# *Penerapan Algortima A* Pada Permainan Bergenre Real-Time Strategy (RTS)*

Haikal Ardzi Shofiyyurrohman - 13521012
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jalan Ganesha 10 Bandung
E-mail: haikalardzi@gmail.com

***Abstrak*— Makalah ini membahas penerapan algoritma A* dalam permainan bergenre *Real-Time Strategy* (RTS). Permainan RTS adalah genre permainan yang mana pemain harus mengelola sumber daya, membangun markas, dan mengatur strategi untuk mengalahkan lawan dalam *real-time*. Dalam permainan RTS, keputusan yang cepat dan efisien sangat penting untuk meraih kemenangan. Algoritma A* adalah algoritma pencarian jalur yang memiliki akurasi dan peforma yang sangat efisien sehingga sering digunakan dalam aplikasi permainan. Dalam konteks permainan RTS, algoritma A* digunakan untuk mencari jalur terpendek antara unit atau pasukan pemain dengan tujuan yang ditentukan, seperti lokasi, lawan, atau objek tertentu.**

***Keywords—Algoritma A*;Permainan Real-Time Strategy (RTS); Pencarian jalur.***

## I. PENDAHULUAN

*Gambar 1.1. Permainan Real-Time Strategy.*
*Sumber: Google image search*

Permainan merupakan industri yang terus berkembang dengan berbagai genre dan konsep yang menarik. Salah satu genre yang populer adalah permainan bergenre Real-Time Strategy (RTS), yang bertujuan agar pemain mengelola sumber daya, membangun markas, dan mengatur strategi untuk mengalahkan lawan dalam waktu yang nyata (*real-time*) [1]. Dalam permainan RTS, kecepatan dan efisiensi dalam pengambilan keputusan sangat penting untuk mencapai kemenangan. Dalam upaya meningkatkan pengalaman bermain dan memperkaya strategi yang dapat diterapkan, *Developer* permainan terus mencari cara untuk mengimplementasikan algoritma cerdas dan efisien dalam mekanisme permainan. Salah satu algoritma yang telah terbukti efektif dalam banyak aplikasi permainan adalah algoritma A* (A-star) karena akurasi dan peformanya dalam mencari jalur yang paling optimal [2].

*Gambar 1.2. Hasil pathfinding pada permainan RTS.*

*Sumber: Company of Heroes 2 Screenshots*

Algoritma A* adalah algoritma pencarian jalur yang memanfaatkan teknik heuristik untuk menemukan jalur terpendek antara dua titik dalam suatu ruang [3]. Dengan menggunakan informasi heuristik, algoritma A* mampu mengurangi jumlah simpul yang harus dieksplorasi, sehingga meningkatkan efisiensi pencarian jalur. Algoritma ini telah banyak digunakan dalam berbagai konteks, termasuk dalam Permainan. Salah satu fitur utama algoritma A* adalah penggunaan heuristik. Heuristik pada *pathfinding* adalah fungsi perkiraan yang memberikan nilai perkiraan jarak atau biaya paling minimal titik tujuan dari suatu simpul tertentu [4]. Heuristik ini memberikan panduan kepada algoritma A* untuk mengarahkan pencarian ke jalur yang lebih mungkin mengarah ke solusi optimal. Dengan menggunakan nilai heuristik, algoritma A* dapat mengurangi jumlah simpul yang harus dieksplorasi, sehingga meningkatkan efisiensi pencarian jalur. Dengan menggunakan algoritma A* dalam permainan RTS, pemain dapat menghindari rintangan, mengambil keputusan yang tepat, dan meningkatkan peluang mereka untuk meraih kemenangan.

