JACIS : Journal Automation Computer Information System Volume: **4**, Nomor **2**, **2024**, *Hal*: 136 - 145

Optimasi Performa Game Dugeon Escape Menggunakan Algoritma A-Star

Sigit Pramono [1], Ike Verawati *[2]

[1] Universitas Amikom Yogyakarta; sigitcold007@gmail.com
[2] Universitas Amikom Yogyakarta; ikeverawati@amikom.ac.id

Abstrak: Pertumbuhan dunia game semakin cepat seiring dengan perkembangan teknologi saat ini telah membawa perubahan besar dalam dunia game. Salah satu aspek yang membuat game semakin menarik adalah penerapan kecerdasan buatan (AI). Biasanya, AI diterapkan pada elemen dalam game, baik pada karakter pemain maupun NPC (Non-Playable Character). Implementasi AI pada NPC memiliki berbagai tujuan, seperti menciptakan pengalaman bermain yang lebih dinamis dan menarik. Salah satu metode AI yang sering digunakan adalah *Pathfinding*, yaitu teknik untuk menemukan jalur yang efisien. Dalam game, metode ini memungkinkan NPC untuk mencari jalur, mengejar, atau menghalangi pemain dalam mencapai tujuan tertentu. Untuk menerapkan metode *Pathfinding*, diperlukan algoritma yang sesuai. Salah satu algoritma yang sering digunakan adalah algoritma A-Star (A*). Algoritma ini dikenal efektif untuk menemukan jalur terpendek dalam berbagai situasi, termasuk dalam lingkungan permainan. Berdasarkan analisis, algoritma A-Star dapat digunakan sebagai solusi optimal untuk navigasi NPC dalam game. Penelitian ini menghasilkan sebuah game bergenre *roguelike*, yang memiliki ciri khas seperti generasi prosedural, tantangan tinggi, dan *gameplay* yang berulang. Game ini dikembangkan menggunakan Unity dan dirancang untuk dijalankan pada perangkat desktop. Penelitian ini mengoptimalkan AI pathfinding pada NPC untuk meningkatkan performa game, menjadikannya lebih responsif dan efisien.
Keywords: pathfinding; algoritma A-star; unity

DOI: https://doi.org/10.47134/jacis
*Correspondensi: Ike Verawati
Email: ikeverawati@amikom.ac.id

Receive: 26 November 2024
Accepted: 29 November 2024
Published: 30 November 2024

Abstrak: The growth of the gaming world is accelerating along with current technological advancements. A game becomes more engaging with the presence of artificial intelligence (AI). The application of AI is typically aimed at game objects, whether players or NPCs. The implementation of AI in NPCs is highly diverse to make the game more interesting. One AI method in games is Pathfinding, which is a method for finding efficient route solutions. The Pathfinding method is used for NPCs to find routes, chase, or block players from reaching objectives in the game. To implement the Pathfinding method, an algorithm is required. One such Pathfinding algorithm is the A-Star algorithm. Based on observations, the A-Star algorithm can be applied as a process for finding the shortest route in a game. Therefore, in this research, a game with the roguelike genre was developed, which includes elements such as procedural generation, repetitive gameplay, and high challenges. This game was built using Unity and can be run on a desktop. The goal of creating this game is to implement AI in enemy NPCs to achieve more naturalbehavior by effectively chasing the player.
Keywords: pathfinding; a-star algorithm; unity

PENDAHULUAN

Perkembangan industri game modern mencakup berbagai genre yang melibatkan berbagai usia, seperti FPS, Fighting, Action, RPG, Adventure, dan Simulation [1]. Di dalam game, Non-Playable Character (NPC) berfungsi sebagai agen otonom yang mewakili karakter dan dapat berinteraksi dengan pemain. Berbeda dengan karakter yang dikendalikan pemain, NPC bergerak dan bertindak secara mandiri [2]. NPC berinteraksi dengan objek di dunia game dengan cara yang sulit diprediksi, dan penerapan AI pada NPC memungkinkan mereka untuk bertindak sesuai dengan peran mereka dalam permainan [3].

