Penerapan Algoritma A* dalam Fitur *Auto-Move Player* pada *Role-Playing Game*

Christopher Justine William 13519006
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jalan Ganesha 10 Bandung
13519006@std.stei.itb.ac.id

***Abstract*—Game merupakan sarana hiburan yang banyak dimainkan oleh orang-orang untuk mengisi waktu luang, ketika ingin melepas lelah, atau hanya sekedar mencari hiburan. Salah satu jenis game yang cukup terkenal dan banyak diminati adalah *Role Playing Game* atau biasa disingkat dengan RPG. Game ini juga memungkinkan pemain untuk dikendalikan dengan bebas dalam suatu peta. Setiap pemain memiliki koordinat yang valid di dalam peta game tersebut dan dapat berpindah dari suatu koordinat ke koordinat yang dituju. Dengan fitur *auto move player,* seorang pemain dapat memilih suatu lokasi yang akan dituju dan secara otomatis sistem akan menggerakkan pemain tersebut ke lokasi yang dituju dengan memilih rute tercepat tanpa perlu dikendalikan oleh pemain. Salah satu algoritma yang cukup banyak digunakan untuk menyelesaikan persoalan *pathfinding* adalah algoritma A* yang memanfaatkan pendekatan heuristik.**

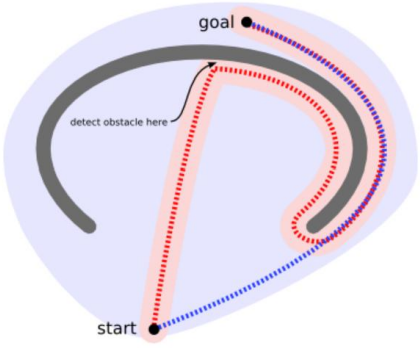
Keywords—algoritma; A*; auto-move player; rute; heuristik

I. PENDAHULUAN

Game merupakan sarana hiburan yang banyak dimainkan oleh orang-orang untuk mengisi waktu luang, ketika ingin melepas lelah, atau hanya sekedar mencari hiburan. Industri game telah berkembang dengan begitu pesat pada zaman ini dikarenakan banyaknya orang-orang yang tertarik untuk memainkan game tergantung dari jenis game-nya. Salah satu jenis game yang cukup terkenal dan banyak diminati adalah *Role Playing Game* atau biasa disingkat dengan RPG.

Gambar 1. Ilustrasi Game RPG 2D
Sumber: https://www.pinterest.com/pin/404479610280501880/

Dalam game RPG, pemain biasanya akan diminta untuk memainkan sebuah peran untuk menyelesaikan *Quest* atau pekerjaan. Ketika sudah berhasil menyelesaikan suatu *Quest*, player biasanya juga akan mendapatkan *reward*. Game ini juga memungkinkan pemain untuk dikendalikan dengan bebas dalam suatu sistem peta (2D atau 3D). Setiap pemain memiliki koordinat yang valid di dalam peta game tersebut dan dapat berpindah dari suatu koordinat ke koordinat yang dituju saat dikendalikan dengan *controller* seperti keyboard. Karena begitu luasnya area peta yang disediakan oleh Game, jika seorang pemain harus berpindah lokasi pada jarak yang sangat jauh dengan dikendalikan secara manual, hal tersebut akan memakan banyak waktu dan mengakibatkan sebagian pemain cepat merasa bosan. Solusi yang telah hadir pada kebanyakan game RPG adalah dengan menggunakan sistem *teleport*, seorang pemain dapat langsung berpindah tempat ke lokasi yang dituju seketika, atau dengan menggunakan sistem *auto move player*, seorang pemain dapat memilih suatu lokasi yang akan dituju dan secara otomatis sistem akan menggerakkan pemain tersebut ke lokasi yang dituju dengan memilih rute tercepat tanpa perlu dikendalikan oleh pemain. Untuk memilih rute tercepat yang akan dilewati pemain pada fitur *auto move player*, terdapat beberapa algoritma yang dapat digunakan, salah satu yang cukup banyak digunakan adalah algoritma A*.