Dalam makalah ini, saya akan membahas penerapan algoritma A* pada permainan bergenre RTS. Tujuan utama kami adalah menjelaskan konsep dasar algoritma A* dan bagaimana algoritma tersebut dapat diterapkan dalam

permainan RTS untuk mencari jalur terpendek antara unit atau pasukan pemain dengan tujuan yang ditentukan. Melalui penerapan algoritma A* dalam permainan RTS, diharapkan pemain dapat mengoptimalkan strategi mereka dengan memanfaatkan pencarian jalur yang efisien. Dengan menggunakan algoritma A*, pemain dapat menghindari rintangan dan mengambil keputusan yang tepat dalam waktu yang singkat, sehingga meningkatkan peluang mereka untuk meraih kemenangan dalam permainan RTS.

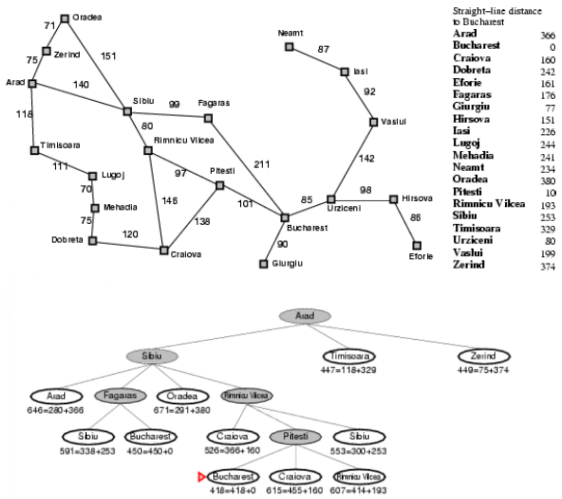
## II. TEORI DASAR

### A. *Path Finding*

*Pathfinding* (penemuan jalan) adalah proses mencari jalur terpendek antara dua lokasi dalam suatu ruang yang terdiri dari berbagai titik atau node yang terhubung oleh jalur. Tujuan dari *pathfinding* adalah menemukan jalur optimal yang meminimalkan jarak atau biaya yang ditempuh untuk mencapai lokasi tujuan. *Pathfinding* digunakan dalam berbagai bidang, termasuk permainan, robotika, navigasi, dan optimisasi rute. Terdapat beberapa algoritma *pathfinding* yang cukup populer digunakan, yaitu

1. Algoritma Dijkstra, (sesuai penemunya Edsger Dijkstra), adalah sebuah algoritma yang dipakai dalam memecahkan permasalahan jarak terpendek untuk sebuah graf berarah. Algoritma Dijkstra bekerja dengan membuat jalur ke satu simpul optimal pada setiap langkah. Jadi pada langkah ke n, setidaknya ada n node yang sudah diketahui jalur terpendek [5].
2. Algoritma A* adalah algoritma pencarian yang sederhana dan efisien yang dapat digunakan untuk menemukan jalur optimal antara dua node dalam sebuah graf. A* menggunakan graf berbobot dalam penerapannya. Graf berbobot menggunakan angka untuk mewakili jarak atau biaya pengambilan jalur [6].
3. Algoritma Bellman-Ford adalah algoritma yang dikembangkan oleh Richard Bellman and Lester Ford, Jr. Algoritma ini sangat mirip dengan Algortima Dijkstra namun algortima ini mampu menangani bobot negatif pada pencarian jalur terpendek pada sebuah graf berbobot [7].
4. Algoritma Floyd-Warshall adalah salah satu varian dari pemrograman dinamis, yaitu suatu metode yang melakukan pemecahan dengan memandang solusi yang akan diperoleh sebagai suatu keputusan yang saling terkait [8]. Algoritma Floyd-Warshall dirancang untuk menyelesaikan semua masalah jalur terpendek untuk graf dengan tepi biaya negatif [9].

