Application of the A* Algorithm for an Adaptive Pathfinding System in a Web-Based Maze Game with Random Maze Patterns

Ferry Khusnil Arief[1], Ilham Ismail[2], Dwi Utami[3]

[1] *Jurusan Teknik Informatika Universitas Nahdlatul Ulama Lampung;*
[2] *Jurusan Teknik Informatika Universitas Nahdlatul Ulama Lampung;*
[3] *Jurusan Teknik Informatika Universitas Nahdlatul Ulama Lampung;*
Email: ceryover@gmail.com

Abstract — *This study aims to develop a web-based maze adventure game by implementing the A algorithm as the primary navigation method to find the shortest path within the maze. The game is designed to provide an interactive gameplay experience while introducing players to the concept of intelligent navigation. The implemented navigation system enables the dynamic discovery of optimal routes based on the player's starting point and destination within the maze. By utilizing the A algorithm, this research is expected to contribute to the development of AI-based game technology that emphasizes efficiency and speed in pathfinding computation. The test results show that the A* algorithm is capable of generating efficient and accurate routes across various maze scenarios with differing levels of complexity. Furthermore, this study offers insights into how heuristic search algorithms can be implemented in web-based game applications to enhance the user experience*

Keywords: *A* Algorithm, Web-Based Game, Maze Game, Shortest Path, and Artificial Intelligence*

Intisari — Penelitian ini bertujuan untuk mengembangkan sebuah game petualangan labirin berbasis web dengan mengimplementasikan algoritma A* sebagai metode navigasi utama untuk menemukan jalur terpendek dalam labirin. Game ini dirancang untuk menyediakan pengalaman bermain yang interaktif sekaligus memperkenalkan konsep navigasi cerdas kepada pemain. Sistem navigasi yang diimplementasikan memungkinkan rute optimal ditemukan secara dinamis berdasarkan posisi awal dan tujuan pemain di dalam labirin. Dengan menggunakan algoritma A*, penelitian ini diharapkan dapat memberikan kontribusi pada pengembangan teknologi game berbasis kecerdasan buatan yang mengedepankan efisiensi dan kecepatan dalam perhitungan jalur. Hasil pengujian menunjukkan bahwa algoritma A* mampu menghasilkan rute yang efisien dan akurat dalam berbagai skenario labirin dengan tingkat kompleksitas yang berbeda. Selain itu, penelitian ini juga memberikan wawasan tentang bagaimana algoritma pencarian heuristik dapat diimplementasikan dalam aplikasi game berbasis web untuk meningkatkan pengalaman pengguna.

***Kata Kunci*:** Algoritma A*, Game Berbasis Web, Game Labirin, Jalur Terpendek, dan Kecerdasan Buatan.

I. PENDAHULUAN

A. Latar Belakang

Perkembangan teknologi game di era digital saat ini telah mengalami transformasi yang signifikan, terutama pada platform berbasis web. Permainan yang diakses melalui web tidak lagi hanya berfungsi sebagai media hiburan semata, tetapi telah berevolusi menjadi wahana untuk pengembangan dan implementasi algoritma kecerdasan buatan yang kompleks. Salah satu tantangan utama dalam desain game berjenis navigasi adalah menciptakan mekanisme pergerakan non-player character (NPC) yang cerdas dan efisien dalam lingkungan virtual.

Labirin, sebagai sebuah konsep permainan, memiliki sejarah panjang dalam dunia game, mulai dari permainan papan klasik hingga iterasi digital modern. Kompleksitas struktural dari labirin menuntut pengembang untuk merancang algoritma navigasi yang andal. Algoritma tersebut harus mampu menghasilkan rute yang optimal, menghindari rintangan secara dinamis, dan pada akhirnya menyajikan pengalaman bermain yang menantang dan menarik bagi pengguna.

Untuk mengatasi permasalahan navigasi pada lingkungan terstruktur berbasis grid, algoritma A* (A-Star) muncul sebagai solusi yang ideal. Keunggulan utamanya terletak pada kemampuannya untuk menemukan jalur terpendek secara efisien dengan memanfaatkan fungsi heuristik untuk memandu proses pencarian. Algoritma A* mengoptimalkan beban komputasi dengan meminimalkan jumlah simpul (node) yang dieksplorasi, menjadikannya sangat sesuai untuk aplikasi real-time di mana lingkungan permainan dapat berubah secara dinamis. Kombinasi antara keoptimalan dan efisiensi ini menjadikan algoritma A* sebagai landasan fundamental dalam sistem navigasi game modern.

