Penerapan dan Perbandingan Algoritma Path-finding dalam Kecerdasan Buatan Hantu pada Permainan Pac-Man

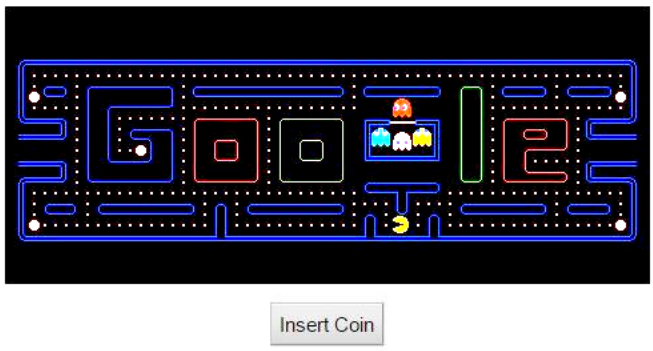
Joshua Aditya Kosasih
Program Studi Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jl. Ganesha 10 Bandung 40132, Indonesia
13514012@std.stei.itb.ac.id

***Abstrak*—Permainan Pac-Man adalah salah satu permainan ikonik yang sangat terkenal. Hantu-hantu musuh dari Pac-Man dibuat oleh perancangnya dengan konsep kecerdasan buatan yang unik dan mempunyai personalitasnya masing-masing. Algoritma path-finding mungkin dapat diterapkan pada setiap hantu-hantu untuk menentukan arah gerak mereka. Algoritma path-finding yang dipakai haruslah efisien dan tepat. Dua buah algoritma path-finding yang memenuhi syarat tersebut adalah A* dan Greedy, masing-masing dengan kelebihan kekurangannya. Namun salah satu dari kedua algoritma tersebut lebih cocok untuk dipakai untuk permainan ini. Selain itu terdapat kemungkinan bahwa algoritma path-finding tidak dapat diterapkan pada semua hantu-hantu unik tersebut.**

Kata Kunci—Kecerdasan Buatan, A-Star, Greedy, Pac-Man

I. PENDAHULUAN

Pac-Man adalah sebuah permainan yang dibuat oleh Namco dan muncul pertama kali di Jepang tahun 1980. Pac-Man merupakan salah satu game terpopuler sejak dirilis hingga sekarang dan dilihat sebagai ikon klasik budaya video game dari tahun 1980an. Bahkan sampai abad ke-20 ini permainan Pac-Man masih dimainkan oleh beberapa kalangan.



Gambar 1. Google Doodle Pac-Man, 2010

Pac-Man tercatat sebagai pengubah sejarah video game. Permainan tersebut juga merupakan salah satu video game dengan keuntungan terbesar, dengan rekor menghasilkan keuntungan 2.5 miliar dollar (33,4 triliun rupiah dengan kurs saat ini) di tahun 1990-an. Selain itu sebuah survey mendapatkan bahwa, *brand* Pac-Man dikenali kurang lebih 94% responden dari seluruh konsumen di Amerika.

Dalam permainan tersebut, pengguna memainkan Pac-Man menjelajahi sebuah labirin, memakan titik-titik yang tersebar di dalamnya. Jika pemain berhasil memakan seluruh titik-titik tersebut, maka Pac-Man akan pindah ke tingkat berikutnya.

Tetapi terdapat 4 musuh Pac-Man berbentuk hantu yang menjelajahi labirin berusaha menangkap Pac-Man. Keempat nama musuhnya adalah Blinky, Pinky, Inky, dan Clyde. Jika salah satu musuh mengenai Pac-Man, maka pemain akan kehilangan satu nyawa. Di sekitar labirin juga terdapat titik-titik yang berukuran lebih besar (disebut pil energi) yang jika dimakan dapat memberikan kemampuan pada pemain untuk memakan musuh-musuhnya selama jangka waktu tertentu. Tentunya musuh-musuhnya yang telah dimakan akan muncul kembali dan kembali mengejar Pac-Man.