Saat ini, penelitian semakin menyoroti pentingnya teknologi pathfinding dalam pengembangan game, khususnya dalam meningkatkan kecerdasan NPC untuk mencari rute terpendek antara dua titik dalam permainan. Salah satu algoritma yang banyak diterapkan dalam konteks ini adalah algoritma A-Star, yang dikenal karena kemampuannya dalam menemukan jalur terpendek di lingkungan yang kompleks [4][5]. Namun, implementasi Algoritma A-Star pada NPC sebagai musuh dapat memerlukan banyak memori, terutama dalam situasi di mana musuh bergerombol, yang berisiko menyebabkan penurunan performa dan lag pada game [6]. Untuk mengatasi masalah ini, solusi yang diusulkan adalah dengan mendistribusikan memori secara lebih efisien. Dengan cara ini, meskipun ada pengorbanan dalam akurasi navigasi NPC, game dapat tetap berjalan dengan lancar dan responsif. Pendekatan ini diharapkan dapat menjaga pengalaman bermain tetap menyenangkan dalam situasi permainan yang dinamis dan kompleks [7].

Penelitian terdahulu menunjukkan bahwa implementasi Algoritma A-Star pada NPC dalam game mampu mengoptimalkan pencarian jalur dengan efisiensi yang signifikan, bahkan dalam kondisi grid yang kompleks. Dimensi baru dengan memperluas kemampuan NPC menggunakan jalur alternatif saat menghadapi penghalang dalam game [8]. Penggunaan Algoritma A-Star dengan teknologi navmesh pada game edukasi mencapai tingkat akurasi yang memuaskan dalam simulasi pengujian [9].

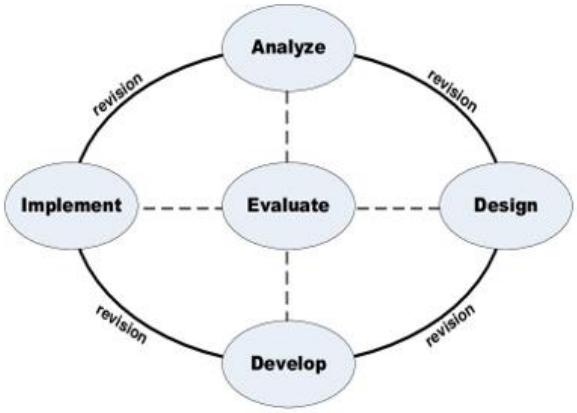
Mengeksplorasi gabungan Algoritma A-Star dengan FSM untuk meningkatkan kecerdasan NPC dalam memilih dan menyesuaikan rute berdasarkan aksi pemain, mencatat peningkatan signifikan dalam efisiensi navigasi NPC dalam game [10]. Implementasi A-star menunjukkan kemampuan Algoritma A-Star dalam menyesuaikan diri secara real-time terhadap perubahan peta dan kondisi permainan, meningkatkan interaktivitas dalam game Rogue-Like [11]. Aplikasi praktis Algoritma A-Star dalam menciptakan pengalaman bermain yang dinamis dengan navigasi NPC dalam labirin permainan [12].

Penelitian ini bertujuan untuk membahas kinerja NPC dan beban kerjanya dalam game "Dungeon Escape" dengan menerapkan Algoritma A-Star sebagai sistem pathfinding. Fokus utama penelitian adalah mengoptimalkan kinerja NPC sehingga pergerakan musuh dalam game terlihat lebih alami dan realistis, sekaligus menjaga efisiensi beban kerja sistem. Sistem pathfinding yang dirancang bertujuan untuk mengurangi penggunaan memori tanpa mengorbankan performa game, terutama saat melibatkan banyak NPC musuh secara bersamaan. Penelitian ini juga akan mengevaluasi dampak

penerapan Algoritma A-Star yang telah dioptimalkan terhadap efisiensi memori, kelancaran jalannya permainan, dan responsivitas sistem terhadap aksi pemain