### B. *Algoritma A**




*Gambar 2.1. Contoh Cara Kerja Algoritma Pathfinding A* Pada Graf Peta Romania Dari Arad ke Bucharest.*
*Sumber: https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2020-2021/Route-Planning-Bagian2-2021.pdf*

Algoritma A* (A-star) adalah algoritma pencarian jalur yang memanfaatkan teknik heuristik untuk menemukan jalur terpendek atau optimal antara dua titik dalam suatu ruang atau graf [6]. Algoritma ini digunakan secara luas dalam bidang permainan video, robotika, navigasi, dan optimisasi rute. Konsep dasar algoritma A* melibatkan pendekatan heuristik untuk memandu pencarian jalur. Algoritma ini mempertimbangkan pencarian dengan menggunakan formula berikut:

$$f(n) = g(n) + h(n) \quad (1)$$

Dengan:

1. $g(n)$ = jarak atau biaya sebenarnya yang telah ditempuh dari titik awal ke simpul n saat ini [10].
2. $h(n)$ = perkiraan jarak atau biaya dari n ke tujuan atau dapat disebut nilai heuristik [10].
3. $f(n)$ = perkiraan jarak atau biaya total dari jalur yang melalui n menuju tujuan [10].

Algoritma A* memilih simpul berikutnya untuk dieksplorasi berdasarkan nilai $f(n)$ terkecil. Pada setiap langkah, algoritma memperbarui nilai $g(n)$ dan $f(n)$ untuk simpul-simpul tetangga yang belum dieksplorasi. Proses ini berlanjut hingga simpul tujuan ditemukan atau seluruh ruang pencarian telah dieksplorasi.

Pada Gambar 2.2. berikut adalah pseudocode dari Algoritma A*:

```
function AStar(start, end)
  //init priorityQueue with start.f determine the priority
  nodePrioQ = new prioQueue(start.f)
  visited = new List()

  source.g = 0
  source.f = source.g + heuristic(source, destination)

  while nodePrioQ is not Empty
    currentNode = nodePrioQ.dequeue()
    if currentNode = destination
      return construct_path(destination) // path found
    endif
    visited.add(currentNode)
    foreach neighbour in currentNode.neighbours
      if visited not contain neighbour
        neighbour.f = neighbour.g + heuristic(neighbour, destination)
        if nodePrioQ not contain neighbour
          nodePrioQ.add(neighbour)
        else
          //update g value if node already in the queue
          existingNeighbour = nodePrioQ.get(nodePrioQ.indexAt(neighbour))
          if neighbour.g < existingNeighbour.g
            existingNeighbour.g = neighbour.g
            existingNeighbour.parent = neighbour.parent
          endif
        endif
      endif
    endfor
  endwhile
  return null
endfunction
```

*Gambar 2.2. Pseudocode Algoritma A**
*Sumber: Dokumen pribadi*

### *C. Permainan Real-Time Strategy*

Permainan *Real-Time Strategy* (RTS) merupakan sebuah genre permainan yang cukup populer. Keunikan dari RTS adalah sebuah permainan strategi dimana waktu terus berjalan untuk semua pemain, Ini membuat situasi dimana player harus menentukan strategi dalam hitungan detik [11]. Berikut adalah beberapa bagian yang terkait dengan permainan *real-time strategy*:

1. Sumber daya merupakan elemen penting dalam permainan RTS. Pemain harus mengelola dan memanfaatkan sumber daya yang tersedia, seperti mineral, kayu, atau energi, untuk membangun struktur, melatih unit, dan melakukan penelitian.
2. Pemilihan strategi yang efektif sangatlah penting dalam permainan RTS. Strategi ini dapat mencakup membangun pasukan besar dan kuat, fokus pada teknologi dan penelitian, atau mengadopsi pendekatan taktis tertentu.
3. Peta permainan RTS sering kali memiliki wilayah-wilayah yang berbeda dengan karakteristik unik. Pemain perlu memahami peta, memanfaatkan fitur alam, dan memilih posisi yang menguntungkan untuk memaksimalkan kekuatan dan pertahanan pasukan mereka.