B. Rumusan Masalah

- Bagaimana mengimplementasikan algoritma A* dalam game labirin?
- Bagaimana merancang mekanisme pencarian jalur otomatis?

C. Tujuan Penelitian

- Mengembangkan game labirin berbasis web.
- Menerapkan algoritma A* untuk navigasi cerdas.

II. LANDASAN TEORI

A. *Algoritma A* (A Star)*

Definisi dan Konsep Dasar Algoritma A* merupakan algoritma pencarian jalur terpendek yang termasuk dalam kategori algoritma informed search. Dikembangkan oleh Peter Hart, Nils Nilsson, dan Bertram Raphael pada tahun 1968, algoritma ini menjadi landasan fundamental dalam bidang kecerdasan buatan dan robotika.

Algoritma A* (A Star) merupakan metode pencarian yang efisien untuk menemukan rute terpendek dalam game petualangan labirin, dengan meminimalkan total biaya lintasan dan mempertimbangkan jarak sebenarnya serta estimasi jarak, sehingga menghasilkan solusi yang optimal [1]. Keunggulan algoritma A* dibandingkan algoritma lainnya terletak pada penggunaan nilai heuristik sebagai nilai pembanding, yang memungkinkan pencarian solusi terbaik dari permasalahan yang ada, asalkan solusi tersebut memang ada [2]. Dengan adanya heuristik, algoritma A* dijamin akan menemukan solusi jika solusi tersebut tersedia, menjadikannya fungsi optimasi yang membuat algoritma ini lebih unggul dibandingkan algoritma lainnya [3]. Selain itu, algoritma A* juga digunakan untuk menentukan rute terpendek objek menuju tujuan dengan menghitung biaya yang diperlukan untuk mencari harga terkecil yang harus dibayarkan[4]. Hal ini sejalan dengan hasil penelitian Nur Budi Nugraha [5], yang menunjukkan bahwa algoritma A* adalah metode yang efektif dalam mencari jalur terpendek, karena menggabungkan biaya perjalanan dan estimasi jarak ke tujuan[6].

Struktur Matematis Algoritma Fungsi utama algoritma A* didefinisikan sebagai: $f(n) = g(n) + h(n)$

Komponen:

- f(n): Total estimasi biaya perjalanan
- g(n): Biaya aktual dari node awal ke node saat ini
- h(n): Perkiraan biaya dari node saat ini ke node tujuan (heuristik)

B. *Teori Game*

Klasifikasi Game

- Tipe: Puzzle Navigation Game, Puzzle Navigation Game adalah jenis permainan di mana pemain dihadapkan pada tantangan untuk mencapai tujuan tertentu dengan memecahkan teka-teki yang melibatkan navigasi di dalam area permainan. Dalam permainan ini, strategi dan pemahaman pola sangat penting untuk menyelesaikan setiap level atau tantangan.
- Mekanisme: Grid-based Movement, Grid-based Movement mengacu pada mekanisme permainan di mana gerakan pemain dibatasi pada sebuah grid (kotak- kotak). Setiap langkah yang diambil akan berpindah dari satu kotak ke kotak lainnya secara horizontal, vertikal, atau diagonal, tergantung pada aturan permainan.
- Kategori: Strategis, Game strategis membutuhkan perencanaan, pemikiran kritis, dan pengambilan keputusan yang matang. Pemain dituntut untuk merancang langkah-langkah terbaik untuk mengatasi tantangan, mengalahkan lawan, atau menyelesaikan misi.

C. *Teori Labirin*

Labirin adalah struktur atau pola berliku-liku yang dirancang sebagai tantangan untuk menemukan jalan keluar atau mencapai tujuan tertentu. Terdiri dari lorong-lorong yang bercabang dengan jalur utama serta jalan buntu, labirin sering kali digunakan untuk menguji kemampuan berpikir logis dan pemecahan masalah. Jenis labirin dapat bervariasi, mulai dari yang sederhana dengan satu jalur (unicursal) hingga yang kompleks dengan banyak percabangan dan teka-teki (*multicursal*). Selain digunakan untuk hiburan seperti di taman bermain atau permainan video, labirin juga memiliki nilai edukatif, melatih keterampilan perencanaan dan navigasi. Dalam konteks budaya, labirin sering diartikan sebagai simbol perjalanan hidup atau pencarian makna, sementara dalam sains digunakan untuk eksperimen perilaku. Labirin mencerminkan perpaduan antara tantangan fisik dan mental, menciptakan pengalaman yang menarik dan mendalam bagi pemain atau pengamat.