METODE

Pada penelitian ini melalui beberapa langkah yang perlu dilakukan meliputi berupa pengumpulan data, analisis kebutuhan, perancangan game, pembuatan asset, produksi dan testing



Gambar 1. Alur Penelitian

a. Analyze
   Mengidentifikasi kebutuhan dan persyaratan game, menganalisis audiens target, tujuan pembelajaran, dan spesifikasi teknis.

b. Design
   Merancang gameplay, alur cerita, karakter, level, dan antarmuka pengguna. Membuat storyboard, prototipe awal, dan menyusun strategi pengujian.

c. Development
   Membangun game berdasarkan desain yang telah dibuat, termasuk pembuatan aset, pengkodean, dan pengujian internal untuk memastikan fungsi yang baik.

d. Implement
   Pada tahapan ini melakukan melakukan pembuatan game secara keseluruhan dengan penerapan algoritma A-star.

e. Evaluate
   Pengujian algoritma dilakukan untuk mengevaluasi kinerja performa game [13]

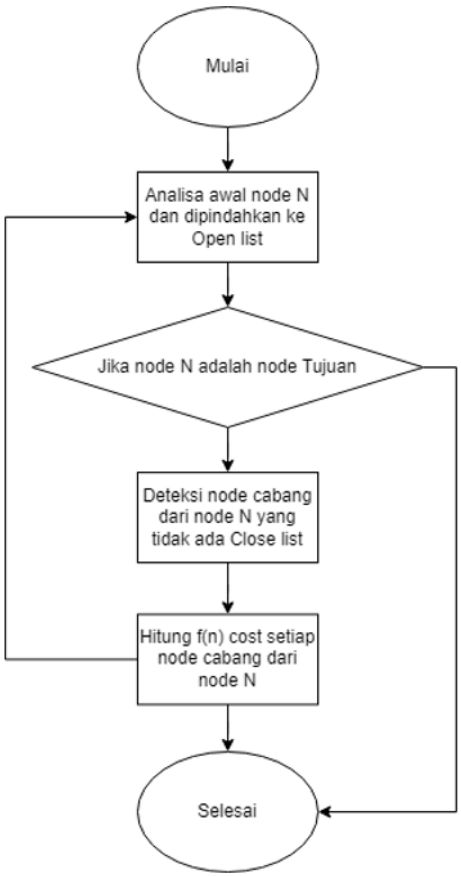
Algoritma A-Star (A*) pertama kali diperkenalkan oleh Hart, Nilson, dan Raphael pada tahun 1968. Algoritma ini termasuk dalam kategori *Branch & Bound* dan memanfaatkan informasi tambahan berupa nilai heuristik untuk menemukan solusi yang optimal. A-Star sering digunakan sebagai algoritma pencarian jalur tercepat (*Pathfinding*) melalui titik-titik yang telah ditentukan [14][15]. Metode ini menggabungkan pendekatan dari algoritma Dijkstra dengan *Greedy Best First Search* melalui modifikasi fungsi heuristiknya. Dalam prosesnya, A-Star mengombinasikan biaya jarak $g(n)$ dengan estimasi jarak ke tujuan $h(n)$,

menghasilkan solusi yang efisien. Meskipun tidak selalu menjamin jalur tercepat, algoritma ini memiliki keunggulan dalam kecepatan pencarian dan akurasi hasil [16].
Persamaan dari algoritma a-star ditunjukkan pada persmaan (1)

$$f(n) = g(n) + h(n) \quad \text{... (1)}$$

Keterangan:

h(n) = Estimasi biaya dari node n menuju tujuan.
g(n) = Biaya perjalanan yang telah ditempuh hingga node n.
f(n) = Perkiraan total biaya termurah untuk mencapai tujuan melalui node n [17].