Gambar 2. Ilustrasi Pathfinding
Sumber:http://theory.stanford.edu/~amitp/GameProgramming/AStarComparison.html

Pada ilustrasi diatas, terdapat dua buah koordinat yang terdiri dari koordinat awal dan koordinat yang akan dituju, terdapat juga dua contoh dari rute yang dapat ditempuh dari

koordinat awal untuk mencapai koordinat akhir. Kedua rute tersebut memiliki total jarak ditempuh yang berbeda, dalam kasus ini rute dengan garis biru memiliki total jarak ditempuh yang lebih sedikit dibandingkan dengan rute dengan garis merah. Algoritma A* akan digunakan untuk mencari rute terpendek untuk mencapai koordinat tujuan berdasarkan rute dengan total jarak yang paling sedikit dengan mengeliminasi rute yang memiliki totak jarak yang lebih besar seperti pada kasus ilustrasi diatas, hasil diharapkan adalah rute dengan garis biru, sedangkan rute dengan garis merah melakukan pencarian dengan memilih jalur dengan titik terdekat yang langsung menuju koordinat tujuan, namun ketika hendak menuju ke koordinat tujuan, ditemukan sebuah hambatan yang menghalangi dan harus dicari jalur yang lain lagi.

II. LANDASAN TEORI

A. *Algoritma A**

Algoritma A* merupakan salah satu algoritma yang digunakan untuk memecahkan persoalan *pathfinding*. Algoritma ini dapat menemukan rute terpendek yang menghubungkan dua buah titik dengan meminimumkan total jarak yang ditempuh dari titik asal ke titik tujuan berdasarkan nilai heuristic tertentu. Algoritma A* juga merupakan perluasan dari algoritma Dijkstra dengan beberapa karakteristik dari *breadth-first search* (BFS). Dibandingkan dengan Dijkstra, algoritma A* bekerja dengan lebih efisien dalam menemukan rute yang mengubungkan titik asal dengan titik tujuan. Hal ini dikarenakan untuk setiap simpul yang akan dilalui dari titik asal untuk sampai ke titik tujuan, algoritma A* menggunakan fungsi *f(n)* yang akan memberikan estimasi dari total biaya dari suatu jalur yang melalui simpul tersebut. Karena algoritma A* menggunakan nilai heuristik sebagai estimasi, algoritma ini tidak selalu dijamin menghasilkan solusi optimal. Hanya jika dengan menggunakan pendekatan heuristik yang tepat dengan nilai *h(n)* selalu lebih kecil atau sama dengan nilai sebenarnya, solusi akan dijamin selalu menghasilkan nilai optimal.

Fungsi yang digunakan dalam menghitung estimasi biaya untuk mencapai sebuah simpul dalam suatu rute adalah sebagai berikut,

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

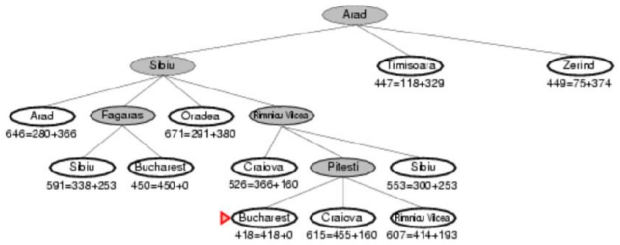
dengan,

f(n) = total estimasi dari rute yang melalui simpul n

g(n) = biaya yang diperlukan untuk mencapai simpul n dari simpul awal

h(n) = biaya estimasi yang diperlukan dari simpul n untuk sampai ke simpul akhir, menggunakan nilai heuristik.

Dalam menyelesaikan permasalahan *pathfinding* dengan menggunakan algoritma A*, dibutuhkan suatu pohon ruang status yang dibentuk berdasarkan simpul-simpul yang dilalui untuk sampai ke titik tujuan, dimulai dari titik asal sebagai simpul awal yang hidup. Terdapat istilah simpul hidup yang berarti simpul terakhir yang telah dilalui dari simpul awal dan simpul dibangkitkan (*expand node*) yang berarti simpul yang memiliki sisi dengan simpul hidup dan belum dilalui sebelumnya. Setiap simpul yang sedang dibangkitkan akan dihitung nilai *f(n)* nya dan simpul dengan nilai f(n) minimum yang akan dipilih sebagai simpul hidup pada iterasi berikutnya. Langkah ini akan terus diulangi hingga simpul hidup terakhir merupakan simpul akhir atau simpul tujuan.