D. *Teknologi Web*

- HTML (*HyperText Markup Language*), HTML adalah bahasa markup yang digunakan untuk membuat struktur dasar sebuah halaman web. Elemen-elemen HTML berfungsi sebagai kerangka situs, menentukan bagian-bagian seperti teks, gambar, tautan, tabel, dan formulir.
- CSS (*Cascading Style Sheets*), CSS digunakan untuk mengatur tampilan dan gaya elemen-elemen yang dibuat dengan HTML. Dengan CSS, pengembang dapat menambahkan warna, ukuran, tata letak, dan animasi sehingga halaman terlihat lebih menarik dan sesuai dengan desain yang diinginkan.
- JavaScript, JavaScript adalah bahasa pemrograman yang digunakan untuk menambahkan interaktivitas dan fungsionalitas dinamis ke halaman web. Dengan JavaScript, pengembang dapat membuat halaman web yang merespons tindakan pengguna seperti mengklik tombol, mengisi formulir, atau menggulir halaman.
- GitHub Pages, adalah fitur yang disediakan oleh GitHub yang memungkinkan pengguna untuk menghosting situs web statis secara gratis, sangat berguna bagi pengembang, penulis, dan siapa saja yang ingin mempublikasikan konten secara online tanpa harus mengelola server atau infrastruktur hosting.

III. METODOLOGI PENELITIAN

A. *Analisis Pembangunan Game*

1) Pengenalan Game : *"Find Hitagi!"* adalah sebuah permainan berbasis web yang mengajak pemain untuk menjelajahi labirin yang dihasilkan secara acak. Pemain harus menggunakan tombol navigasi untuk menggerakkan karakter Araragi Koyomi melalui labirin dan mencapai titik akhir yang ditandai sebagai Senjougahara Hitagi. Game ini dapat diakses melalui browser pada computer maupun smartphone yang memiliki akses internet pada *https://ariefshecter.github.io/GameLabirin/*

2) Alur Permainan:

a) Mulai Permainan: Saat pemain membuka game, mereka akan disambut dengan judul dan deskripsi singkat tentang tujuan permainan, yaitu membantu Araragi Koyomi menemukan Senjougahara Hitagi.
b) Pengaturan Awal: Permainan secara otomatis menghasilkan labirin acak menggunakan algoritma tertentu. Pemain akan diberikan titik awal (start) dan titik akhir (end) yang ditandai dalam labirin.
c) Navigasi: Pemain menggunakan tombol panah pada keyboard atau tombol navigasi di layar untuk menggerakkan karakter Araragi Koyomi. Tujuannya adalah untuk bergerak melalui jalur yang tersedia dan menghindari dinding (tembok) yang menghalangi.
d) Mengumpulkan Statistik: Selama permainan, jumlah langkah yang diambil akan dihitung dan ditampilkan. Pemain juga dapat melihat jarak terpendek yang dapat dicapai untuk menyelesaikan labirin.
e) Mencapai Titik Akhir: Pemain harus menavigasi labirin hingga mencapai titik akhir yang ditandai. Ketika karakter mencapai titik akhir, permainan akan memberikan umpan balik kepada pemain, termasuk jumlah langkah yang diambil dan jarak terpendek yang telah dicapai.
f) Opsi Tambahan: Pemain memiliki opsi untuk mereset permainan jika ingin memulai lagi, membuat peta baru untuk tantangan yang berbeda, atau menampilkan jalur terpendek yang telah ditemukan sebelumnya untuk membantu navigasi.
g) Selesai Permainan: Setelah mencapai titik akhir, pemain dapat melihat pesan kemenangan yang menunjukkan langkah yang diambil dan jalur terpendek. Pemain dapat memilih untuk bermain lagi atau menutup permainan.