Mekanik permainan dalam genre Real-Time Strategy (RTS) mencakup sejumlah konsep dan elemen dasar yang membentuk cara pemain berinteraksi dengan permainan. Berikut adalah penjelasan dasar tentang mekanik permainan RTS:

1. Pengelolaan Sumber Daya

RTS melibatkan pengelolaan sumber daya yang penting untuk membangun dan mengembangkan kekuatan pemain. Pemain harus mengumpulkan sumber daya seperti kayu, logam, atau energi untuk membangun struktur, melatih unit, dan melakukan penelitian teknologi baru. Mekanik ini mendorong pemain untuk membuat keputusan yang tepat dalam alokasi sumber daya, menyeimbangkan antara ekspansi dan pertahanan, serta mengoptimalkan produksi dan konsumsi sumber daya.

2. Konstruksi dan Pembangunan

Dalam RTS, pemain memiliki kemampuan untuk membangun struktur, seperti markas, bangunan produksi, pertahanan, atau pusat penelitian. Mekanik ini memungkinkan pemain untuk merancang dan mengatur basis mereka sendiri, membangun infrastruktur yang mendukung produksi dan pelatihan unit, serta mengembangkan teknologi yang lebih canggih. Kemampuan pemain dalam mengelola konstruksi dan pembangunan mempengaruhi daya tahan dan kekuatan keseluruhan mereka dalam permainan.

3. Produksi dan Pelatihan Unit

RTS melibatkan produksi dan pelatihan unit-unit tempur yang digunakan untuk menghadapi musuh atau mencapai tujuan tertentu. Pemain harus mengalokasikan sumber daya untuk memproduksi unit-unit tersebut dari bangunan produksi mereka. Mekanik ini melibatkan pengambilan keputusan dalam pemilihan unit yang tepat untuk situasi tertentu, mengatur strategi dalam penempatan unit di medan perang, serta mengelola pasokan dan pergerakan unit secara efektif.

4. Taktik dan Pertempuran

Pertempuran adalah aspek sentral dalam RTS, di mana unit-unit pemain berinteraksi dengan musuh untuk mencapai kemenangan. Mekanik ini melibatkan pemilihan taktik yang tepat, seperti pengaturan formasi unit, penggunaan serangan jarak jauh atau serangan mendekat, dan penggunaan kemampuan khusus unit. Pemain juga harus mengawasi dan mengatur pergerakan dan penempatan unit selama pertempuran untuk memaksimalkan efektivitas serangan dan pertahanan mereka.

5. Penelitian dan Perkembangan

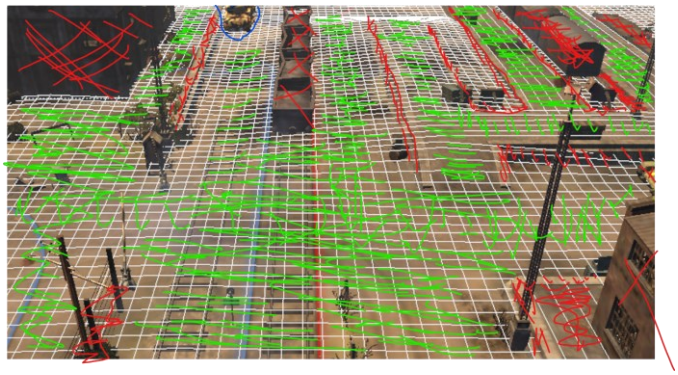
Dalam RTS, pemain memiliki kemampuan untuk melakukan penelitian dan pengembangan teknologi baru yang dapat meningkatkan kekuatan dan kemampuan mereka. Mekanik ini memungkinkan pemain untuk meningkatkan unit dan struktur mereka, membuka kemampuan baru, dan mendapatkan keunggulan strategis atas musuh. Penelitian dan perkembangan menjadi aspek penting dalam membangun kekuatan yang kompetitif dan beradaptasi dengan perubahan dalam permainan.