*Gambar 3. Ilustrasi Pohon Ruang Status A**
Sumber: https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2020-2021/Route-Planning-Bagian2-2021.pdf/

Berikut penjelasan dari algoritma A* dalam menyelesaikan persoalan *pathfinding*

1. Pilih simpul asal sebagai simpul hidup pertama.
2. Bangkitkan semua simpul yang memiliki sisi dengan simpul hidup dan belum dilalui sebelumnya.
3. Hitung biaya *f(n)* dari semua simpul yang sedang dibangkitkan.
4. Pilih simpul dengan biaya *f(n)* terendah sebagai simpul hidup berikutnya.
5. Ulangi langkah 2 hingga ditemukan simpul hidup yang merupakan simpul akhir atau simpul tujuan ditandai dengan nilai *h(n)* = 0 dan *f(n)* = *g(n)*.

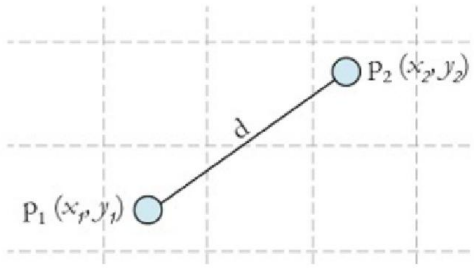
B. *Heuristik*

Penentuan nilai heuristik yang tepat untuk digunakan sangat penting dalam menghasilkan solusi yang optimal dari algoritma A*. Untuk menghasilkan solusi yang optimal, maka nilai heuristik yang dipilih haruslah lebih kecil atau sama dengan nilai yang sebenarnya. Nilai heuristik yang digunakan dalam algoritma A* untuk menentukan rute terpendek dapat berbeda-beda tergantung kasusnya, misalnya pada suatu permukaan tiga dimensi akan menghasilkan solusi yang berbeda dengan permukaan dua dimensi jika diterapkan heuristik yang sama. Pada kasus seperti bentuk aktual permukaan bumi, dapat digunakan jarak latitude dan longitude dari dua buah titik di permukaan bumi sebagai nilai heuristiknya. Pada suatu game yang mempunyai peta berupa permukaan tiga dimensi dapat digunakan vektor yang memperhitungkan bentuk permukaan, jarak, serta ketinggian dari dua buah titik, sedangkan pada kasus seperti game dengan peta berupa permukaan dua dimensi, dapat digunakan *Euclidean Distance* serta *Manhattan Distance* dari dua buah titik sebagai nilai heuristiknya. Dalam makalah ini, akan dibahas strategi algoritma A* dengan menggunakan heuristik *Euclidean Distance* serta *Manhattan Distance*.

C. *Euclidean Distance*

Euclidean Distance merupakan jarak antara dua buah titik yang dihitung berdasarkan Teorema Phytagoras dalam suatu ruang euclidean. Berikut merupakan rumus untuk menghitung jarak euclidean,

$$h = \sqrt{(x_{start} - x_{destination})^2 + (y_{start} - y_{destination})^2}$$

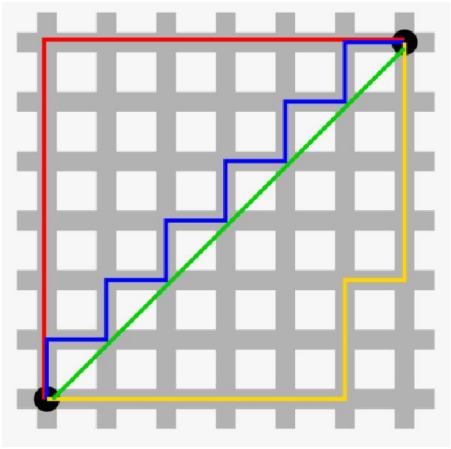


Gambar 4. Ilustrasi Euclidean Distance
Sumber: https://brilliant.org/wiki/a-star-search/

D. *Manhattan Distance*

Manhattan Distance merupakan jarak antara dua buah titik yang dihitung berdasarkan total jumlah persegi secara horizontal dan vertikal untuk mencapai titik tujuan dari titik asal. Berikut merupakan rumus untuk menghitung jarak manhattan,

$$h = |x_{start} - x_{destination}| + |y_{start} - y_{destination}|$$



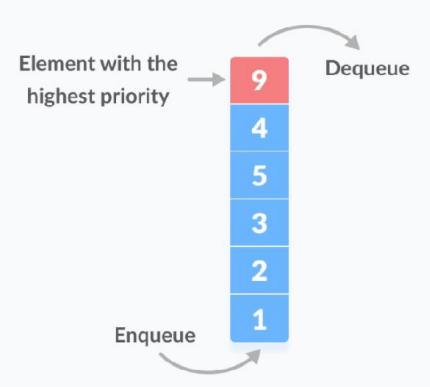
Gambar 5. Ilustrasi Manhattan Distance
Sumber: https://brilliant.org/wiki/a-star-search/

