunan kosong.
5. Derajat
   Derajat suatu simpul pada graf tak berarah

adalah jumlah sisi yang bersisian dengan simpul tersebut.

6. Siklus atau sirkuit
   Siklus atau sirkuit adalah lintasan yang berawal dan berakhir pada simpul yang sama.
7. Terhubung
   Graf tak berarah G disebut graf terhubung jika untuk setiap pasang simpul u dan v di dalam himpunan V terdapat lintasan dari u ke v.
8. Upagraf
   Misalkan G = (V,E) adalah sebuah graf. G1 = (V1,E1) adalh upagraf dari G jika V1 merupakan bagian dari V dan E1 merupakan bagian dari E.
9. Graf berbobot
   Graf berbobot adalah graf yang setiap sisinya diberi sebuah harga (bobot).



Gambar 2 : graf berbobot

Lintasan Euler adalah lintasan yang melalui masing-masing sisi di dalam graf tepat satu kali. Bila lintasan tersebut kembali ke simpul asal, membentuk sebuah lintasan tertutup (sirkuit), maka lintasan tertutup itu disebut **sirkuit Euler**. Jadi, sirkuit Euler adalah sirkuit yang melewati tiap sisi masing-masing tepat satu kali.

Lintasan Hamilton adalah lintasan yang melalui tiap simpul di dalam graf tepat satu kali. Jika lintasan itu tepat kembali ke titik awal dan membentuk sirkuit, maka litasan tertutup ini disebut **sirkuit Hamilton**. Jadi, sirkuit Hamilton adalah sirkuit yang melalui tiap simpul dalam graf tepat satu kali, kecuali simpul asal yang dilalui sebanyak dua kali.

Secara umum representasi berupa graf banyak dimanfaatkan dalam dalam aplikasi game atau program-program yang lain.

Pohon

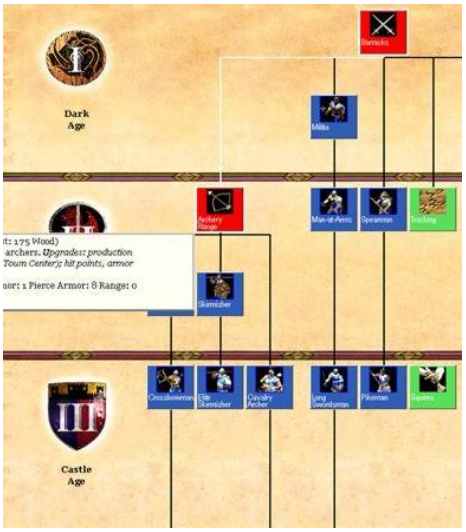
Pohon adalah graf tak berarah terhubung yang tidak mengandung sirkuit.

Pohon yang sebuah simpulnya diperlakukan sebagai akar da sisi-sisinya diberi arah menjauh dari akar dinamakan **pohon berakar** (*rooted tree).*

Pohon keputusan digunakan untuk memodelkan persoalan yang terdiri dari serangkaian keputusan yang mengarah ke solusi. Tiap simpul dalam menyatakan keputusan, sedangkan daun menyatakan solusi.