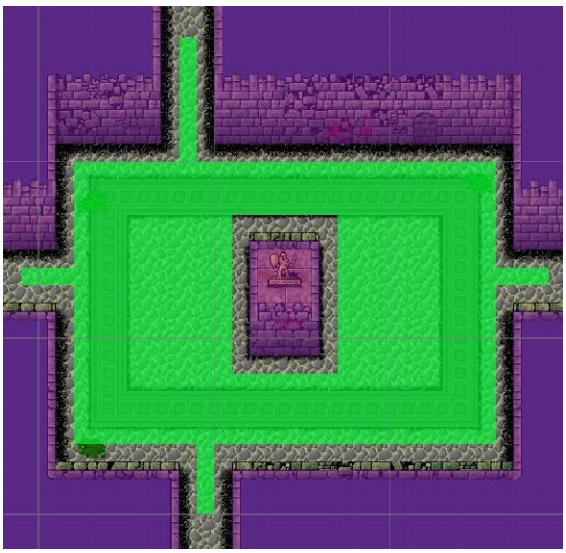
Gambar 2. Alur Algoritma A-Star

Pada gambar 2 merupakan alur dari algoritma a-star, berikut tahapan yang dilakukan pada proses algoritma a-amistar sedang berjalan:

1. Inisialisasi node awal n dan menambahkannya ke dalam daftar open list
2. Perulangan
   a. Cari node n dengan nilai f(n) terkecil dalam open list. Pindahkan node nnn ke closed list dan simpan.
   b. Identifikasi semua node cabang dari nnn yang belum berada di closed list dan tambahkan ke open list.
   c. Hitung nilai f(n) untuk setiap node yang baru ditambahkan ke open list.
   d. Hentikan perulangan jika node n adalah node tujuan.

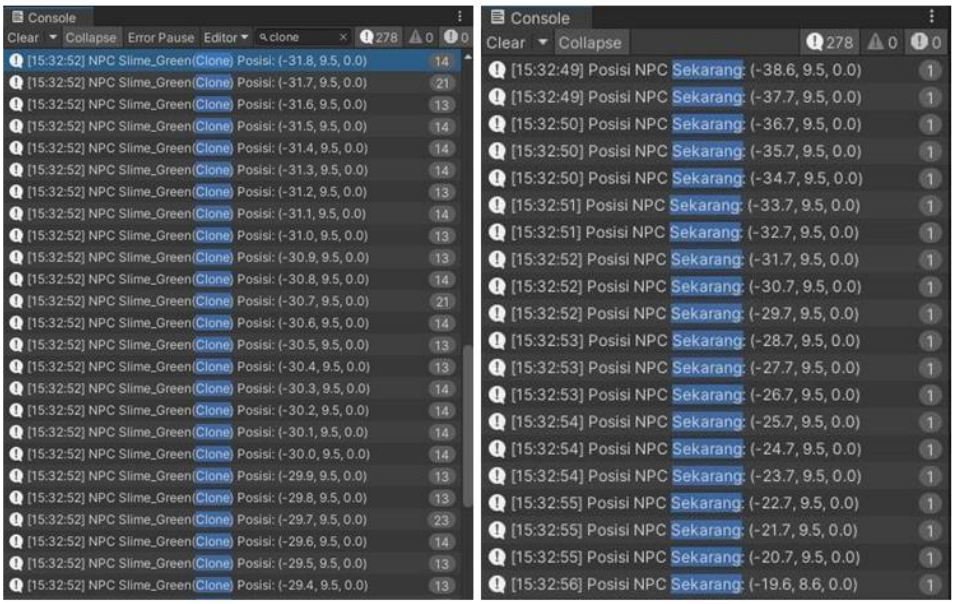
HASIL DAN PEMBAHASAN

Pada pengujian simulasi algoritma, dilakukan verifikasi untuk memastikan bahwa algoritma A-Star yang diterapkan pada game berfungsi sebagaimana mestinya.