E. *Priority Queue*

Queue merupakan sebuah struktur data linear yang mengikuti prinsip *First In First Out* (FIFO) yang berarti elemen pertama yang masuk ke dalam queue akan keluar pertama kali sedangkan element terakhir yang masuk ke dalam queue akan keluar terakhir kali, misalnya pada sebuah antrian pelanggan, pelanggan yang pertama kali mengantri akan dilayani terlebih dahulu dan setelah dilayani dapat meninggalkan barisan. Terdapat istilah enqueue yang berarti menambahkan elemen pada baris belakang queue dan dequeue yang berarti mengambil elemen dari baris terdepan queue.

Priority Queue merupakan queue dengan beberapa aturan tambahan, yaitu

1. Setiap elemen memiliki nilai prioritas.
2. Element dengan prioritas lebih tinggi akan meninggalkan queue lebih dulu dibandingkan elemen dengan prioritas rendah.
3. Jika terdapat dua elemen dengan nilai prioritas yang sama, maka akan didahulukan elemen yang lebih dulu masuk ke dalam queue untuk dipanggil keluar dari queue.

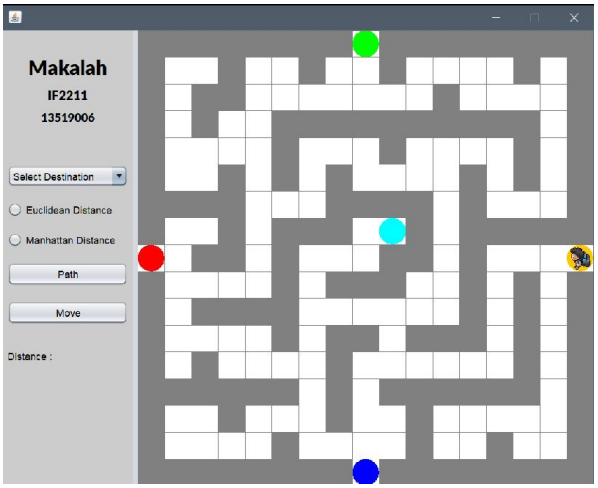


Gambar 6. Ilustrasi Priority Queue
Sumber: https://www.programiz.com/dsa/priority-queue

Dalam makalah ini, *priority queue* akan digunakan sebagai implementasi dari pohon ruang status yang dibentuk oleh algoritma A* dengan elemennya merupakan simpul-simpul pada pohon ruang status dan prioritas tertingginya diurutkan berdasarkan biaya *f(n)* terendah dari setiap simpul.

III. Hasil dan Pembahasan

Untuk memahami lebih lanjut mengenai penerapan algoritma A* dalam fitur *Auto Move Player* dalam *Role Playing Game*, penulis telah mencoba membuat program simulasi sederhana sebuah game RPG yang terdiri atas seorang player yang berada dalam peta dua dimensi.



Gambar 7. Program Simulasi Auto-Move Player

Peta tersebut terdiri atas blok-blok persegi berjumlah 17x17 yang diimplementasikan dengan menggunakan matrix. Player dapat bergerak dengan bebas di dalam peta pada posisi yang valid, hanya pada blok persegi berwarna putih secara horizontal dan vertikal, tidak dapat bergerak secara diagonal. Pada simulasi tersebut juga terdapat lima buah destinasi tujuan yang ditandai dengan lingkaran berwarna (Biru, Merah, Hijau, Oren, Cyan). Player dapat memilih dari kelima destinasi tujuan tersebut sebagai posisi tujuan yang akan dituju dari posisi awal player yang dapat berada dimana saja selama posisinya valid di dalam peta. Berikut merupakan cara kerja simulai tersebut,