3) Misi Permainan :

a) Menemukan Senjougahara Hitagi: Misi utama dari permainan ini adalah membantu karakter Araragi Koyomi untuk menemukan Senjougahara Hitagi yang berada di titik akhir labirin. Pemain harus menavigasi melalui jalur yang tersedia dan menghindari dinding untuk mencapai tujuan ini.
b) Menyelesaikan Labirin dengan Efisien: Pemain diharapkan untuk menyelesaikan labirin dalam jumlah langkah yang minimal. Ini menantang pemain untuk berpikir strategis dalam memilih jalur yang akan diambil, sehingga mereka dapat mencapai titik akhir dengan seefisien mungkin.
c) Mengumpulkan Statistik: Selama permainan, pemain akan mengumpulkan statistik terkait jumlah langkah yang diambil dan jarak terpendek yang dapat ditempuh. Misi ini mendorong pemain untuk berusaha meningkatkan kinerja mereka dalam setiap permainan.
d) Mengatasi Tantangan Labirin: Pemain harus mampu mengatasi berbagai tantangan yang muncul dalam labirin, seperti dinding yang menghalangi jalur dan memikirkan langkah-langkah yang tepat untuk mencapai titik akhir. Ini menciptakan elemen tantangan dan ketegangan dalam permainan.
e) Menggunakan Opsi Permainan: Pemain memiliki opsi untuk mereset permainan, membuat peta baru, atau menampilkan jalur terpendek. Misi ini memberikan fleksibilitas kepada pemain untuk menyesuaikan pengalaman bermain mereka dan mencoba strategi yang berbeda.

*B. Analisis Algoritma A**

1) Inisialisasi Variabel

```
const start = {x: this.player.x, y: this.player.y};
const end = {x: this.end.x, y: this.end.y};
```

Tujuan: Mendefinisikan titik awal dan akhir berdasarkan posisi pemain (this.player) dan tujuan akhir (this.end) di labirin.

```
const h = (pos) => {
  return Math.abs(pos.x - end.x) + Math.abs(pos.y - end.y);
};
```

Tujuan: Membuat fungsi heuristik (h) menggunakan jarak Manhattan untuk memperkirakan jarak dari titik saat ini ke tujuan akhir. Heuristik digunakan untuk memandu algoritma A* menuju solusi.

2) Inisialisasi Struktur Data

```
const pq = new PriorityQueue();
const visited = new Set();
const gScore = {};
const fScore = {};
const cameFrom = {};
```

Tujuan:

- PriorityQueue: Menyimpan node yang akan diproses, dengan prioritas berdasarkan nilai fScore.
- visited: Menyimpan node yang sudah dikunjungi agar tidak diproses ulang.
- gScore: Menyimpan jarak terpendek yang diketahui dari titik awal ke setiap node.
- fScore: Menyimpan perkiraan total jarak dari titik awal ke tujuan melalui node tertentu (fScore = gScore + h).

- cameFrom: Melacak node asal untuk merekonstruksi jalur setelah menemukan solusi.

3) Inisialisasi Nilai untuk Titik Awal

```
gScore[`${start.x},${start.y}`] = 0;
fScore[`${start.x},${start.y}`] = h(start);
pq.enqueue(start, fScore[`${start.x},${start.y}`]);
```

Tujuan:

- Mengatur gScore titik awal menjadi 0 karena jaraknya dari dirinya sendiri adalah 0.
- Menghitung fScore untuk titik awal menggunakan fungsi heuristik.
- Menambahkan titik awal ke antrian prioritas dengan nilai fScore sebagai prioritasnya.

4) Algoritma Utama (While Loop)

```
while (pq.values.length > 0) {
  const current = pq.dequeue().val;
  const currentKey = `${current.x},${current.y}`;
```

Tujuan:

- Melakukan iterasi selama antrian prioritas tidak kosong.
- Mengambil node dengan prioritas tertinggi (nilai fScore terkecil) dari antrian.

5) Pengecekan Titik Akhir

```
if (current.x === end.x && current.y === end.y) {
  this.shortestPath = gScore[currentKey];
  this.shortestElement.textContent = this.shortestPath;
```

Tujuan: Jika node saat ini adalah tujuan akhir:

- Menyimpan jarak terpendek dalam variabel shortestPath.
- Menampilkan jarak terpendek pada elemen UI (shortestElement).

6) Rekonstruksi Jalur Terpendek

```
this.shortestPathCells = [];
let curr = end;
while (curr) {
  this.shortestPathCells.push(curr);
  curr = cameFrom[`${curr.x},${curr.y}`];
}
this.shortestPathCells.reverse();
return;
```

Tujuan:

- Menelusuri jalur terpendek dari tujuan kembali ke awal menggunakan cameFrom.
- Membalik urutan jalur untuk mendapatkan urutan dari awal ke tujuan.
- Menyimpan jalur terpendek dalam shortestPathCells.