Gambar 3. Rute NPC

Pada Gambar diatas rute NPC yang dapat dilalui ditandai dengan warna hijau. Pencarian rute menggunakan algoritma A-Star pada unity didapatkan dari console pada unity, maka akan mendapatkan data seperti gambar 4.



Gambar 4. Console log

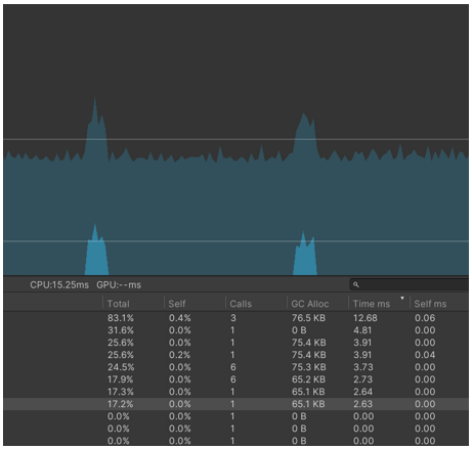
Berdasarkan Gambar 4 terdapat 2 console log yang dipakai, pada gambar sebelah kiri posisi NPC direkam setiap detik, sedangkan yang sebelah kanan posisi NPC yang direkam setiap dia bergerak dari satu node ke node lainnya. Ini berfungsi untuk mendapatkan nilai Node Dilewati, Node Duplikat, dan Waktu Tempuh. Yang nantinya akan digunakan untuk menghitung waktu rata-rata seperti yang ditunjukkan pada tabel 1.

Tabel 1. Data yang didapatkan berasal dari console log

Level	Node Dilewati	Waktu Tengah	Node Nuplikasi
1	22	00:08	23
2	21	00:10	23
3	25	00:10	26

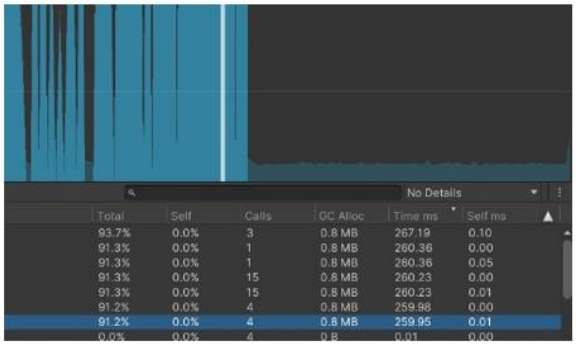
Berdasarkan tabel 1maka kita dapat menghitung waktu rata-rata yang akan diperlukan oleh NPC untuk mengejar player dan juga nilai akurasi algoritma A-Star. Berikut perhitungan dari pengujian.

1. Nilai rata-rata waktu tempuh Average Time = (22: 8 + 21: 10 + 25: 10): 3
   = (2, 75 + 2, 1 + 2, 5): 3
   = (7, 35): 3
   = 2, 45
2. Nilai dari akurasi algoritma A-Star
   Dari table 1, terdapat 23 node duplikat di level 1, 23 di level 2, dan 26 di level 3. Duplikasi ini terjadi karena sistem membaca node secara berulang. Dengan demikian, akurasi algoritma A-Star dapat disimpulkan sebagai berikut:
   Level 1 akurasi =
   $$\frac{22}{23} \; x \; 100\% = 95{,}65\%$$
   Level 2 akurasi =
   $$\frac{21}{23} \; x \; 100\% = 91{,}30\%$$
   Level 3 akurasi =
   $$\frac{24}{26} \; x \; 100\% = 92{,}30\%$$
   Sehingga didapatkan rata-rata akurasi keseluruhan adalah 93,08% yang menunjukkan performa algoritma A-Star yang cukup baik dalam kondisi yang diujikan.
3. Kinerja Game
   Setelah menghitung node dan akurasi, dilakukan evaluasi kinerja game untuk menentukan apakah FPS sudah memenuhi standar kelayakan permain.