## III. PEMBAHASAN

*Gambar 3.1. Jalur yang dipetakan oleh algoritma.*
*Sumber: Dokumen pribadi.*

Penerapan algoritma A* dilakukan saat sebuah unit yang dipilih oleh pemain diperintah untuk berpindah dari tempat ke tujuan. Dalam hal ini sebuah unit kendaraan yang berada dilingkaran biru pada Gambar 3.1. diperintahkan oleh pemain untuk berpindah ke lokasi X yang berada dikanan atas peta, dapat dilihat di Gambar 3.1. Berikut aspek yang terlibat dalam penentuan jalur yang akan dilalui oleh unit tersebut.

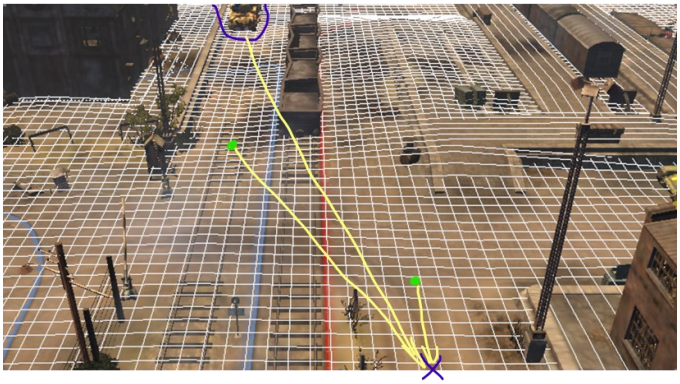
### *A. Representasi Peta*



*Gambar 3.2. Representasi peta.*
*Sumber: Dokumen pribadi*

Dalam implementasi algoritma A*, peta permainan RTS perlu direpresentasikan sebagai graf atau struktur data yang sesuai. Dalam hal ini dapat dilihat pada Gambar 3.2. seluruh peta direpresentasikan dalam bentuk *grid* setiap kotak pada peta dapat menjadi posisi atau titik yang menjadi simpul dalam graf, dan sisi-sisi menghubungkan posisi yang dapat dijangkau secara langsung oleh unit kendaraan tersebut. Bobot atau biaya pada setiap sisi dapat mencerminkan jarak dalam melintasi area tertentu. Pada gambar 3.2. tiap kotak sama dengan 1 $m^2$. Representasi peta ini memungkinkan algoritma A* untuk melakukan pencarian jalur secara efisien.

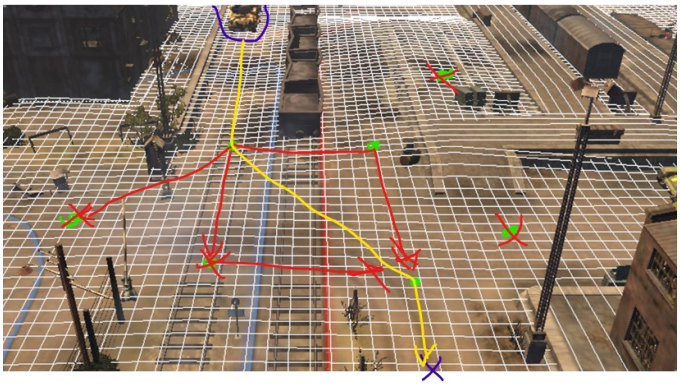
### *B. Heuristik*



*Gambar 3.3. Contoh Heuristic Dengan Euclidean Distance.*
*Sumber: Dokumen pribadi.*

Dalam implementasi algoritma A* pada game RTS, diperlukan heuristik yang relevan untuk menghitung estimasi biaya yang tersisa dari setiap simpul ke tujuan. Heuristik ini dapat berdasarkan faktor-faktor seperti *Euclidean Distance* atau *Manhattan Distance* antara simpul saat ini dan tujuan, medan yang harus dihindari, atau wilayah musuh yang mungkin menjadi ancaman. Heuristik ini membantu algoritma A* untuk fokus pada jalur yang paling menjanjikan menuju tujuan. Contoh nilai heuristic ialah pada Gambar 3.3. *Euclidean Distance* dari tank pada lingkaran biru hingga titik X yang ditarik garis kuning ialah 57.5 meter dari perhitungan 13 kotak pada *grid* sumbu X serta 56 kotak pada *grid* sumbu Y.

