Penerapan Algoritma A* Pada *Pathfinding AI* Dalam *Turn-Based Strategy Game*

Trevin Matthew Robertsen – 13515027
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jl. Ganesha 10 Bandung 40132, Indonesia
13515027@std.stei.itb.ac.id

***Abstract*—Algoritma A* adalah salah satu algoritma yang penerapannya sangat banyak pada berbagai masalah, termasuk dalam pembuatan game. Pada proses pembuatan game salah satu faktor paling penting yang perlu diperhatikan adalah kemampuan AI dalam proses *pathfinding*. Dalam makalah ini akan dibahas penerapan algoritma A* dalam *Turn-Based Strategy Game,* khususnya dalam pergerakan AI. Dengan algoritma A* maka pergerakan dan kemampuan *pathfinding* dari AI akan meningkat dan membuat AI dari game semakin "cerdas" dan responsif**

Keywords—artificial intelligence, turn-based, simpul, sisi, cost

I. PENDAHULUAN

Dengan semakin berkembangnya teknologi dan pemahaman akan komputasi, maka semakin banyak pula masalah yang dapat diselesaikan dengan komputasi algoritma dan masalah yang solusinya dapat dirancang lebih optimal dan lebih baik. Permasalahan ini dapat beragam mulai dari melakukan pencarian (search) sampai dengan melakukan penelusuran (pathfinding). Dengan semakin banyaknya masalah ini muncul pula berbagai macam cara untuk menyelesaikan suatu masalah tersebut. Misalnya pada masalah perncarian metode yang dapat dilakukan adalah dengan BFS, DFS, IDS, dll. Pada masalah string matching yang dapat dilakukan adalah dengan KMP, Boyer-Moore, Regex, dll.

Banyak solusi pun dapat ditemukan pada penyelesaian masalah penelusuran (pathfinding) dimana solusinya meliputi A* , Djikstra, D*, dll. Semua solusi ini dapat dipakai untuk melakukan penyelesaian masalah pathfinding. Masalah pathfinding ini meliputi pada pencarian runtutan solusi optimal, pencarian jalur perjalanan optimal, dll.

Masalah pathfinding ini pun akan dialami dalam proses pembuatan game. Pada pembuatan game salah satu komponen paling penting adalah kemampuan dari AI untuk menemukan pemain(player), dan masalah ini akan memakai prinsip pathfinding. AI dari game akan berusaha untuk menemukan jalaur yang sesuai untuk mencapai ke posisi dari pemain untuk "menyerang" pemain tersebut. Pergerakan dan kemampuan penentuan jalur dari AI game ini lah yang akan menjadi acuan apakah AI game tersebut "pintar" dan responsif serta mampu memberikan tantangan bagi pemain ketika bermain dalam game tersebut.

Konsep ini sangatlah penting untuk game yang bertema/genre *turn-based stratergy* dimana pergerakan dari AI menjadi bagian/ komponen paling penting dari alur permainan (gameplay). Pemain akan melihat pergerakan dari AI dan akan melakukan gerakan berdasarkan hasil gerak AI tersebut. Sehingga kualitas dari suatu game *turn-based strategy* akan sangat bergantung pada sistem AI yang dimiliki khususnya pada kemampuan *pathfinding* dari AI untuk "mencari dan mengejar" pemain. Game-game ini pun pada umumnya telah mencoba mencari solusi paling optimal dalam membuat pencarian jalur AI game lebih efektif dan lebih "pintar" , banyak cara yang dilakukan mulai dari exhaustive search pada game dengan ukuran peta kecil, DFS dan BFS pada game dengan peta linear, algortima Greedy, dll. Disinilah algoritma A* dapat diterapkan pada game agar pergerakan dari AI game menjadi lebih optimal.

Algortima A* dapat diterapkan dengan sedikit modifikasi agar dapat berjalan dengan baik pada AI game *turn-based strategy.* Dengan memakai algoritma A* maka pergerakan dari AI game akan lebih terkalkulasi dan terjamin akan selalu optimal ketika bergerak sehingga akan menyediakan tantangan yang konsisten bagi pemain ketika menghadapi AI. Dengan algoritma A* ini juga maka jalur yang dipilih oleh AI akan optimal dan tidak terjadi kesalahan yang dapat dimanfaatkan oleh pemain akibat jalur yang tidak optimal dari AI game.