1. Player dapat digerakkan dengan bebas pada posisi yang valid di dalam peta dengan menggunakan *keyboard* (w, a, s, d).
2. Posisi player saat berhenti akan ditandai sebagai posisi awal atau simpul awal.
3. Player dapat memilih destinasi tujuan pada menu di sebelah kiri peta pada bagian *dropdown* yang bertuliskan "*Select Destination*".
4. Pilih destinasi yang dituju dari kelima pilihan yang disediakan (Biru, Merah, Hijau, Oren, Cyan), destinasi tersebut akan ditandai sebagai posisi akhir atau simpul akhir.
5. Player harus memilih metode heuristik yang akan digunakan algoritma A* dalam pencarian rute, tersedia pilihan *Euclidean Distance* serta *Manhattan Distance*.
6. Ketika player menekan tombol "*Path*", akan ditampilkan rute yang akan dilalui ditandai dengan jalur berwarna kuning, posisi player masih berada di posisi awal atau simpul awal.
7. Ketika player menekan tombol "*Move*", player akan bergerak secara otomatis ke posisi simpul akhir atau destinasi dengan melalui rute terpendek.
8. Program juga dapat menampilkan total jarak yang dilalui dari simpul asal ke simpul tujuan.

Berikut merupakan algoritma A* dari program simulasi tersebut yang ditulis dalam notasi algoritmik

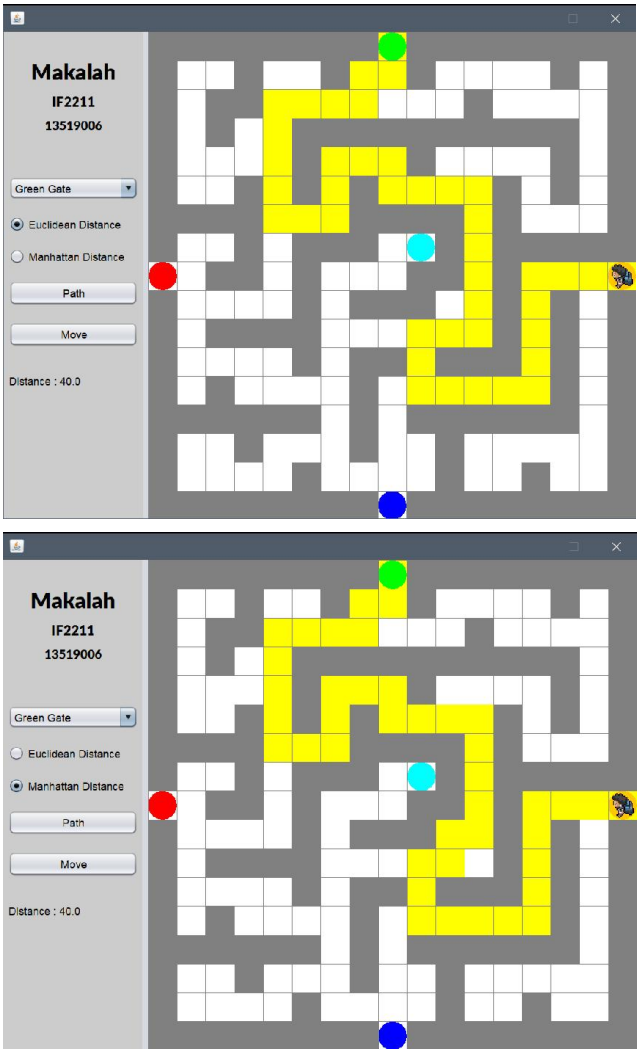
```
function AStarAlgorithm(input Psrc: Point, Pdest:
Point)-> List of Point

KAMUS

   PQ: PriorityQueue of Node

   Node: <current: Point, g: double, h: double, f:
   double, path: List of Point>

ALGORITMA

   Node.current <- Psrc

   {Hitung Node.g, Node.h, Node.f, dan masukkan ke
   Node.path}

   While(Node.h != 0 and Node.f != Node.g) do

        W <- Point(x, y-1)

        A <- Point(x-1, y)

        S <- Point(x, y+1)

        D <- Point(x+1, y)

        if (W valid and belum dikunjungi) then

            WNode.current <- W

            {Hitung WNode.g, WNode.h, WNode.f, dan
            update Node.path ke WNode.path}

            PQ <- WNode

        if (A valid and belum dikunjungi) then

            ANode.current <- A

            {Hitung ANode.g, ANode.h, ANode.f, dan
            update Node.path ke ANode.path}

            PQ <- ANode

        if (S valid and belum dikunjungi) then

            SNode.current <- S

            {Hitung SNode.g, SNode.h, SNode.f, dan
            update Node.path ke SNode.path}

            PQ <- SNode

        if (D valid and belum dikunjungi) then

            DNode.current <- D

            {Hitung DNode.g, DNode.h, DNode.f, dan
            update Node.path ke DNode.path}

            PQ <- DNode

        Node = PQ.dequeue()

   -> Node.path
```

Untuk kode program selengkapnya ditulis dalam bahasa Java dan dapat diakses di https://github.com/cjustinw/AStar-maze.