Selama permainan, para hantu akan terus mengejar Pac-Man dalam gerakan yang tidak mudah ditebak. Pengejaran tersebut menggunakan algoritma *path-finding*. Dalam bab-bab berikutnya akan dibahas kandidat algoritma yang dipakai untuk mempercerdas hantu-hantu dalam mengejar Pac-Man. Algoritma *path-finding* yang akan dibahas adalah:

- A *
- Greedy

Algoritma Dijkstra tidak dibahas karena dalam labirin, biaya jarak antar node sama besar sehingga Dijkstra hanya akan bekerja seperti BFS dan tidak efisien sama sekali.

Pengejaran yang dilakukan hantu-hantu tidak monoton dan susah diprediksi pemain. Hal ini disebabkan kecerdasan buatan yang diimplementasi untuk masing-masing hantu dibuat dengan sangat baik (mengingat permainan ini dibuat tahun 90an).

Pembuat permainan ini, Toru Iwatani pun mengaku bahwa bagian perancangan yang tersusah adalah saat merancang kecerdasan buatan hantu-hantu.

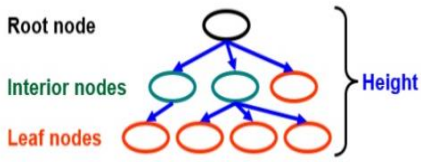
"The algorithm for the four ghosts who are dire enemies of the Pac Man - getting all the movements lined up correctly. It

was tricky because the monster movements are quite complex. This is the heart of the game. I wanted each ghostly enemy to have a specific character and its own particular movements, so they weren't all just chasing after Pac Man in single file, which would have been tiresome and flat…" – Toru Iwatani

II. Dasar Teori

A. *Pohon Pencarian*

Dalam algoritma path-finding, proses pencarian dapat digambarkan dalam sebuah struktur pohon pencarian.



Gambar 2. Ilustrasi pohon pencarian

Pencarian dimulai dari root menuju ke salah satu daun yang merupakan target / tujuan yang dicari. Pada path-finding, pohon pencarian dibuat seiring algoritma berjalan karena seluruh jalur belum diketahui.

B. *A **

Algoritma A* (A-Star) adalah salah satu algoritma yang menggunakan fungsi heuristik. Dengan menerapkan konsep heuristik, algoritma ini membuang langkah-langkah yang tidak perlu dengan pertimbangan bahwa langkah-langkah yang dibuang sudah pasti merupakan langkah yang tidak akan mencapai solusi yang optimal.

Algoritma A* membangkitkan node yang paling mendekati solusi. Node ini kemudian disimpan suksesornya ke dalam list sesuai dengan urutan yang paling mendekati solusi terbaik. Kemudian, node pertama pada list diambil, dibangkitkan suksesornya dan kemudian suksesor ini disimpan ke dalam list sesuai dengan urutan yang terbaik untuk solusi.

Algoritma ini dapat mengunjungi suatu node secara mendalam selama node tersebut merupakan node yang terbaik. Jika node yang sedang dikunjungi ternyata tidak mengarah kepada solusi yang diinginkan, maka akan melakukan runut balik ke arah node akar untuk mencari node anak lainnya yang lebih menjanjikan dari pada node yang terakhir dikunjungi. Bila tidak ada juga, maka akan terus mengulang pencarian ke arah node akar sampai ditemukan node yang lebih baik untuk dibangkitkan suksesornya.

Untuk dapat memilih node yang terbaik, algoritma ini menghitung nilai-nilai setiap node pada pohon pencariannya. Perhitungan penilaian setiap node dilakukan dengan menggunakan fungsi evaluasi *f(n)*, dimana:

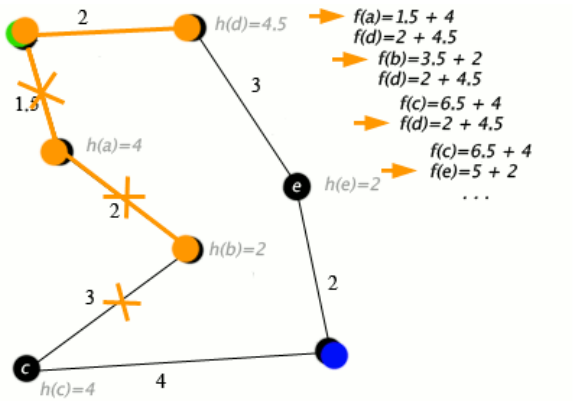
$f(n) = g(n) + h(n)$

$g(n)$ = biaya untuk mencapai node n saat ini

$h(n)$ = perkiraan biaya untuk mencapai target dari node n

Fungsi $h(n)$ inilah yang telah disebutkan sebelumnya sebagai fungsi heuristic. Fungsi heuristic dapat dibuat dengan berbagai pertimbangan tetapi yang terpenting harus bersifat admissible.

Dalam path-finding, jalur solusi yang optimum haruslah memiliki biaya seminimal mungkin. Maka node yang terbaik adalah node dengan nilai biaya terkecil. Agar fungsi heuristic admissible, nilai dari $h(n)$ tidak boleh overestimated (tidak boleh lebih besar dari biaya sesungguhnya). Semakin dekat nilai $h(n)$ dengan biaya sesungguhnya akan menjadikan pencarian A* semakin baik (efisien).



Gambar 3. Pencarian solusi menggunakan A*

C. *Greedy*

Greedy adalah sebutan untuk algoritma yang menggunakan pendekatan penyelesaian masalah dengan memilih pilihan terbaik yang ada pada setiap langkahnya. Pada kebanyakan kasus, algoritma greedy tidak akan menghasilkan solusi paling optimal, tetapi algoritma greedy biasanya memberikan solusi yang mendekati nilai optimum dalam waktu yang sangat cepat dibanding algoritma lainnya.

Dalam pembahasan selanjutnya, algoritma greedy yang dimaksud adalah Greedy Best-first Search. Nilai Best-first Search akan didapat dari fungsi heuristic. Tetapi tidak seperti A*, Greedy tidak selalu menghasilkan nilai optimum. Hal ini disebabkan pada Greedy Best-first Search, $g(n) = 0$. Sehingga $f(n) = h(n)$.

III. Analisis Kecerdasan Buatan Hantu-hantu pada Permainan Pac-Man

A. *Variasi Pergerakan Hantu*

Hantu-hantu musuh Pac-Man dibuat dengan konsep personalitas / karakter yang berbeda-beda. Mereka diprogram untuk memiliki artificial intelligence yang berbeda. Perbedaan tersebut dapat dilihat dari pergerakan-pergerakannya.

Gambar 4. Hantu-hantu di Pac-Man

Pergerakan hantu-hantu dalam Pac-Man memiliki beberapa variasi:

- **Blinky** adalah hantu merah yang selalu mengejar Pac-Man. Sering dianggap sebagai pemimpin para hantu
- **Pinky** adalah hantu pink yang sering memposisikan dirinya agar menghadang Pac-Man.
- **Inky** adalah hantu biru cerah yang pergerakannya lebih susah ditebak. Kadang dia mengejar Pac-Man seperti Blinky, kadang dia memposisikan dirinya di depan seperti Pinky, bahkan kadang dia pergi sendiri kemana-mana seperti Clyde.
- **Clyde** adalah hantu oranye yang paling bodoh. Dia akan mengejar Pac-Man seperti Blinky, tetapi jika berada dekat dengan Pac-Man dia akan pergi sendiri kemana-mana.

B. Pola Umum Pergerakan

Walaupun memiliki karakteristik pergerakan yang berbeda-beda, hantu-hantu tersebut memiliki beberapa kesamaan. Setiap hantu seringnya memiliki suatu petak / lokasi tujuan masing-masing yang perlu dicapai. Dan karakteristik mereka yang berbeda-beda dipengaruhi usah mereka untuk mencapai petak tujuan tersebut.

Semua hantu memiliki metode-metode yang identic untuk mencapai petak tujuan mereka. Tetapi karakteristik pergerakan mereka menjadi berbeda-beda akibat cara pemilihan target petak mereka.