Pada makalah ini penulis akan membahas penerapan yang dapat dilakukan pada sistem game untuk memasukkan algoritma A* dalam pencarian jalur jalan AI game pada t*urn-based strategy game* khususnya pada game dengan *tile system map* agar didapatkan solusi jalur paling optimal.

II. TEORI DASAR

A. Video Game

Permainan yang menggunakan interaksi dengan antarmuka pengguna melalui gambar yang dihasilkan oleh peranti video. Permainan video umumnya menyediakan sistem penghargaan – misalnya skor – yang dihitung berdasarkan tingkat keberhasilan yang dicapai dalam menyelesaikan tugas-tugas yang ada di dalam permainan. Video game memiliki beragam genre mulai dari Fighting game, First Person Shooter, Third Person Shooter, Racing Game, Sport Game, Role Playing Game, Action Adventure Game, Real Time Strategy Game, Turn-Based Game,dll. Video game akan di-"mainkan" pada alat/sistem elektronik yang disebut dengan *platform* dan akan menggunakan input device yang bermacam-macam mulai dari joypad, controller, keyboard, mouse,dll.

B. *Turn-Based Strategy Game*

Merupakan salah satu sub-genre dari genre strategy game dimana pemain akan melakukan gerakan secara bergantian sampai semua pemain melakukan aksi yang diinginkan lalu proses permainan akan diulang lagi hingga ditemukan pemenang. Salah satu bentuk dari permainan dengan prinsip turn-based adalah catur serta berbagai macam permainan papan (boardgame). Dalam konsep video game maka game dengan prinsip *turn-based* akan berfokus pada manajemen sumberdaya (resource management) dan manajemen gerak (move management). Manajemen sumber daya (resource management) adalah salah satu konsep yang sangat umum dipakai dalam permainan bergenre *turn-based* game, dimana pemain akan diberikan beberapa sumberdaya yang terbatas dan tujuan pemain adalah memanfaatkan sumber daya yang dimiliki untuk memenangkan permainan atau memiliki sumber daya yang lebih banyak dibandingkan pemain lain untuk setiap giliran. Manajemen gerak (move management) merupakan fokus game berbasis *turn-based* game dimana game akan memberikan pemain sejumlah gerakan yang dapat dilakukan untuk setiap giliran yang dapat dimanfaatkan semaksimal mungkin untuk mencapai tujuan dari game, sistem ini akan memberikan jumlah gerakan yang sama banyak untuk setiap pemain.

Gambar II.1 Contoh game Turn-Based Strategy (Sid Meier's Civilization VI)

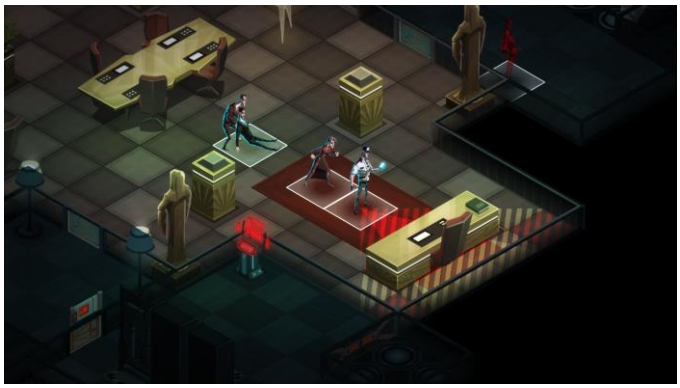
Ada dua komponen dari game turn-based strategy yang akan menjadi fokus dalam makalah ini yaitu AI (artificial intelligence) dan juga peta (Map).

i. AI (Artificial Intelligence)

Dalam game turn-based strategy maka akan dibutuhkan komponen AI, dalam hal ini AI bukan lah benar-benar AI, tapi merupakan komponen permainan yang akan dikontrol oleh algoritma game. AI ini akan menjadi komponen yang dinilai untuk menentukan apakah sebuah game dengan genre turn-based memiliki desain game yang bagus dan memberikan tantangan bagi pemain.