Algoritma A* pada program simulasi tersebut akan menerima input berupa dua buah titik, yaitu titik asal atau posisi awal player dan titik akhir atau posisi tujuan player, dan akan mengembalikan sebuah list dari kumpulan titik yang ketika dihubungkan akan membentuk rute dengan jarak terdekat. Rute tersebut yang kemudian akan digunakan oleh program untuk menggerakkan player.

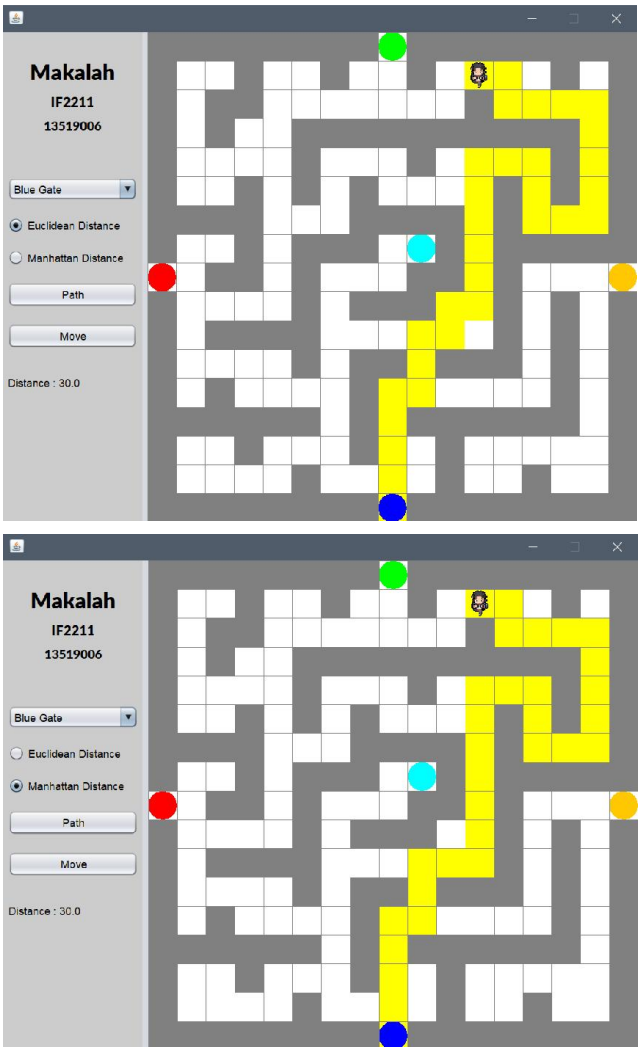
Ketika algoritma A* tersebut dijalankan, diperlukan suatu struktur data berupa *Priority Queue of Node* sebagai implementasi dari pohon ruang status dan juga struktur data *Node* sebagai implementasi simpul dari pohon ruang status tersebut, implementasi pada program yang sebenarnya ditulis dalam bahasa Java adalah dengan menggunakan kelas dan objek. Setiap Node akan mencatat informasi mengenai simpul hidup saat ini, nilai dari *f(n)*, *g(n)*, dan *h(n)*, serta *path* atau jalur yang telah dilalui untuk sampai ke simpul hidup saat ini. Untuk setiap simpul hidup, akan dibangkitkan simpul-simpul yang memiliki sisi dengan simpul hidup tersebut dan belum pernah dilewati. Pada program simulasi tersebut, kemungkinan simpul yang dapat dibangkitkan adalah blok pada peta dengan posisi (x+1, y), (x-1, y), (x, y+1), dan (x, y-1) yang valid dan belum dikunjungi dari posisi simpul hidup di dalam peta. Simpul dibangkitkan yang memiliki biaya *f(n)* atau *cost* terendahlah yang akan dipilih sebagai simpul hidup berikutnya dan begitu seterusnya hingga ditemukan simpul akhir atau posisi tujuan.