Gambar 5. Grafik performa games

Saat memasuki ruangan dan menemukan satu musuh bisa dilihat latensi dari algoritma A-Star memiliki latensi yang rendah.



Gambar 6. Grafik performa saat banyak musuh

Ketika musuh bergerombol, latensi algoritma A-Star meningkat hingga 259,95 ms, menyebabkan penurunan kinerja algoritma. Hal ini berdampak pada penurunan FPS game, sehingga game menjadi tidak layak untuk dimainkan.

Implementasi Metode Distribusi

Untuk mengatasi masalah ini, diterapkan metode distribusi perhitungan. Metode ini mendistribusikan tugas perhitungan jalur (pathfinding) secara paralel ke beberapa core prosesor.

1. Kondisi Distribusi Frame
   Disini memastikan bahwa pathfinding hanya dilakukan pada frame tertentu dengan modulus operator.

```
if (Time.frameCount %
Settings.targetFrameRateToSpreadPathfindingOver
!= updateFrameNumber) return;
```

Gambar 7. Script distribusi frame

   Kode ini memastikan bahwa proses pathfinding tidak dilakukan di setiap frame, melainkan hanya pada frame tertentu berdasarkan pengaturan yang ada

2. Motode untuk mengatur frame update
   Metode ini mengatur frame dimana pathfinding harus dilakukan.

```
public void SetUpdateFrameNumber(int
updateFrameNumber)
{
  this.updateFrameNumber = updateFrameNumber;
}
```

Gambar 8. Script mengatur frame update

   Implementasi kode ini menunjukkan pendekatan yang efisien untuk mengelola sumber daya komputasi dalam pengembangan aplikasi real-time

3. Pengaturan frame update untuk musuh
   Metode ini memastikan setiap musuh memiliki frame update yang berbeda untuk pathfinding.

```
private void SetEnemyMovementUpdateFrame(int enemySpawnNumber)
{
  enemyMovementAI.SetUpdateFrameNumber(enemySpawnNumber %
Settings.targetFrameRateToSpreadPathfindingOver);
}
```

Gambar 9. Script frame update pathfinding

Gambar 9 menunjukkan kode yang dirancang untuk mendistribusikan perhitungan pathfinding musuh secara merata ke beberapa frame.

Berdasarkan hasil pahtfinding yang dilakukan diperoleh data dari console log seperti tabel 2.

Table 2. Data yang didapat dari console log

Level	Node Dilewati	Waktu Tengah	Node Nuplikasi
1	31	00:13	42
2	31	00:14	33
3	24	00:10	27

Tabel 2 berisi data yang didapat setelah menguji ulang NPC setelah menerapkan metode distribusi frame. Setelah menerapkan metode distribusi, ada perbedaan data yang dimana dapat kita lakukanuntuk menghitung rata-rata yang akan diperlukan oleh NPC untuk mengejar player dan juga nilai akurasi algoritma A-Star. Berikut perhitungan dari pengujian.

1. Nilai rata-rata waktu tempuh

   Average time = (31:13 + 31:14 + 24:10) : 3
   = (2,38 + 2,21 + 2,4) : 3
   = (6,99) : 3
   = 2,33

2. Nilai dari akurasi algoritma A-Star

   Dari table 1, terdapat 42 node duplikat di level 1, 33 di level 2, dan 27 di level 3. Duplikasi ini terjadi karena sistem membaca node secara berulang. Dengan demikian, akurasi algoritma A-Star dapat disimpulkan sebagai berikut:

   Level 1 : $Akurasi = (31/42)$ x $100\% = 73{,}80\%$

   Level 2 : $Akurasi = (31/33)$ x $100\% = 93{,}93\%$

   Level 3: $Akurasi$ =-$(24/27)$ x $100\% = 85{,}53\%$

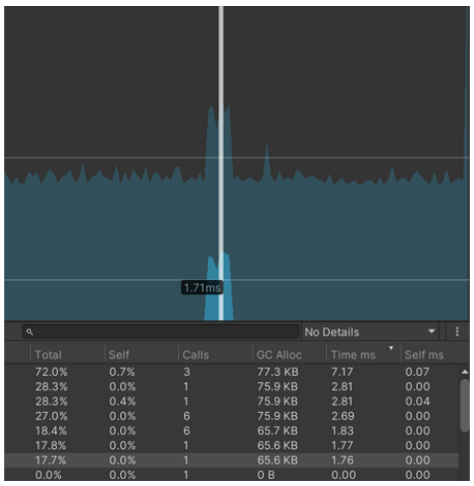
   Berdasarkan perhitungan diatas, dapat disimpulkan setelah melakukan solusi untuk mendistribusikan beban ke tiap frame, bahwa penerapan algoritma pada game ini untuk NPC musuh menggunakan algortima A-Star menghasilkan rata-rata waktu 2,33 node per detik dari titik awal ke titik akhir. Sedangkan tingkat akurasi algoritma A-Star di Unity menjadi 85,53%.

3. Kinerja games

   Selesai menghitung node dan akurasi, dilakukan evaluasi kinerja game untuk menentukan apakah FPS sudah memenuhi standar kelayakan permainan seperti Gambar 10. Setelah menerapkan metode distribusi beban kinerja, performa sistem mengalami peningkatan yang signifikan, dengan nilai latensi yang tercatat sebesar 1,76 ms. Ketika memasuki ruangan dan menghadapi satu musuh, dapat diamati bahwa algoritma A-Star menunjukkan latensi yang rendah.

Table 3. Table perbandingan

	Tanpa Distribusi	Dengan Distribusi
Node per-detik	4,25	2,33
Akurasi	93,08%	85,53%
Latensi NPC	259,95ms	1,76ms



Gambar 10. Grafik performa game

SIMPULAN

Berdasarkan hasil yang diperoleh tanpa menggunakan metode distribusi beban kerja, didapatkan kinerja sebesar 4,25 node per detik, akurasi 93,08%, dan latensi pergerakan NPC sebesar 259,95 ms. Setelah menerapkan metode distribusi beban kerja, diperoleh kinerja sebesar 2,33 node per detik, akurasi 85,53%, dan latensi pergerakan NPC sebesar 1,76 ms. Rata-rata node per detik turun menjadi 2,33, dan tingkat akurasi menurun menjadi 85,53%. Namun, latensi pergerakan NPC berkurang secara signifikan menjadi 1,76 ms. Dengan kata lain, meskipun metode distribusi ini berhasil mengurangi latensi pergerakan NPC dengan sangat baik, terdapat penurunan dalam akurasi dan kinerja pathfinding yang perlu diperhatikan.

DAFTAR PUSTAKA

[1] A. Mulachela, K. Rizki, and Y. A. Wahyudin, "Analisis Perkembangan Industri Gamedi Indonesia Melalui Pendekatan Rantai Nilai Global (Global Value Chain)."

[2] C. Reynolds and C. W. Reynolds, "Steering Behaviors For Autonomous Characters." [Online]. Available: http://www.red.com/cwr/cwr@red.com

[3] M. Faris *et al.*, "Penerapan Finite State Machine dalam Perancangan Perilaku Musuh pada Game 2D Platformer bertema Sejarah," vol. 14, pp. 236–243, 2024, doi: 10.36350/jbs.v14i2.

[4] A. Candra, M. A. Budiman, and R. I. Pohan, "Application of A-Star Algorithm on Pathfinding Game," in *Journal of Physics: Conference Series*, IOP Publishing Ltd, Jun. 2021. doi: 10.1088/1742-6596/1898/1/012047.