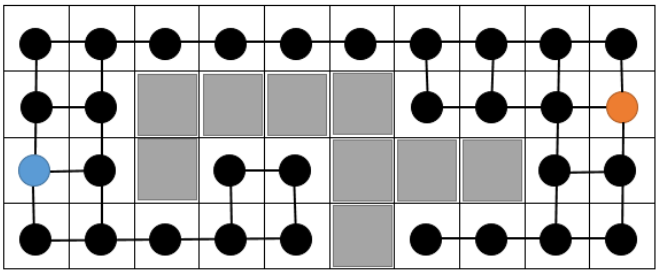
ii. Peta (Map)

Dalam game turn-based strategy maka komponen lain yang diperlukan adalah peta atau lokasi dimana permain akan dilakukan. Dalam peta inilah semua gerakan pemain maupun AI akan dilakukan dan akan berakibat pada peta itu sendiri. Pada permainan berbasis turn-based peta yang digunakan akan bermacam-macam mulai tapi pada umumnya akan memakai sistem petak (Tile System), yang memiliki sistem dimana pemain dapat bergerak sesuai dengan posisi petak sekarang dan pemain hanya dapat bergerak dari petak tersebut ke petak yang ada disamping petak posisi pemain.



Gambar II.2 Contoh game Turn-Based Strategy dengan Tile System Map (Invisible Inc.)

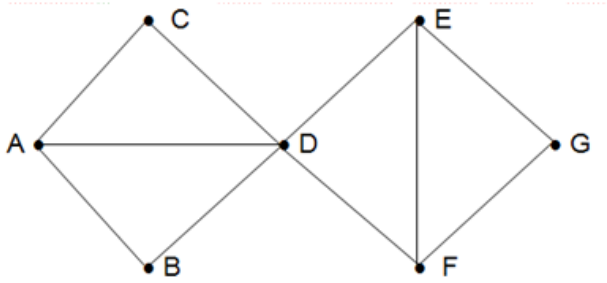
Dalam sistem petak (Tile System) maka bentuk pergerakan dan keterhubungan dari setiap petak akan dapat diilustrasikan dengan bentuk graf, dimana setiap petak akan menjadi simpul dan setiap keterhubungan antar petak akan menjadi sisi. Dalam makalah ini penulis akan fokus pada game turn-based strategy dengan peta *tile system.*



Gambar II.3 Representasi peta game dengan Tile System memakai graf, dengan simpul biru posisi AI, simpul oranye posisi pemain, simpul hitam petak yang bisa dilewati

C. Graf

Secara informal, suatu graf adalah himpunan benda-benda yang disebut "simpul" (vertex atau node) yang terhubung oleh "sisi" (edge). Untuk makalah ini akan berfokus pada graf tak-berarah berbobot.



Gambar II.4 Contoh gambar graf tak berarah

*D. Algoritma A**

Algoritma A* merupakan salah satu algoritma yang digunakan untuk memecahkan permasalahan pathfinding serta untuk penelusuran graf, di mana prosesnya adalah penentuan jalur paling efisien dari satu simpul ke simpul lainnya. Algoritma A* untuk saat ini banyak digunakan untuk pemecahan berbagai masalah dikarenakan performansi dan akurasi dari algoritma ini.

Algoritma A* merupakan *informed search algorithm*, dimana algoritma akan melakukan pemilihan dengan melihan semua pilihan solusi dan akan melakukan pengambilan ketika ditemukan solusi dengan cost paling rendah. Lalu akan dilakukan penelusuran pada simpul tersebut dan akan dilihat lagi solusi yang bisa didapatkan dari simpul tersebut yang mengarah pada simpul tujuan. Proses ini akan terus dilakukan sampai ditemukan simpul yang dituju tersebut.

Proses looping akan dilakukan terus menerus sampai simpul tujuan dicapai dan hal tersebut dilakukan dengan cara mengunjungi simpul yang memiliki cost paling rendah dan hal ini akan ditentukan dengan memakai persamaan :

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

Dimana n merupakan simpul terakhir di jalur saat ini, f(n) merupakan cost dari suatu simpul yang menjadi calon simpul yang akan dipilih, g(n) merupakan cost dari simpul awal untuk mencapai simpul saat ini, serta h(n) yang merupakan nilai heuristik yang memperkirakan cost paling rendah untuk mencapai simpul tujuan. Nilai heuristik adalah nilai yang tidak universal dapat digunakan untuk semua permasalahan A*, tapi nilai heuristik akan tergantung pada masalah yang sedang dihadapi. Syarat yang harus dipenuhi untuk nilai heuristic yang dipakai adalah nilai yang ditentukan/diperkirakan tidak melebihi dari nilai cost sebenarnya untuk mencapai tujuan.