### *C. Pemilihan Jalur*

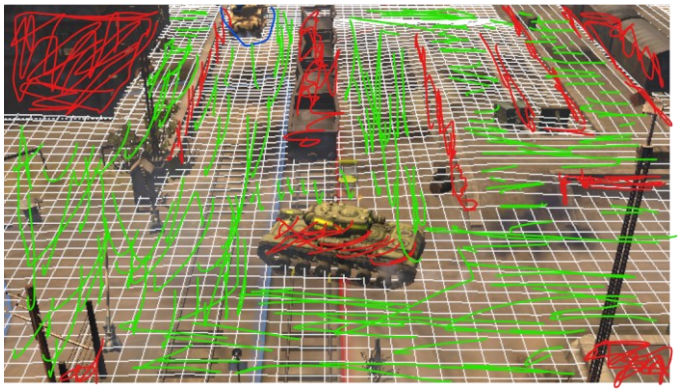


*Gambar 3.4. Pemilihan jalur Algoritma A**
*Sumber: Dokumen pribadi*

Algoritma A* diterapkan untuk mencari jalur terpendek atau optimal dari posisi awal unit atau karakter ke tujuan yang ditentukan. Pada setiap langkah, algoritma A* memilih simpul dengan biaya total terendah untuk dieksplorasi selanjutnya. Dengan memilih simpul dengan biaya total terendah, algoritma A* berupaya menemukan jalur terpendek dengan efisien. Jalur yang dihasilkan akan menjadi urutan simpul-simpul yang harus dilalui oleh unit atau karakter untuk mencapai tujuan. Contoh pada Gambar 3.4. jalur dengan representasi garis kuning dipilih oleh algoritma karena jarak total f(n)-nya paling kecil, yaitu dengan nilai 59.95 meter

dibandingkan dengan melewati jalur pada garis merah yang memiliki jarak total f(n) yang lebih besar.

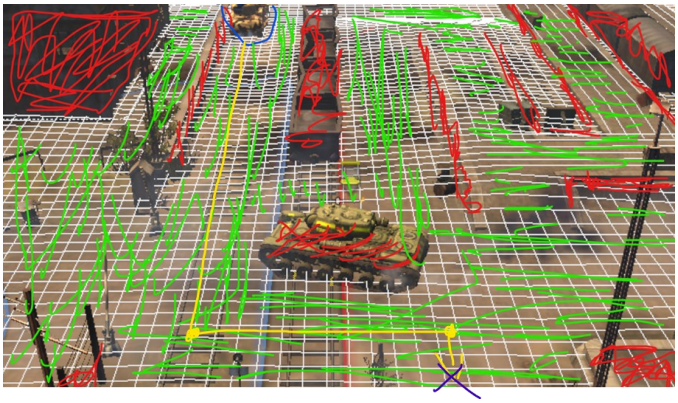
*D. Penyesuaian Peta*



*Gambar 3.5. Penyesuaian peta setelah tank berpindah tempat.*
*Sumber: Dokumen pribadi*

Implementasi algoritma A* dalam game RTS dapat melibatkan penyesuaian peta atau lingkungan permainan berdasarkan jalur yang dihasilkan. Misalnya, pemain dapat membangun struktur pertahanan atau mengatur formasi unit di sepanjang jalur yang dihitung oleh algoritma A*. Oleh karena itu, diperlukan penyesuaian setiap detiknya. Contohnya pada Gambar 3.2. peta akan disesuaikan menjadi seperti Gambar 3.5. dimana ada kendaraan yang berpindah tempat sehingga rintangan pada peta disesuaikan sehingga rintangan yang direpresentasikan dengan warna merah akan terbarui dan jalur yang dapat dilalui pun ruangannya akan berkurang dikarenakan ada rintangan yang baru muncul setelah *tank* berpindah.