7) Tandai Node Sebagai Dikunjungi

```
visited.add(currentKey);
```

Tujuan: Menambahkan node saat ini ke dalam set visited agar tidak diproses ulang

8) Proses Tetangga

```
const directions = [[0, 1], [1, 0], [0, -1], [-1, 0]];
for (let [dx, dy] of directions) {
  const newX = current.x + dx;
  const newY = current.y + dy;
```

Tujuan: Menentukan arah pergerakan (atas, kanan, bawah, kiri) dan menghitung koordinat tetangga.

```
if (
  newX >= 0 && newX < this.maze[0].length &&
  newY >= 0 && newY < this.maze.length &&
  this.maze[newY][newX] !== 1
) {
```

Tujuan: Memastikan tetangga berada dalam batas labirin (this.maze) dan bukan dinding (1).

9) Update Tetangga

```
const neighbor = {x: newX, y: newY};
const neighborKey = `${newX},${newY}`;
if (visited.has(neighborKey)) continue;
```

Tujuan:

- Membuat objek tetangga (neighbor) dan kunci uniknya (neighborKey).
- Melewati tetangga yang sudah dikunjungi.

```
const tentativeGScore = gScore[currentKey] + 1;
if (!gScore[neighborKey] || tentativeGScore < gScore[neighborKey]) {
  cameFrom[neighborKey] = current;
  gScore[neighborKey] = tentativeGScore;
  fScore[neighborKey] = tentativeGScore + h(neighbor);
  pq.enqueue(neighbor, fScore[neighborKey]);
}
```

Tujuan:

- Menghitung nilai sementara gScore untuk tetangga.
- Memperbarui cameFrom, gScore, dan fScore jika jarak baru lebih pendek dari jarak yang diketahui sebelumnya.
- Menambahkan tetangga ke antrian prioritas.

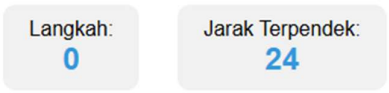
IV. HASIL DAN PEMBAHASAN

A. Fitur Game

1. Navigasi: Pemain dapat menggunakan tombol panah pada keyboard atau tombol navigasi yang disediakan untuk bergerak ke atas, bawah, kiri, atau kanan dalam labirin.

Gambar 1. Navigasi

2. Statistik Permainan: Terdapat informasi mengenai jumlah langkah yang diambil dan jarak terpendek yang berhasil dicapai.



Gambar 2. Statistik Permainan

3. Kontrol Permainan: Pemain dapat mereset permainan, membuat peta baru, atau menampilkan jalur terpendek yang telah ditemukan.

Gambar 3. Kontrol Permainan

4. Karakter dan Titik Finish: Pemain dapat melihat karakter Araragi Koyomi dan titik finish Senjougahara Hitagi dalam permainan.

Gambar 4. Karakter

Gambar 5. finish

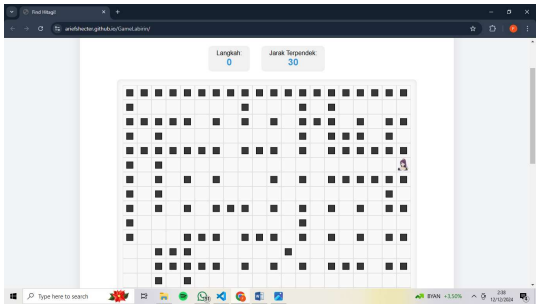
5. Instruksi: Terdapat panduan yang menjelaskan cara bermain, yaitu dengan menggunakan tombol navigasi untuk mencapai titik finish secepat mungkin.