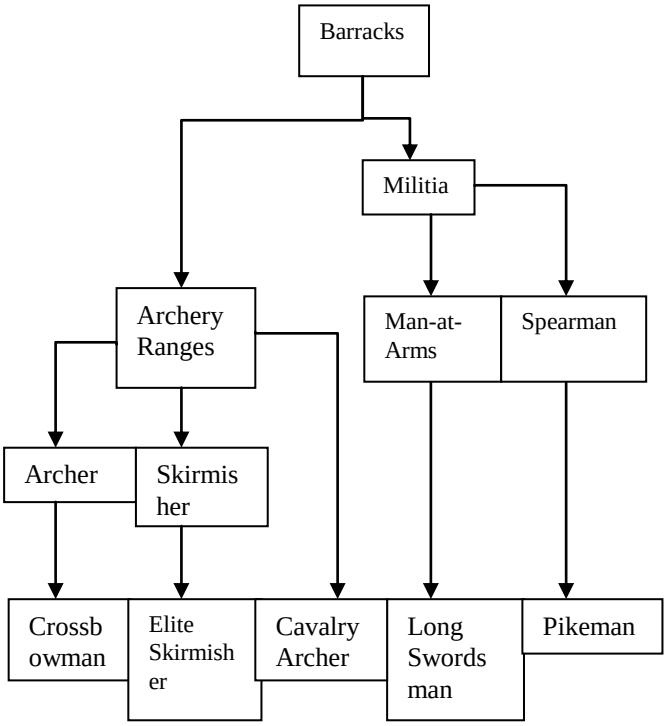
II. DEPENDENCY GRAPH

Dependency graph banyak ditemui pada game-game ber-*genre* RTS, karena game-game dengan jenis tersebut berbasis *resource management* dalam membangun unit-unit yang terlibat didalamnya, dapat berupa bangunan, berbagai jenis pasukan perang, *research*, dll. Misalnya dalam game Age of Empire II, menggunakan *dependency graph* untuk menggambarkan ketergantungan antar unit dalam game tersebut seperti yang ditunjukkan dalam gambar berikut.



Gambar 3 : Technology tree dalam game Age of Empire

Technology tree tersebut dapat digambarkan sebagai *dependency graph* sebagai berikut.



Gambar 4 : Contoh dependency graph pada game Age of Empire II

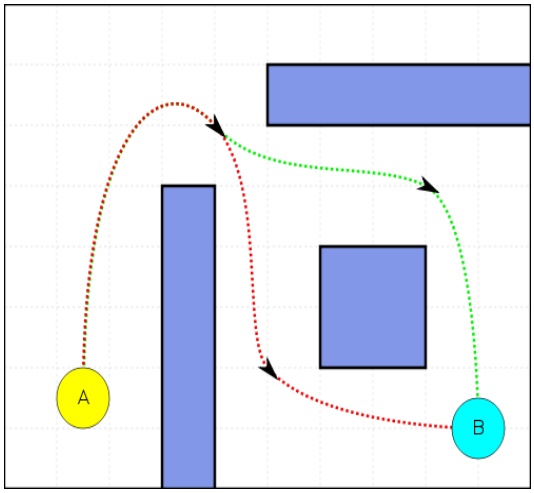
Gambar di atas menunjukkan bahwa jika kita ingin membuat unit dengan jenis Cavalry Archer, maka terlebih dahulu kita harus terlebih dahulu menyelesaikan *research* Castle Age dan membangun Archery Range. Contoh lain, misalnya kita ingin membuat unit Militia, maka kita cukup mencapai Dark Age dan membangun Barrack. Tampak bahwa ada kebergantungan dari suatu unit pada unit yang lain.

Konsep *dependency graph* ini sangat menentukan strategi dalam bermain terutama untuk memilih jalan yang paling cepat dan efektif untuk membangun unit yang diinginkan. Selain itu, dengan graf tersebut seorang pemain dapat memprediksi keadaan lawan, contohnya jika lawan tampak menyerang dengan unit Crossbowmen, dapat dirunut dan diambil kesimpulan bahwa ia sudah membangun Archery Range dan sudah mencapai Dark Age.

III. TEORI PATHFINDING

Pathfinding atau disebut juga *navigation graph* adalah sebuah graf yang merepresentasikan lingkungan virtual dalam game dimana *agent* dapat bergerak atau berpindah tempat. Bentuk graf yang digunakan adalah graf berbobot (weighted graph). Setiap simpul dalam graf merepresentasikan posisi atau area dan sisi menggambarkan jalur yang menghubungkan simpul-simpul. Biasanya sisi dalam *navigation graph* memiliki bobot atau jarak antara masing-masing simpul yang dihubungkan oleh sisi tersebut.

Navigation graph yang sudah dibuat oleh pembuat game nantinya akan menentukan lintasan mana saja yang dapat dilewati dan tidak dapat dilewati oleh *agent*. Misalnya di sebuah simpul tertentu terdapat objek yang menghalangi, maka *agent* tentu tidak dapat melewati lintasan yang melalui simpul tersebut.