Dalam penyelesaian masalah A* akan digunakan struktur data Queue(antrian) yaitu priority queue yang akan menyimpan urutan pengaktifan dari simpul berikutnya berurutan sesuai dengan cost dari simpul awal ke simpul tersebut, dan ketika suatu simpul diaktifkan maka simpul tersebut akan dikeluarkan dari queue lalu anak-anak simpul dari simpul yang baru saja diaktifkan akan dimasukkan ke dalam queue sesuai dengan cost untuk setiap anak-anak simpul tersebut. Proses ini akan terus dilanjutkan sampai simpul tujuan ditemukan. Setelah itu maka akan didapatkan jalur graf yang memiliki cost paling rendah dan hasilnya merupakan solusi dari permasalahan yang diajukan.

Layaknya BFS, maka penyelesaian A* merupakan complete, dimana algoritma A* akan selalu mendapatkan hasil/solusi dari permasalahan dan juga akan menghasilkan solusi yang optimal.

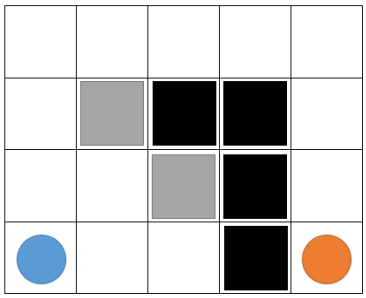
III. IMPLEMENTASI ALGORITMA

Implementasi algoritma A* untuk game *turn-based strategy* yang akan dipaparkan oleh penulis, membutuhkan beberapa hal dan asumsi yang dipakai agar dalam penjelasannya tidak terjadi kebingungan akan implementasi algoritma tersebut. Pertama-tama jenis game *turn-based strategy* yang dipakai merupakan game yang memiliki fokus pada manajemen gerak (move management) sehingga konsep manajemen sumber daya (resource management) tidak akan berpengaruh dan tidak akan dipakai dalam implementasi algoritma. Konsep manajemen gerak (move management) akan dipakai sebagai nilai yang dicari dalam pergerakan AI menuju pemain. Dan dalam hal ini algoritma akan mencari jumlah gerakan dan jalur gerak yang paling minimum(cost terendah) untuk mencapai pemain.

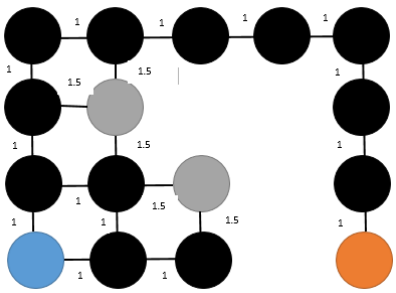
Kedua, implamentasi algoritma A* akan akan dilakukan pada game turn-based strategy yang memakai sistem peta tile sistem, yaitu peta permainan yang dibagi menjadi petak-petak dan pada satu saat pemain akan menduduki sebuah petak dan AI/musuh akan menduduki sebuah petak. Pergerakan yang dapat dilakukan oleh pemain maupun AI akan tergantung pada petak-petak yang berada disamping pemain maupun AI/musuh. Dalam implementasinya algoritma A* akan mencoba mencari jalur perpindahan dari petak ke petak lain yang berawal dari petak yang diduduki oleh AI/musuh sampai dengan petak yang diduduki pemain. Dalam hal ini representasi yang akan dipakai oleh penulis adalah graf tidak berarah yang memiliki komponen yaitu simpul awal yang menjadi representasi petak awal dari AI/musuh yang merupakan titik mulai algoritma dan simpul akhir yang menjadi representasi petak dimana pemain berada serta menjadi simpul tujuan dari simpul awal, dan juga simpul-simpul yang kosong dan dapat dilalui oleh pemain serta AI yang menjadi calon solusi jalur algortima A*.

Komponen lain yang menjadi tambahan untuk sistem peta yang dipakai adalah dengan adanya petak blok dan petak obstacle(halangan). Petak blok adalah petak dalam game yang tidak dapat diduduki oleh pemain ataupun AI, petak ini tidak akan direpresentasikan sebagai simpul dalam graf peta sehingga tidak bisa menjadi salah satu petak yang menjadi jalur solusi algoritma. Petak obstacle adalah petak dalam game yang dapat diduduki dan dilewati oleh pemain maupun AI, tapi memiliki cost pergerakan yang lebih tinggi dibandingkan dengan perpindahan ke petak yang bisa. Pada sistem game petak ini akan memiliki sistem kerja dimana ketika pemain atau AI

bergerak ke petak tersebut, maka akan memakan satu pergerakan tapi ketika pemain atau AI di giliran baru maka pemain tidak dapat bergerak sampai giliran berikutnya. Karena dalam konteks algortima ini cukup sulit untuk melakukan representasi sistem tersebut maka yang dapat akan dilakukan oleh penulis dalam impelentasinya adalah dengan mengubah cost dari pergerakan ke petak obstacle lebih tinggi dibandingkan dengan pergerakan ke petak yang normal. Representasi yang dapat dilakukan untuk petak obstacle akan merupakan dengan simpul graf bisa dengan hubungan ketetanggan dengan simpul lain memiliki bobot lebih tinggi Dengan contoh representasinya sebagai berikut :