[5] A. A. Maliq, Y. Darmi,) ; Dwita Deslianti, and A. R. Walad Mahfuzi, "Implementation Of The A* (A-Star) Algorithm For Website-Based Search Of Tourism Sites In The City Of Bengkulu Implementasi Algoritma a* (A-Star) Pada Pencarian Lokasi Pariwisata Di Kota Bengkulu," *JURNAL KOMITEK*, vol. 3, no. 2, p. page, 2023, doi:

10.53697/jkomitek.v3i2.

[6] S. L. Pardede, F. R. Athallah, Y. N. Huda, and F. D. Zain, "A Review of Pathfinding in Game Development," *[CEPAT] Journal of Computer Engineering: Progress, Application and Technology*, vol. 1, no. 01, p. 47, May 2022, doi: 10.25124/cepat.v1i01.4863.

[7] G. Koulaxidis and S. Xinogalos, "Improving Mobile Game Performance with Basic Optimization Techniques in Unity," *Modelling*, vol. 3, no. 2, pp. 201–223, Jun. 2022, doi: 10.3390/modelling3020014.

[8] G. Mutaqin, J. N. Fadilah, and F. Nugroho, "Implementasi Metode Path Finding dengan Penerapan Algoritma A-Star untuk Mencari Jalur Terpendek pada Game 'Jumrah Launch Story,'" *Walisongo Journal of Information Technology*, vol. 3, no. 1, pp. 43–48, Jun. 2021, doi: 10.21580/wjit.2021.3.1.7042.

[9] P. Harsadi and S. Siswanti, "Penerapan Pathfinding Menggunakan Algoritma A* Pada Non Player Character (NPC) Di Game," *Jurnal Ilmiah SINUS*, vol. 17, no. 2, p. 39, Jul. 2019, doi: 10.30646/sinus.v17i2.423.

[10] E. Reginald, C. San, and E. Handriyantini, "Penerapan Metode Pathfinding Pada Pengembangan Game 'The Book of Aksara' Pada Perangkat Bergerak."

[11] E. Gumelar, A. #1, D. E. #2, and A. Fauzi, "Implementasi Metode Pathfinding dengan Algoritma A* pada Game Rogue-like menggunakan Unity", doi: 10.34818/indojc.2022.7.3.677.

[12] D. Ghifari Auliansyah, R. Erfa Saputra, and R. Astuti Nugrahaeni, "Penerapan Algoritma A* Sebagai Sistem Pencarian Rute Pada NPC Game Labirin."

[13] T. Wibowo, "Perancangan Game RPG untuk Pembelajaran Bahasa Inggris dengan Metode ADDIE An RPG Game Design for English Learning using ADDIE Methods".

[14] W. Bismi, W. Gata, T. Asra, I. Komputer, and U. Nusa Mandiri, "Penerapan Algoritma Hybrid Dalam Menentukan Rute Terpendek Antara Cabang Kampus," *Ultima Computing : Jurnal Sistem Komputer*, vol. 13, no. 1, 2021.

[15] L. Safira, P. Harsadi, and S. Harjanto, "Penerapan Navmesh Dengan Algoritma A Star Pathfinding Pada Game Edukasi 3d Go Green," *Jurnal Teknologi Informasi dan Komunikasi (TIKomSiN)*, vol. 9, no. 1, p. 17, Apr. 2021, doi: 10.30646/tikomsin.v9i1.540.

[16] A. C. Prasetyo, M. Prayoga Arnandi, H. S. Hudnanto, and B. Setiaji, "Perbandingan Algoritma Astar dan Dijkstra Dalam Menentukan Rute Terdekat 36 Jurnal Ilmiah SISFOTENIKAJuly201xIJCCS Perbandingan Algoritma Astar dan Dijkistra Dalam Menentukan Rute Terdekat Astar and Dijkistra Algorithm Comparison for Determining the Shortest Route."

[17] I. Ahmad and W. Widodo, "khazanah informatika Jurnal Ilmu Komputer dan Informatika 57 Penerapan Algoritma A Star (A*) pada Game Petualangan Labirin Berbasis Android."