C. Mode Pergerakan Hantu

Terdapat 3 mode pergerakan hantu yaitu:

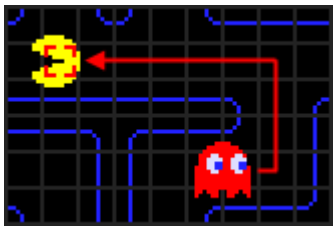
- Kejar
- Menyebar
- Kabur

Mode kejar adalah mode paling umum, yaitu saat hantu-hantu mengejar Pac-Man. Mode menyebar adalah mode saat pemain seperti diberi istirahat dari kejaran hantu. Dalam mode ini hantu-hantu berhenti mengejar Pac-Man dan pergi menyebar ke sudut-sudut labirin. Mode kabur aktif saat Pac-Man telah memakan pil energi. Saat mode ini aktif, gerakan hantu akan melambat dan mereka akan bergerak secara pseudorandom menjauhi Pac-Man.

Selama permainan berlangsung, pemain tidak akan terus-menerus dikejar oleh hantu-hantu. Akan ada selang waktu dimana hantu akan masuk dalam mode menyebar dan berhenti mengejar pemain. Hal ini memang disengaja agar pemain tidak tertekan.

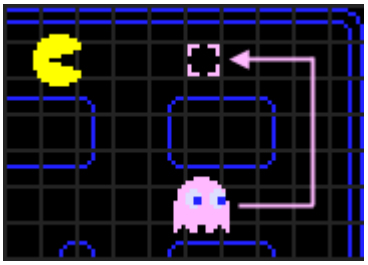
D. Karakteristik Blinky (Hantu Merah)

Blinky memilih petak posisi Pac-Man sebagai petak target nya. Karena hal tersebut maka Blinky akan sering terlihat mengejar Pac-Man kemanapun dia pergi.


Gambar 5. Petak target Blinky

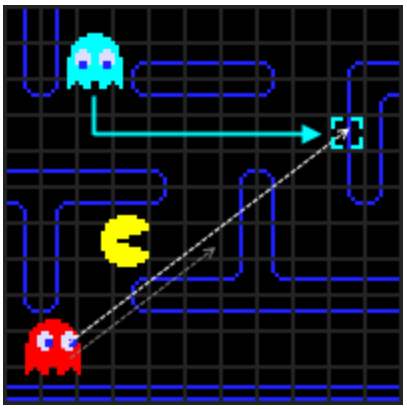
E. Karakteristik Pinky (Hantu Pink)

Petak target Pinky dipilih berdasarkan posisi Pac-Man dan juga arah kemana mulut Pac-Man menunjuk. Petak target yang dipilih adalah petak berjarak 4 satuan dari depan Pac-Man. Karena pilihan petak tersebut, pinky akan sering seolah-olah menghadang pemain daripada mengejarnya.


Gambar 6. Petak target Pinky

F. Karakteristik Inky (Hantu Biru Muda)

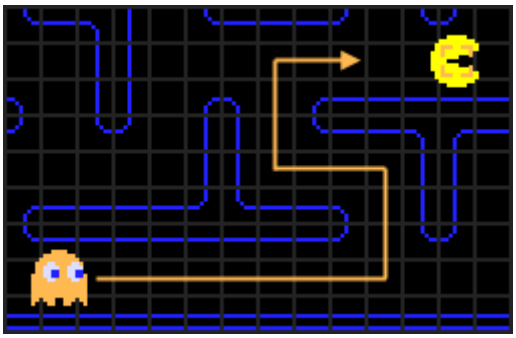
Petak pilihan Inky adalah yang paling rumit untuk dihitung. Inky memilih petak target berdasarkan posisi Blinky, posisi Pac-Man dan arah orientasi Pac-Man. Petak target dihitung dengan pertama ambil petak berjarak 2 satuan di depan Pac-Man (seperti untuk Pinky). Lalu tarik garis lurus dari posisi Blinky ke petak tsb. Kemudian tarik garis tsb (melanjutkan arah tarikan dari petak tsb) sampai 2 kali panjang mula-mula. Di tempat ujung garis tersebut beradalah, dipilih petak target Inky. Karena factor pemilihan petak target yang dinamis, gerakan Inky lebih susah diprediksi.