Gambar III.1 Representasi peta game dengan, petak abu-abu obstacle, petak hitam blok, titik biru posisi AI, titik oranye posisi pemain



Gambar III.2 peta game diubah menjadi graf berbobot

Untuk algoritma A* itu sendiri yang akan dipakai adalah model algortima A* biasa dengan memakai nilai heuristik yang admissible. Algoritma akan bermula dari simpul awal yaitu petak posisi AI/musuh dan akan berakhir pada petak posisi pemain. Algoritma akan mencari jalur pergerakan dari petak ke petak yang memiliki cost paling rendah. Seperti pada algortima A* untuk berbagai masalah penentuan dari cost suatu simpul adalah dengan memakai persamaan

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

Dimana n merupakan simpul saat ini, f(n) merupakan cost yang diperlukan untuk mencapai simpul tersebut, nilai g(n) adalah cost yang diperlukan untuk mencapai simpul tersebut yang dalam konteks ini adalah jumlah gerakan(move) yang perlu dilakukan untuk mencapai simpul ini, serta h(n) yaitu nilai heuristik yang dipakai sebagai perkiraan cost terendah dari titik simpul tersebut mencapai simpul tujuan/akhir.

Nilai heuristik yang akan dipakai dalam algoritma ini adalah jumlah gerakan/jumlah petak minimum dari suatu petak/simpul ke petak tujuan/simpul tujuan. Nilai heuristic ini akan memakai nilai *Manhattan Distance*, yaitu nilai dari suatu titik ke titik tujuan secara vertical dan horizontal. Jadi ketika titik pertama ada pada (1,3) dan titik ke dua ada pada (3,5) maka Manhattan Distancenya adalah 2+2 yaitu 4 (2 vertikal ke atas dan 2 horizontal ke kanan). Dalam algoritma Manhattan Distance akan diukur dari simpul tertentu ke simpul akhir/tujuan sesuai dengan jumlah petak vertical dan horizontal

Algoritma A* juga akan memakai struktur data berupa priority queue yang akan berisi petak-petak yang akan diaktifkan ketika melakukan pencarian cost terendah menuju simpul akhir.

Garis besar dari cara kerja algoritma A* dalam penerapannya di game *turn-based strategy* adalah sebagai berikut :

1. Algortima akan mulai dengan parameter simpul awal dan simpul akhir/tujuan.
2. Buatlah graf baru yang hanya terdiri dari simpul awal.
3. Buatlah list baru (OpenNode), yang akan menyimpan semua simpul yang perlu dan akan diaktifkan, serta masukkan simpul awal ke dalam list OpenNode.
4. Buatlah list baru (ClosedNode), yang akan menyimpan semua simpul yang telah diaktifkan, sehingga pada keadaan awal list akan kosong.
5. Akan dilakukan looping dengan syarat ketika list OpenNode kosong maka tidak ditemukan solusi, tapi selama list OpenNode tidak kosong maka proses akan berlanjut dan akan terus berulang sampai solusi ditemukan atau list kosong.
6. Keluarkan simpul yang berada pada awal dari list OpenNode dan masukkan ke dalam list ClosedNode.
7. Akan dilakukan pengecekan apakah node yang diaktifkan merupakan simpul tujuan, jika benar maka algoritma akan keluar dari looping lalu akan dilakukan konstruksi solusi.
8. Lalu aktifkan simpul tersebut (misalnya simpul n) dan telusuri semua anak simpul yang berasal dari simpul n tersebut.
9. Ketika semua anak simpul telah ditelusuri maka akan dilakukan perhitungan untuk setiap cost dari anak simpul tersebut.
10. Perhitungan cost tersebut akan sesuai dengan persamaan f(n) = g(n) +h(n) dengan data heuristic telah tersedia.
11. Semua anak simpul dari n akan dimasukkan ke dalam list OpenNode dengan keterurutan dari cost paling kecil sampai paling besar.
12. Proses diulang dari tahap ke 5.