Hasil Pengujian



Gambar 8. Hasil Pengujian Pertama

Dari hasil pengujian pertama, player ingin menuju lokasi lingkaran berwarna hijau dari posisi awal berada pada lingkaran berwarna oren. Dengan menggunakan dua pendekatan heuristik yang berbeda yaitu *euclidean distance* dan *manhattan distance* didapatkan hasil kumpulan simpul yang saling dihubungkan membentuk suatu rute yang ditandai dengan lintasan berwarna kuning. Kedua pendekatan heuristik tersebut menghasilkan rute dengan total jarak yang sama yaitu sebesar 40 yang merupakan total jarak lintasan terpendek yang dapat ditempuh.



Gambar 9. Hasil Pengujian Kedua

Dari hasil pengujian kedua, player ingin menuju lokasi lingkaran biru dari posisi awal berada pada sembarang posisi di dalam peta. Kedua pendekatan heuristik tersebut menghasilkan rute dengan total jarak yang sama yaitu sebesar 30 yang merupakan total jarak lintasan terpendek yang dapat ditempuh.

IV. Kesimpulan

Algoritma A* merupakan salah satu algoritma yang dapat digunakan untuk menyelesaikan persoalan *pathfinding* dengan optimal. Algoritma ini banyak digunakan dalam pengembangan aplikasi video game, seperti pada fitur *auto-move player* dalam *role-playing* game untuk menemukan rute ditempuh yang paling optimal. Penulis telah mencoba untuk membuat simulasi mengenai fitur *auto-move player* pada game dua dimensi sederhana dengan menggunakan algoritma A* untuk menemukan rute terpendek yang akan dipilih. Dengan menggunakan pendekatan heuristik *Euclidean Distance* serta *Manhattan Distance,* program simulasi tersebut dapat menemukan rute terdekat menuju lokasi tujuan. Kedua pendekatan heuristik tersebut hanya dapat digunakan secara optimal pada game dengan peta yang memiliki permukaan dua dimensi. Untuk game yang memiliki bentuk permukaan tiga dimensi dapat digunakan pendekatan heuristik lainnya yang lebih sesuai asalkan estimasi pendekatan heuristik yang digunakan haruslah memiliki nilai yang lebih kecil atau sama dengan nilai yang sebenarnya.

V. Ucapan Terima Kasih

Penulis ingin menyampaikan puji syukur kepada Tuhan Yang Maha Esa atas berkat dan rakmat-Nya makalah ini dapat diselesaikan dengan baik dan tepat waktu. Penulis juga ingin berterima-kasih kepada dosen mata kuliah IF2211 Strategi Algoritma kelas K-01, Bapak Ir. Rila Mandala, M.Eng., Ph.D., atas bimbingannya selama mengajar K-01 pada semester genap tahun ajaran 2020/2021, serta kepada Bapak Ir. Rinaldi Munir, M.T., yang telah menyediakan website kuliah beserta dengan video kuliah online. Terakhir, penulis ingin mengucapkan terima kasih kepada semua pihak yang telah mendukung penulis.

Video Link at Youtube

https://youtu.be/7iQgu2FSFQA

Repository Github

https://github.com/cjustinw/AStar-maze

References

[1] Munir, Rinaldi. 2021. "Penentuan Rute (Route/Path Planning) Bagian2: Algoritma A*". https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2020-2021/Route-Planning-Bagian2-2021.pdf. Diakses pada tanggal 9 Mei 2021.

[2] Geeksforgeeks.org. 2021. " A* Search Algorithm". https://www.geeksforgeeks.org/a-search-algorithm/. Diakses pada tanggal 10 Mei 2021.

[3] Theory.stanford.edu. "Introduction to A*". http://theory.stanford.edu/~amitp/GameProgramming/AStarComparison.html. Diakses pada tanggal 10 Mei 2021.

[4] Brilliant.org. "A* Search". https://brilliant.org/wiki/a-star-search/. Diakses pada tanggal 10 Mei 2021

[5] Programiz.com. "Priority Queue". https://www.programiz.com/dsa/priority-queue. Diakses pada tanggal 10 Mei 2021.

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, 10 Mei 2021

Christopher Justine William
13519006