Gambar 7. Petak target Inky

G. *Karakteristik Clyde (Hantu Oranye)*

Clyde memiliki 2 buah submode yang berbeda pada mode kejarnya. Clyde berganti submode berdasarkan jaraknya dengan Pac-Man. Jika jaraknya dengan Pac-Man lebih besar dari 8 satuan petak, dia akan memilih petak posisi Pac-Man sebagai petak targetnya (mirip Blinky). Tetapi jika Clyde sudah berjarak dekat dengan Pac-Man, ia akan memilih petak targetnya sama dengan petak target saat mode menyebar.



Gambar 8. Petak target Clyde

IV. Penerapan dan Perbandingan Algoritma Path-finding

A. *Penerapan Algoritma Path-finding*

Dari bahasan bab sebelumnya, didapat bahwa algoritma path-finding diperlukan agar hantu-hantu tersebut dapat mencapai petak target mereka masing-masing. Dengan algoritma tersebut, mereka dapat mencapai target mereka melalui jalur yang tepat dan optimal.

Berikut pseudocode cara kerja algoritma path-finding pada saat hantu Blinky berada dalam mode kejar:

```
while (mode = mode_kejar) do
begin
    target.set(pacman.posX(), pacman.posY())
    jalur = find_path_to(target)
    move_to(jalur.first())

    if (pacman.energized()) then
           mode = mode_kabur
    if (timer.istirahat()) then
           mode = mode_menyebar
end
```

Fungsi find_path_to itulah yang perlu diimplementasi dengan salah satu algoritma path-finding.

B. *Fungsi Heuristik*

Kedua buah algoritma path-finding yang dibahas memakai fungsi heuristic *h(n)*. Terdapat dua pilihan fungsi heuristic yang dapat digunakan:

- Euclidean distance
- Manhattan distance

Euclidean distance menghasilkan nilai jarak dari titik awal sampai tujuan dengan menggunakan fungsi phytagoras.

$$\mathrm{d}(\mathbf{p},\mathbf{q}) = \sqrt{(q_1-p_1)^2+(q_2-p_2)^2}.$$

Manhattan distance mendapatkan hasil jaraknya hanya dengan menjumlahkan perbedaan absis dan ordinat antara titik awal dan tujuan.

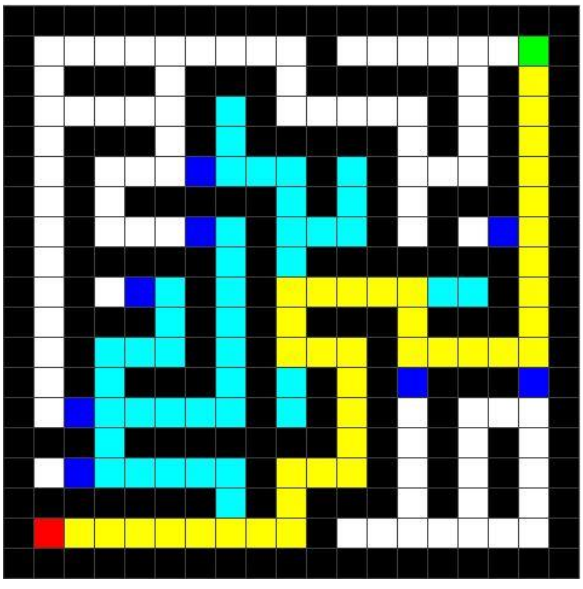
$$d_1(\mathbf{p},\mathbf{q}) = |p_1-q_1| + |p_2-q_2|$$

Fungsi heuristik yang sesuai adalah Manhattan distance, karena lebih mendekati nilai biaya sesungguhnya (yaitu untuk mencapai lokasi tujuan, hantu-hantu harus mengikuti belokan-belokan dan tidak bisa pergi menembus tembok). Fungsi Manhattan distance juga tidak akan pernah overestimate dan maka dari itu adalah admissible.