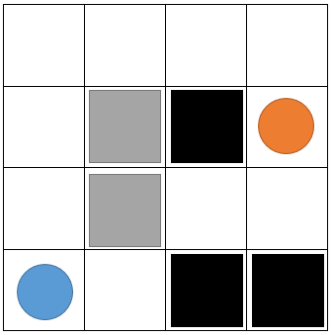
13. Algoritma selesai dengan hasil tidak ditemukan solusi atau telah ditemukan solusi.
14. Akan dilakukan kontruksi jalur solusi yang didapatkan dari list ClosedNode

Sistem priority queue diimplementasikan pada saat pengaturan isi list OpenNode dimana akan diurutkan berdasarkan cost terendah dan ketika dimasukkan simpul-simpul baru maka simpul-simpul tersebut dapat disisipkan diantara simpul yang sudah ada sebelumnya jika cost dari simpul itu memang kecil.

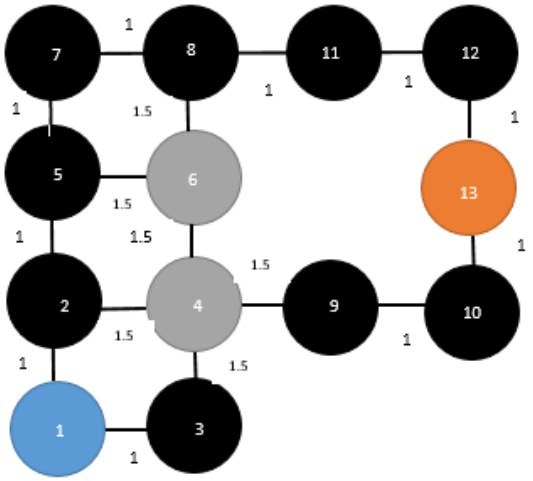
Implementasi Algoritma A* pada game *turn-based strategy* dapat dinyatakan dalam bentuk pseudocode sebagai berikut :

```
function A*(startNode, goalNode) {

OpenNode <- {startNode}
ClosedNode <-{}
currentNode : node

while not (OpenNode == empty) {
      currentNode <- OpenNode[1]

      if ( currentNode == goalNode ) then
            -> break

      Generate each state successorNode
      that  come after currentNode

      for each successorNode of
      currentNode do

            successor_currentcost <-
            g(currentNode) + w(currentNode,
            successorNode)

            if ( successorNode is in the
            OpenNode ) then
                  if g(successorNode) <=
                  successor_currentcost then
                  { -> break }

            else if ( successorNode is in
            the ClosedNode ) then
                  if g(successorNode) <=
                  successor_currentcost
                  then
                  { -> break }

                  ClosedNode.delete
                  (successorNode)

                  OpenNode.add(successorNode)

            else

                  OpenNode.add(successorNode)

                  h(successorNode) <-
                  NilaiHeuristik(successorNode,
                  goalNode)

                  g(successorNode) <- successor_current_cost

                  successorNode.parent <- currentNode

                  ClosedNode.add(currentNode)

if not ( currentNode == goalNode ) then
      -> ERROR

}
```

Implementasi dari algoritma A* pada game *turn-based strategy* dapat diilustrasikan dengan contoh ini :



Gambar III.3 contoh peta game Tile System



Gambar III.4 peta game diubah menjadi graf berbobot

Lalu setelah itu dapat dibuatlah nilai heuristik untuk setiap simpul menuju simpul akhir/tujuan dengan menggunakan nilai Manhattan Distance. Hasil dari nilai-nilai heuristik illustrasi graf peta di atas diuraikan pada tabel ini :

N (KE SIMPUL 13)	NILAI
1	5
2	4
3	4
4	3
5	3
6	2
7	4
8	3
9	2
10	1
11	2
12	1
13	0

Tabel III.1 Nilai heuristik dari graf peta game

Setelah nilai heuristic dan graf peta didapatkan maka proses dari algortima A* dapat dilakukan, proses iterasinya dapat dituliskan pada table di bawah ini dengan uraian setiap iterasi dari simpul awal sampai dengan simpul akhir/tujuan :