Gambar 6. Instruksi

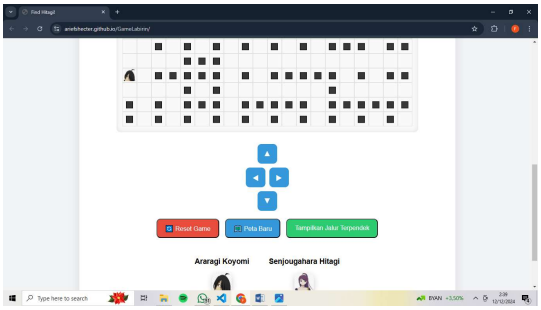
B. *Hasil Implementasi Algoritma A* Pada Game*

Dalam permainan "Find Hitagi!", penerapan algoritma A* berperan penting dalam membantu pemain menemukan jalur terpendek dari titik awal karakter, Araragi Koyomi, menuju titik finish, Senjougahara Hitagi. Untuk memanfaatkan fitur ini, pemain dapat menggunakan opsi "Tampilkan Jalur Terpendek", yang akan menampilkan hasil pencarian jalur terpendek yang dilakukan oleh algoritma A*. Dengan demikian, pemain dapat dengan mudah menentukan rute yang paling efisien untuk mencapai tujuan mereka dalam permainan.

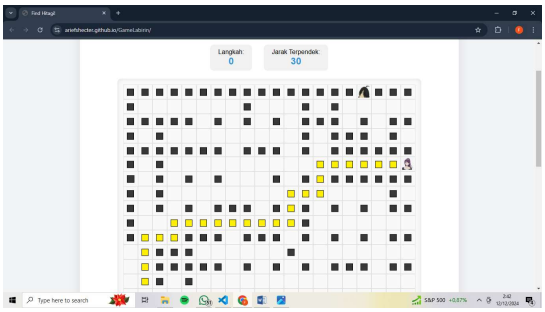
Gambar 7. Fitur Tampilkan Jalur Terpendek



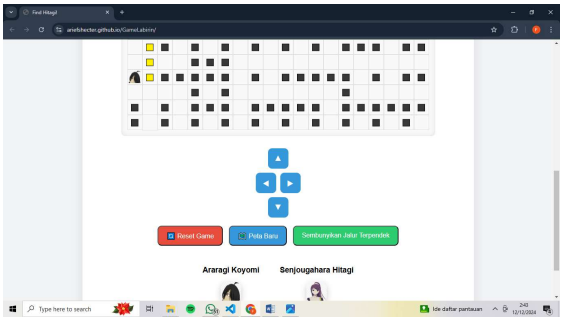
Gambar 8. Map Sebelum Fitur A* 1



Gambar 9. Map Sebelum Fitur A* 1



Gambar 10. Map Sesudah Fitur A* 1



Gambar 11. Map Sesudah Fitur A* 2

IV. KESIMPULAN DAN SARAN

A. Kesimpulan

Penelitian ini berhasil mengembangkan game petualangan labirin berbasis web dengan menerapkan algoritma A* untuk menemukan jalur terpendek. Berdasarkan rumusan masalah yang diajukan, berikut adalah kesimpulan yang menjawab kedua pertanyaan tersebut:

1. Implementasi algoritma A* dalam game labirin dilakukan dengan langkah-langkah sebagai berikut:
   - Inisialisasi Posisi: Pertama, posisi awal (start) dan posisi tujuan (end) ditentukan. Fungsi heuristik yang digunakan adalah jarak Manhattan, yang menghitung estimasi biaya dari posisi saat ini ke posisi tujuan.
   - Struktur Data: Antrian prioritas (*Priority Queue*) digunakan untuk menyimpan dan mengambil node yang akan dieksplorasi berdasarkan nilai f-score mereka. Selain itu, gScore dan fScore diinisialisasi untuk melacak biaya perjalanan aktual dan estimasi biaya total.
   - Loop Pencarian: Algoritma A* melakukan pencarian dengan mengeksplorasi node yang memiliki fScore terendah. Jika node saat ini adalah tujuan, jalur terpendek ditemukan dan direkonstruksi. Jika tidak, tetangga dari node saat ini dieksplorasi, dan gScore serta fScore diperbarui sesuai dengan biaya yang ditemukan.
   - Rekonstruksi Jalur: Setelah mencapai tujuan, jalur terpendek direkonstruksi dengan melacak dari node tujuan kembali ke node awal menggunakan struktur data yang telah disiapkan.
2. Mekanisme pencarian jalur otomatis dirancang dengan mempertimbangkan beberapa aspek:
   - Penggunaan Fungsi Heuristik: Fungsi heuristik yang efisien, seperti jarak Manhattan, memungkinkan algoritma untuk memperkirakan biaya perjalanan dengan baik, sehingga mempercepat proses pencarian jalur.
   - Pengelolaan Node yang Dijalani: Dengan menggunakan struktur data seperti antrian prioritas dan set untuk melacak node yang telah dikunjungi, algoritma dapat menghindari eksplorasi ulang pada node yang sama, sehingga meningkatkan efisiensi pencarian.
   - Interaksi Pengguna: Pemain diberikan opsi untuk menampilkan jalur terpendek yang ditemukan oleh algoritma A*, yang tidak hanya memberikan umpan balik langsung tetapi juga membantu pemain memahami strategi navigasi yang lebih baik.
   - Fleksibilitas dalam Permainan: Pemain dapat mereset permainan, membuat peta baru, atau menampilkan jalur terpendek, memberikan pengalaman bermain yang dinamis dan interaktif.