C. *Pengujian Perbandingan A* dengan Greedy*

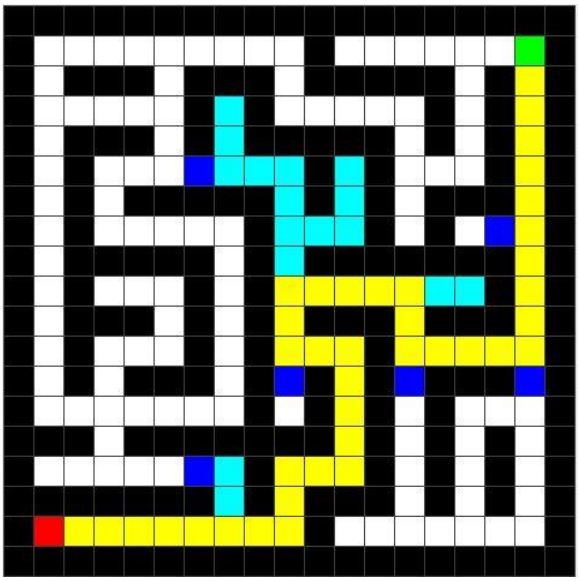
Untuk menguji perbandingan efisiensi dan optimalitas dari algoritma A* dan Greedy dalam menyelesikan pencarian jalur dalam labirin, penulis menggunakan alat bantu visualisasi yang dibuat oleh Nikos Karnagias, dari Hellenic Open University, Athena.

Dalam program visualisasi tersebut, algoritma A* dan Greedy yang diimplementasikan mirip dengan penjelasan yang terdapat di bab 2. Berikut hasil pengujiannya:



Gambar 9. Visualisasi A*

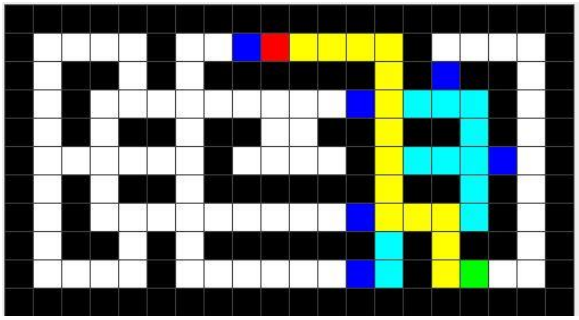
Algoritma A* pada labirin berukuran 20x20 petak tersebut membangkitkan 80 node pada pohon pencariannya. Solusi yang didapat adalah jalur yang diwarnai kuning (mulai dari kotak merah dengan tujuan kotak hijau). Jalur tersebut berjarak 40 petak (optimal).



Gambar 10. Visualisasi Greedy

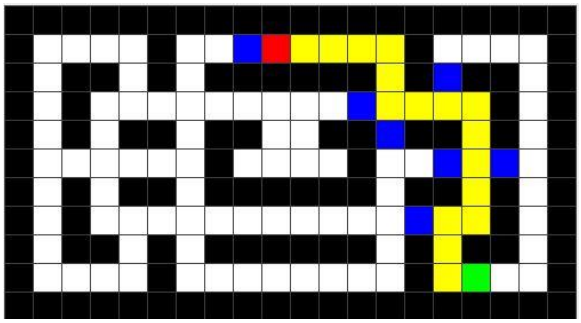
Algoritma Greedy, pada labirin yang sama membuat 56 node pada pohon pencariannya. Dengan solusi yang didapat juga bernilai optimum (dengan jarak 40 petak).

Untuk lebih mensimulasikan proses path-finding pada labirin Pac-Man, dibuat labirin yang lebih kecil (berukuran 20x11) dengan tembok penghalang yang lebih terputus-putus.



Gambar 11. Visualisasi A* pada labirin simulasi Pac-Man

Algoritma A* berhasil mencapai solusi optimum (dengan jarak 15 petak) dengan membangkitkan 26 node selama pencarian.