ITERASI	SIMPUL DIEKSPAN	SIMPUL HIDUP (OpenNode)
1	1	2 $_{1}$ = 1+4 = 5 3 $_{1}$ = 1+4 = 5
2	2 $_{1}$	3 $_{1}$ = 5 5 $_{2-1}$ = 2 + 3 = 5 4 $_{2-1}$ = 2.5 + 3 = 5.5
3	3 $_{1}$	5 $_{2-1}$ = 521 = 5 4 $_{2-1}$ = 5.5 5 $_{3-1}$ = 2.5 + 3 = 5.5
4	5 $_{2-1}$	4 $_{2-1}$ = 5.5 4 $_{3-1}$ = 5.5 (sama 4 $_{2-1}$) 6 $_{5-2-1}$ = 3.5 + 2 = 5.5 7 $_{5-2-1}$ = 3 +4 = 7
5	4 $_{2-1}$	6 $_{5-2-1}$ = 5.5 6 $_{4-2-1}$ = 4 + 2 = 6 9 $_{4-2-1}$ = 4 + 2 = 6 7 $_{5-2-1}$ = 7
6	6 $_{5-2-1}$	6 $_{4-2-1}$ = 6 9 $_{4-2-1}$ = 6 7 $_{5-2-1}$ = 7 4 $_{6-5-2-1}$ = 5 +3 = 8 8 $_{6-5-2-1}$ = 5 +3 = 8
7	6 $_{4-2-1}$	9 $_{4-2-1}$ = 6 7 $_{5-2-1}$ = 7 4 $_{6-5-2-1}$ = 8 8 $_{6-5-2-1}$ = 8 5 $_{6-4-2-1}$ = 5.5 + 3 = 8.5 8 $_{6-5-2-1}$ = 5.5 + 3 = 8.5

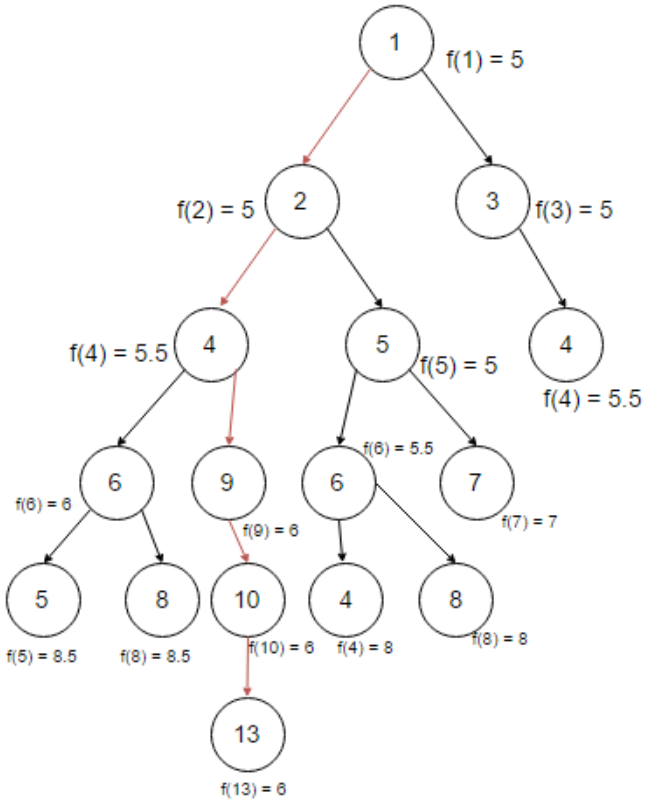
8	9 4-2-1	10 9-4-2-1 = 5 + 1 = 6 7 5-2-1 = 7 4 6-5-2-1 = 8 8 6-5-2-1 = 8 5 6-4-2-1 = 8.5 8 6-5-2-1 = 8.5
9	10 9-4-2-1	13 10-9-4-2-1 = 6 + 0 = 6 7 5-2-1 = 7 4 6-5-2-1 = 8 8 6-5-2-1 = 8 5 6-4-2-1 = 8.5 8 6-5-2-1 = 8.5
10	13 10-9-4-2-1	SELESAI

Tabel III.2 Iterasi dari algoritma A* pada peta game

Jalur solusi yang didapatkan adalah 1 – 2 – 4 – 9 – 10 – 13

Cost yang didapatkan untuk mencapai simpul akhir adalah 6

Proses penyelesaian dapat juga digambarkan dengan pohon ruang solusi :