Dengan demikian, penelitian ini menunjukkan bahwa penerapan algoritma A* dalam game labirin tidak hanya efektif dalam menemukan jalur terpendek, tetapi juga memberikan pengalaman bermain yang lebih menarik dan menantang bagi pemain.

B. Saran

Dalam rangka meningkatkan kualitas dan kedalaman penelitian di bidang pengembangan game berbasis web dengan algoritma A*, beberapa saran untuk penelitian selanjutnya dapat dipertimbangkan. Pertama, eksplorasi algoritma pencarian lainnya, seperti Dijkstra, Breadth-First Search (BFS), atau Depth-First Search (DFS), dapat dilakukan untuk membandingkan efektivitas masing-masing dalam konteks labirin. Selain itu, pengembangan kecerdasan buatan (AI) untuk karakter non-pemain (NPC) yang dapat berinteraksi dengan pemain dan menyesuaikan strategi mereka berdasarkan tindakan pemain dapat meningkatkan dinamika permainan. Penelitian lebih mendalam tentang pengalaman pengguna (UX) juga penting untuk memahami interaksi pemain dengan game, serta penerapan teknik pembelajaran mesin untuk mengadaptasi algoritma pencarian berdasarkan pola perilaku pemain. Selain itu, pengembangan game edukasi yang mengajarkan konsep-konsep matematika, logika, atau pemrograman melalui mekanisme permainan interaktif dapat menjadi fokus penelitian. Studi kasus yang membandingkan performa dan pengalaman pengguna game labirin di berbagai platform, seperti desktop, tablet, dan smartphone, juga dapat memberikan wawasan berharga. Penelitian tentang penerapan teknologi baru, seperti Virtual Reality (VR) atau Augmented Reality (AR), dalam game labirin dapat mengeksplorasi bagaimana teknologi ini dapat meningkatkan pengalaman bermain. Terakhir, analisis kinerja algoritma A* dalam berbagai jenis labirin dengan tingkat kompleksitas dan ukuran yang berbeda dapat membantu memahami batasan dan keunggulan algoritma dalam konteks yang lebih luas. Dengan mengikuti saran-saran ini, penelitian selanjutnya diharapkan dapat memberikan kontribusi yang lebih besar terhadap pengembangan game berbasis web dan penerapan algoritma pencarian.

DAFTAR PUSTAKA

[1] I. Ahmad and W. Widodo, “Penerapan Algoritma A Star (A*) pada Game Petualangan Labirin Berbasis Android,” 2017.

[2] E. P. Widiyanto, “Penerapan Algoritma A* (A Star) Pada Game Edukasi The Maze Island Berbasis Android,” 2014, doi: 10.13140/RG.2.2.16459.77601.

[3] Rengga Dionata Putra, “Pencarian Rute Terdekat Pada Labirin Menggunakan Metode A*,” 2012.

[4] D. Y. A. Fallo and V. R. Bulu, “PENERAPAN ALGORITMA A STAR (A*) PADA GAME LABIRIN,” Jurnal Pendidikan Teknologi Informasi (JUKANTI), no. 5, pp. 2621–1467, 2022.

[5] Irianto, B. T. D., Andryana, S., & Gunaryati, A. (2021). Penerapan Algoritma A-Star Dalam Mencari Jalur Tercepat dan Pergerakan NonPlayer Character Pada Game Petualangan Labirin Tech-Edu. *J. Media Inform. Budidarma*, *5*(3), 953.

[6] P. Studi Informatika and S. I. Tinggi Teknologi Dumai Jl Utama Karya Bukit Batrem, “Rancang Bangun Game Labirin Menggunakan Algoritma A Star Berbasis Mobile Nur Budi Nugraha,” vol. 11, no. 2, 2018.