Gambar 12. Visualisasi Greedy pada labirin simulasi Pac-Man

Sedangkan algoritma Greedy mencapai sebuah solusi yang kurang optimum (berjarak 17 petak) tetapi dengan membangkitkan hanya 17 node selama pencariannya.

Dari hasil pengujian, didapat bahwa hasil Greedy yang kurang optimal dapat di kompromi dengan waktu pencarian yang lebih cepat. Contoh hasil solusi 2 petak lebih jauh (12% lebih panjang dari optimal), tetapi jumlah node yang dibangkitkan lebih kecil (34% lebih efisien) dari solusi A*.

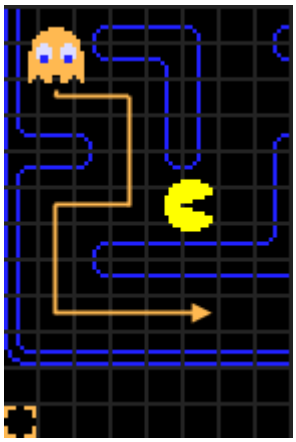
Untuk kasus dimana labirin lebih kompleks, Greedy dapat menghasilkan solusi yang jauh lebih buruk. Tetapi kasus yang dipakai sekarang adalah kasus labirin yang lebih sederhana (tembok penghalang pendek-pendek dan terdapat banyak persimpangan) yaitu labirin Pac-Man. Maka dari itu algoritma path-finding yang paling sesuai untuk permainan Pac-Man adalah algoritma Greedy Best First-search.

D. Kelemahan Algoritma Path-finding

Meskipun bisa didapat pergerakan yang optimum dengan proses yang efisien, algoritma path-finding mempunyai satu kelemahan yang besar. Kelemahan tersebut adalah saat path-finding tidak dapat mencapai target tujuannya.

Ada kalanya petak target hantu berada pada lokasi yang tidak dapat dicapai, contohnya berada diluar labirin atau berada di dalam tembok penghalang. Kondisi petak target berada diluar labirin paling umum terjadi saat hantu-hantu bergerak dalam mode menyebar.

Saat mode menyebar, hantu-hantu pergi ke sudut labirin masing-masing. Misalnya, Clyde akan pergi ke ujung kiri bawah labirin. Hal ini disebabkan petak targetnya berada diluar labirin, dekat ujung kiri bawahnya.



Gambar 13. Petak target Clyde saat menyebar

Jika dilihat lagi pseudocode cara kerja path-finding hantu.

```
//.....
    jalur = find_path_to(target)
    move_to(jalur.first())
//.....
```

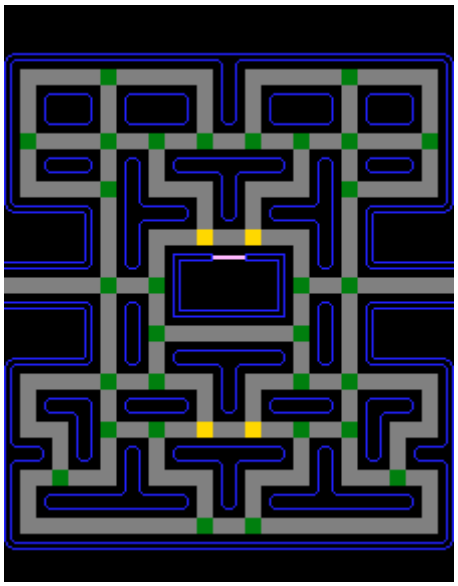
Saat algoritma path-finding tidak dapat menemukan jalur ke petak tujuan, fungsi `find_path_to` akan mengembalikan nilai null. Kode pun akan terus berjalan ke baris berikutnya saat hantu harus bergerak (dengan `move_to`). Tetapi akan terjadi runtime error karena program mengakses nilai null.