Gambar 5 : Ilustrasi *Pathfinding*

Persoalan berikutnya dalam *pathfinding* adalah bagaimana *agent* dapat menentukan jalur mana yang merupakan lintasan terpendek dari simpul awal ke simpul yang dituju.

Contoh kasus menemukan lintasan terpendek dalam game Age of Empire II, misalnya kita memilih agent berupa unit Cavalry (yang diberi lingkaran berwarna kuning)

Gambar 6 : Contoh pathfinding dalam game Age of Empire II

Ketika kita meng-klik kanan pada suatu tempat, yang

sebenarnya merupakan simpul dalam graf *navigation graph*, maka secara otomatis *agent* yang sedang dijalankan akan memilih jalur terpendek ke tempat yang dituju (seperti ditunjukkan dengan garis merah). Algoritma yang popular dan dianggap mangkus untuk menemukan solusi dari pencarian jalur terpendek adalah Algoritma Djikstra.

Contoh Algoritma *Pathfinding*

Berikut adalah salah satu contoh algoritma mencari lintasan terpendek menggunakan representasi koordinat.

Kita memiliki sebuah peta dengan sebuah koordinat START(S) dan koordinat FINISH(F), X merupakan tembok (tidak dapat dilewati), dan _ merupakan ruang yang dapat dilewati.

```
  8 7 6 5 4 3 2 1
x x x x x x x x x x
x _ _ _ x x _ x _ x  1
x _ x _ _ x _ _ _ x  2
x S x x _ _ _ x _ x  3
x _ x _ _ x _ _ _ x  4
x _ _ _ x x _ x _ x  5
x _ x _ _ x _ x _ x  6
x _ x x _ _ _ x _ x  7
x _ _ F _ x _ _ _ x  8
x x x x x x x x x x  9
```

Pertama kita buat sebuah list yang terdiri dari koordinat koordinat yang akan mungkin dilalui. Kita awali dari koordinat START yaitu ((3,8,0)). Kemudian dilanjutkan dengan langkah-langkah sebagai berikut :

1. Buat list dari empat sel yang merupakan sel atau simpul tetangga dari sel sebelumnya, kemudian variabel counter (anggota koordinat yang paling belakang) ditambahkan dengan +1.
   Untuk contoh diatas sel yang didapatkan adalah ((2,8,1),(3,7,1),(4,8,1),(3,9,1))
2. Cek semua sel, kemudian lakukan langkah berikut :
   - Jika sel adalah tembok, hapus dari list
   - Jika dalam list utama terdapat sel yang sama dengan nilai counter yang berbeda, hapus dari list

3. Tambahkan sel yang tersisa ke list utama
4. Maju ke sel selanjutnya di dalam list

Jika disimulasikan akan memberikan hasil :

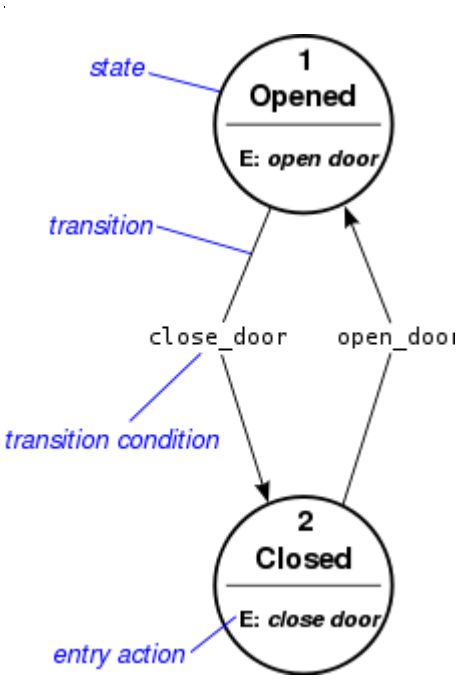
- Langkah pertama, list yang terbentuk : ((3,8,0),(2,8,1),(4,8,1))
- Langkah ke-2 : ((3,8,0),(2,8,1),(4,8,1),(1,8,2),(4,7,2))
- Langkah ke-3 : (...(1,7,3),(4,6,3),(5,7,3))
- Langkah ke-4 : (...(1,6,4),(3,6,4),(6,7,4))
- Langkah ke-5 : (...(1,5,5),(3,5,5),(6,6,5),(6,8,5))
- Langkah ke-6 : (...(1,4,6),(2,5,6),(3,4,6),(6,5,6),(7,8,6))
- Setelah langkah ke-7 : didapatkan solusi ((1,3,7))

```
  8 7 6 5 4 3 2 1
x x x x x x x x x x
x _ _ _ x x _ x _ x  1
x _ x _ _ x _ _ _ x  2
x S x x _ _ _ x _ x  3
x 6 x 6 _ x _ _ _ x  4
x 5 6 5 x x 6 x _ x  5
x 4 x 4 3 x 5 x _ x  6
x 3 x x 2 3 4 x _ x  7
x 2 1 0 1 x 5 6 _ x  8
x x x x x x x x x x  9
```