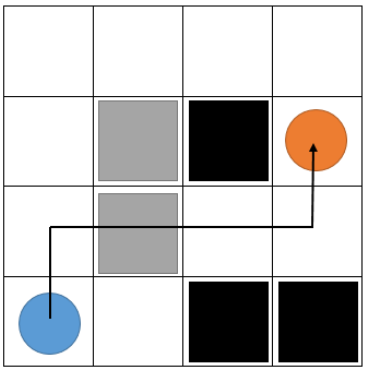


Gambar III.5 pohon ruang solusi algoritma A* peta game

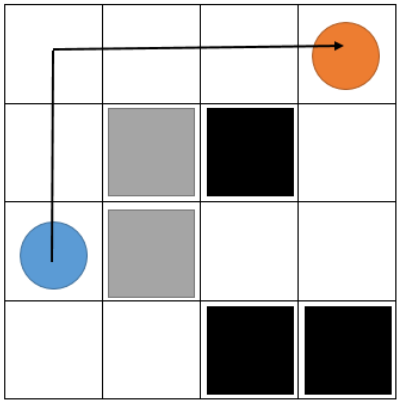
Dengan demikian maka proses dari algoritma A* telah selesai dan jalur solusi telah didapatkan yang berawal dari simpul awal (posisi AI) ke simpul akhir (posisi pemain). Didapatkan pula jumlah gerakan(cost) yang perlu dilakukan oleh AI untuk mencapai pemain.

Secara algoritma maka proses penentuan gerakan dari AI telah selesai dikarenakan sudah ditemukan jalur paling singkat dari AI tersebut menuju pemain, tapi game akan terdiri dari gerakan AI dan juga gerakan pemain. Gerakan-gerakan tersebut akan dilakukan secara bergantian sehingga AI hanya dapat melakukan gerakan menuju pemain sesuai jalur yang ditentukan algoritma A* sebesar satu petak. Dikarenakan hal ini maka sistem game akan menjalankan proses algoritma A* setiap kali giliran dari AI aktif, sehingga jalur yang paling optimum dari AI menuju pemain akan selalu paling optimal dan selalu benar dimanapun pemain bergerak pada gilirannya. Proses besar pergerakan dan pathfinding AI akan berbentuk loop dimana AI akan menunggu giliran lalu akan menjalankan algoritma A* lalu akan bergerak menuju pemain sesuai dengan hasil algoritma, lalu proses akan terus dilakukan sampai AI

mencapai pemain. Proses sistem game tersebut dapat digambarkan dengan :



Gambar III.6 peta game dan jalur yang ditentukan oleh algoritma A*



Gambar III.7 peta game dan jalur yang ditentukan oleh algoritma A* setelah dilakukan pergerakan dari AI dan juga pemain

Perlu diperhatikan juga bahwa algoritma yang dibahas dan telah diimplementasikan akan berpengaruh pada pergerakan dan pathfinding dari AI untuk menuju pemain, tapi masih banyak sistem lain yang akan berpengaruh dalam perilaku AI khususnya pada game *turn-based strategy* tapi komponen tersebut tidak akan dibahas oleh penulis karena tidak berkaitan dengan algoritma A* dan proses *pathfinding* dari AI itu.

IV. KESIMPULAN

Dengan semakin berkembangnya teknologi komputasi maka semakin banyak pula permasalahan yang dapat diselesaikan oleh algoritma-algoritma, hal ini juga termasuk dalam pengembangan game. Dalam game turn-based pergerakan dari AI sangatlah penting dan menjadi salah satu tolak ukur dari kualitas game. Hal ini dapat ditingkatkan kualitasnya dengan memakai algoritma A* untuk melakukan pathfinding oleh AI game untuk mencapai posisi pemain.

REFERENSI

[1] Slide A-Star-Best-FS-dan-UCS-(2016) oleh Rinaldi Munir diakses pada 16 Mei 2017

[2] "EfficientPoint-to-PointShortestPath Algorithms" *(PDF)*. from Princeton University diakses pada 16 Mei 2017

[3] http://www.geeksforgeeks.org/a-search-algorithm/ diakses pada 16 Mei 2017

[4] http://gameranx.com/wp-content/uploads/2016/09/CivilizationVINorway2-1400x788.jpg diakses pada 16 Mei 2017

[5] http://media.kotaku.foxtrot.future.net.uk/wp-content/uploads/sites/52/2015/12/invisible-inc-favourite-games-of-2015-620x349.png diakses pada 17 Mei 2017

[6] https://adrianstarkblog.files.wordpress.com/2014/04/3.png diakses pada 17 Mei 2017

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, 17 Mai 2017

Trevin Matthew Robertsen - 13515027