Tentu runtime error dapat dihindari dengan memasang pengaman '`if not null`', tetapi hantu tetap tidak akan tahu harus bergerak kearah mana. Atau mungkin masalah ini dapat 'diselesaikan' dengan menaruh petak tujuan ditempat yang dapat diakses. Tetapi hal ini akan menambah banyak kerumitan program. Petak yang tidak dapat diakses ini akan sering didapat pada saat Inky/Pinky telah menghitung petak tujuannya. Petak tujuannya akan sering berada di dalam tembok ataupun diluar labirin.

E. Solusi Original: Cara Mencapai Petak

Satu-satunya solusi termudah adalah dengan menggunakan algoritma original milik Toru Iwatani. Arah pergerakan hantu-hantu tidak diatur setiap saat dalam algoritmanya, tetapi hanya pada saat hantu mencapai setiap persimpangan yang ada. Petak-petak dalam labirin yang merupakan persimpangan ditandai dan setiap hantu yang melewati petak tersebut akan menghitung arah yang akan ia ambil.

Saat hantu mencapai petak persimpangan, arah geraknya akan ditentukan oleh sebuah algoritma Greedy yang sangat sederhana. Perlu dicatat bahwa algoritma Greedy ini sangat berbeda dengan Greedy Best First-Search yang telah dibahas sebelumnya. Algoritma Greedy ini akan langsung menghitung petak-petak jalur dari persimpangan apakah lebih dekat dengan petak tujuan atau tidak menggunakan heuristic Manhattan distance. Jika sudah mengambil arah gerak, hantu-hantu itu akan terus maju mengikuti arah gerak tsb sampai di petak persimpangan berikutnya.



Gambar 14. Petak-petak persimpangan labirin

V. KESIMPULAN

Kecerdasan buatan hantu-hantu dalam permainan Pac-Man memiliki suatu kemiripan yaitu mempunyai cara untuk mencapai suatu petak tujuan. Algoritma path-finding dapat dipakai agar hantu-hantu dapat mencapai petak tujuan mereka masing-masing. Dari berbagai algoritma path-finding yang ada, algoritma Greedy Best-first Search adalah yang paling sesuai. Tetapi ada kalanya petak tujuan hantu-hantu tersebut setelah dikalkulasi tidak dapat dicapai oleh algoritma path-finding. Hal ini menyebabkan algoritma path-finding biasa tidak dapat diterapkan dalam kecerdasan buatan hantu-hantu pada permainan Pac-Man. Walaupun begitu, hal ini dapat diakali dengan membuat agar setiap petak tujuan hantu harus dapat diakses, meskipun akan menambah banyak kerumitan pada pembuatan program.

VI. TERIMA KASIH

Saya berterimakasih pada Tuhan Yang Maha Esa karena berkat bantuan-Nya makalah ini dapat selesai. Saya juga berterimakasih pada dosen saya, Pak Rinaldi Munir dan Bu Nur Ulfa Maulidevi yang telah mengajari saya tentang algoritma-algoritma path-planning yang menginspirasi saya untuk mengambil topik penerapan dan perbandingan algoritma path-finding sebagai topik makalah saya. Terima kasih juga untuk segenap keluarga dan teman-teman yang telah membantu menguatkan saya dalam menyelesaikan makalah ini.

REFERENSI

- Trueman, Doug (November 10, 1999). "The History of Pac-Man". GameSpot.
- Heuristics
  http://theory.stanford.edu/~amitp/GameProgramming/Heuristics.html
  diakses tanggal 6 Mei 2016

- Path Finding Algorithms (DFS, BFS, A*, Greedy, Dijkstra) in Java
  https://www.youtube.com/watch?v=CLbqqb53DLA
  diakses tanggal 6 Mei 2016
- Game Internals – Understanding Pac-Man Ghost Behaviour
  http://gameinternals.com/post/2072558330/understanding-pac-man-ghost-behavior
  diakses tanggal 6 Mei 2016
- PAC-MAN Museum
  http://pacmanmuseum.com/
  diakses tanggal 7 Mei 2016

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, 8 Mei 2016

Joshua Aditya Kosasih, 13514012