Ketika ditelusuri, didapatkan solusi berupa lintasan (1,3,7) -> (1,4,6) -> (1,5,5) -> (1,6,4) -> (1,7,3) -> (1,8,2) -> (2,8,1) -> (3,8,0).

III. STATE GRAPH

State graph atau disebt juga sebagai *state diagram* merupakan graf atau diagram yang menggambarkan perilaku mesin, program, atau sistem. Sebuah *state graph* inyatakan dengan *state* atau kondisi yang dapat dicapai oleh sistem beserta transisi dan kondisi transisinya.



Gambar 7 : contoh *state graph*

Dengan melihat graf pada gambar di atas, kita dapat mengidentifikasi perilaku dari sistem tersebut. Pada sistem seperti yang ditunjukkan gambar di atas, terdapat dua *state* yaitu keadaan *closed* dan *opened*, dengan dua kondisi transisi yaitu *close door* dan *open door*. Jika *state* awal sistem adalah *closed*, jika dilakukan aksi berupa *open door*, maka sistem akan berubah menjadi *state opened*, kemudian dari state tersbut jika dilakukan aksi berupa close door, maka sistem akan kembali ke *state closed*. Contoh di atas merupakan contoh yang sangat sederhana, pada implementasinya banyak variasi-variasi yang bisa dibuat dengan *state diagram* atau *state graph* ini.

V. KESIMPULAN

Konsep dan teori graf dapat digunakan sebagai solusi untuk merepresentasikan kasus-kasus tertentu yang dibutuhkan dalam membangun sebuah game atau program.

Game RTS yang berbasis *resource management* menggunakan *dependency graph* untuk merepresentasikan kebergantungan dari unit-unit dalam game tersebut.

Teori Pathfinding dapat diterapkan dalam program untuk kemangkusan dalam mencari lintasan terpendek dari suatu kasus.

Untuk menggambarkan *state* apa saja yang dapat dicapai oleh suatu program, sistem, atau mesin dapat digunakan *state graph* yang kemudian dapat dianalisis lebih lanjut.

Teori graf dapat digunakan untuk dalam berbagai bidang untuk menggambarkan kasus-kasus yang terjadi atau menemukan solusi dari kasus-kasus tersebut.

DAFTAR REFERENSI

[1] G Munir, Rinaldi.2009. "Matematika Diskrit", Edisi Ketiga, Bandung : Penerbit Informatika.

[2] http://en.wikipedia.org/wiki/Pathfinding .waktu akses : 14 Desember 2010 14.00

[3] http://www.freshpatents.com/-dt20090611ptan20090150790.php waktu akses : 14 Desember 2010 15:30

[4] http://id.wikipedia.org/wiki/Real-time_strategy waktu akses : 14 Desember 2010 15:00

[5] http://en.wikipedia.org/wiki/State_diagram waktu akses : 15 Desember 2010 19:00

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

ttd

Yudha Okky Pratama 13509005