*E. Pembaruan Jalur*



*Gambar 3.6. Pembaruan Jalur Setelah Penyesuaian Peta.*
*Sumber: Dokumen pribadi*

Dalam game RTS, jalur yang dihitung oleh algoritma A* perlu diperbarui secara dinamis karena kondisi permainan dapat berubah. Perubahan dalam musuh, hambatan baru, atau perubahan kondisi lingkungan dapat mempengaruhi jalur yang dipilih oleh unit atau karakter. Oleh karena itu, pembaruan jalur menjadi penting untuk memastikan bahwa unit atau karakter dapat menyesuaikan pergerakan mereka dengan situasi terkini dan menghindari hambatan baru yang mungkin muncul. Dalam pembaruan jalur, algoritma A* dapat dijalankan kembali dengan mempertimbangkan perubahan kondisi dan mencari jalur terbaru yang optimal dari posisi awal ke tujuan.

*F. Iterasi Pembaruan Jalur Pada Setiap Log/Thick.*

Dalam implementasi algoritma A* pada game RTS, pembaruan jalur biasanya dilakukan pada setiap *log* atau *thick* tertentu selama permainan berlangsung. Proses pembaruan jalur melibatkan menjalankan kembali algoritma A* untuk mencari jalur terbaru dari posisi awal ke tujuan, dengan mempertimbangkan perubahan kondisi atau situasi permainan yang terjadi sejak pembaruan jalur sebelumnya. Iterasi pembaruan jalur dilakukan secara berulang dalam interval tertentu, seperti setiap *log* atau *thick* yang diatur dalam permainan. Selama setiap iterasi, algoritma A* dieksekusi untuk mencari jalur terbaru, mengulang kembali dari poin B hingga poin E secara terus-menerus.

Dengan melakukan pembaruan jalur secara teratur, unit atau karakter dalam permainan RTS dapat menyesuaikan pergerakan mereka dengan perubahan kondisi permainan. Hal ini memungkinkan mereka untuk menghindari hambatan yang baru muncul, menavigasi melalui jalur yang optimal berdasarkan situasi saat ini, dan menjalankan strategi yang efektif dalam menghadapi perubahan yang terjadi dalam permainan. Pembaruan jalur dalam iterasi log atau thick yang teratur memastikan bahwa unit atau karakter dapat tetap bergerak secara efisien dan mengikuti jalur terpendek menuju tujuan dengan memperhitungkan kondisi permainan yang berubah. Dengan demikian, pemain dapat menjaga kecerdasan buatan dalam permainan RTS agar responsif terhadap perubahan dan tetap efektif dalam mencapai tujuan mereka.

## IV. PENUTUP

*A. Kesimpulan*

Penerapan algoritma A* pada permainan bergenre Real-Time Strategy (RTS) memiliki dampak yang signifikan dalam meningkatkan kompetisi serta aspek realistis dalam permainan. Berdasarkan penjelasan yang telah disampaikan, dapat disimpulkan bahwa:

1. Representasi peta permainan sebagai graf memungkinkan algoritma A* untuk melakukan pencarian jalur dengan lebih efisien.

2. Penggunaan nilai heuristik dalam algoritma A* membantu dalam menghitung estimasi biaya yang tersisa dari setiap simpul ke tujuan, sehingga memandu algoritma untuk memilih jalur yang paling menjanjikan.

3. Penerapan algoritma A* dalam game RTS memungkinkan pemain untuk mengoptimalkan strategi mereka dengan memanfaatkan jalur terpendek menuju tujuan, menghindari hambatan, dan memaksimalkan efektivitas pergerakan unit.

4. Pembaruan jalur secara dinamis menggunakan algoritma A* memungkinkan unit atau karakter untuk menyesuaikan pergerakan mereka dengan perubahan kondisi permainan, sehingga aspek realistis menjadi lebih dirasakan oleh para pemain.

Selebihnya, Penerapan algoritma A* pada permainan RTS membuka peluang untuk pengembangan *Artificial Intelligence* yang lebih canggih dan strategi yang lebih adaptif. Dengan menggunakan algoritma ini, pemain dapat menghadapi tantangan dalam permainan dengan lebih efisien, mengoptimalkan penggunaan sumber daya, serta merancang strategi yang lebih baik untuk mencapai tujuan dalam permainan RTS.

## UCAPAN TERIMA KASIH

Puji syukur penulis panjatkan kepada Allah Swt. sehingga makalah ini dapat diselesaikan. Tidak lupa kepada orang tua penulis yang telah mendukung proses pendidikan penulis sehingga makalah ini dapat disusun. Juga kepada Pak Ir. Rila Mandala, M. Eng., Ph.D. sebagai dosen pengampu IF2211 Strategi Algoritma yang telah membimbing penulis pada mata kuliah Strategi Algoritma serta kepada seluruh kolega yang telah mendukung pula pembuatan makalah ini.

## REFERENSI

[1] M. Rouse, "What Does Real-Time Strategy Mean?," Techopedia, 21 April 2015. [Online]. Available: https://www.techopedia.com/definition/1923/real-time-strategy-rts. [Accessed 21 Mei 2023].

[2] A. Rafiq, T. A. A. Kadir and S. N. Ihsan, "Pathfinding Algorithms in Game Development," IOP, Boston, 2020.

[3] B. Roy, "A-Star (A*) Search Algorithm," Medium, 29 September 2019. [Online]. Available: https://towardsdatascience.com/a-star-a-search-algorithm-eb495fb156bb#:~:text=A%2Dstar%20(also%20referred%20to,s%20to%20find%20the%20solution.. [Accessed 21 Mei 2023].

[4] A. Patel, "Heuristics: From Amit's Thoughts on Pathfinding," Stanford University, 20 Februari 2023. [Online]. Available: http://theory.stanford.edu/~amitp/GameProgramming/Heuristics.html. [Accessed 21 Mei 2023].

[5] A. S. Girsang, "Algoritma Dijkstra," Binus University, 28 November 2017. [Online]. Available: https://mti.binus.ac.id/2017/11/28/algoritma-dijkstra/. [Accessed 22 Mei 2023].

[6] R. A. S., "Simplilearn - Online Certification Training Course Provider," 24 April 2023. [Online]. Available: https://www.simplilearn.com/tutorials/artificial-intelligence-tutorial/a-star-algorithm#:~:text=MLExplore%20Program-,What%20is%20an%20A*%20Algorithm%3F,shortest%20path%20to%20be%20taken.. [Accessed 2023 Mei 21].

[7] D. J. Bawole and H. P. Chernovita, "Algoritma Bellman-Ford untuk Menentukan Jalur Terpendek dalam Survey Klaim Asuransi," INOBIS: Jurnal Inovasi Bisnis dan Manajemen Indonesia, vol. 03, no. 01, p. 41, 2019.

[8] F. W. Ningrum and T. Andrasto, "Penerapan Algoritma Floyd-Warshall dalam Menentukan Rute Terpendek pada Pemodelan Jaringan Pariwisata di Kota Semarang," Jurnal Teknik Elektro, vol. 8, no. 1, pp. 21-24, 2016.

[9] D. S. Hochbaum, "Lecture Notes for IEOR 266: Graph Algorithms.

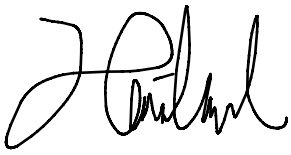
[10] R. Munir, Penentuan Rute (Route/Path Planning) Bagian 2: Algoritma A*, Bandung, 2021.

[11] O. Ener, G. S. Bhudi and Liliana, "Game Real-Time Strategy dengan menggunakan Artificial Intelligence Quantified Judgement Model dan Backpropagation Neural Network," [Online]. Available: https://media.neliti.com/media/publications/106197-ID-none.pdf. [Accessed 22 Mei 2023].

## 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, 22 Mei 2023



Haikal Ardzi Shofiyyurrohman
13521012