OPTIMASI RUTE MENGGUNAKAN ALGORITMA A* DAN *BRUTE FORCE* PADA *GAME* SIMULASI *TRAVELING SALESMAN PROBLEM*

SKRIPSI

Oleh :
RIDWAN JAUHAR KAFABIHI
NIM. 210605110069

PROGRAM STUDI TEKNIK INFORMATIKA
FAKULTAS SAINS DAN TEKNOLOGI
UNIVERSITAS ISLAM NEGERI MAULANA MALIK IBRAHIM
MALANG
2025

OPTIMASI RUTE MENGGUNAKAN ALGORITMA A* DAN *BRUTEFORCE* PADA *GAME* SIMULASI TRAVELING SALESMAN PROBLEM

SKRIPSI

Diajukan kepada:
Universitas Islam Negeri Maulana Malik Ibrahim Malang
Untuk memenuhi Salah Satu Persyaratan dalam
Memperoleh Gelar Sarjana Komputer (S.Kom)

Oleh :
RIDWAN JAUHAR KAFABIHI
NIM. 210605110069

PROGRAM STUDI TEKNIK INFORMATIKA
FAKULTAS SAINS DAN TEKNOLOGI
UNIVERSITAS ISLAM NEGERI MAULANA MALIK IBRAHIM MALANG
2025

HALAMAN PERSETUJUAN

OPTIMASI RUTE MENGGUNAKAN ALGORITMA A* DAN *BRUTE FORCE* PADA *GAME* SIMULASI TRAVELING SALESMAN PROBLEM

SKRIPSI

Oleh :
RIDWAN JAUHAR KAFABIHI
NIM. 210605110069

Telah Diperiksa dan Disetujui untuk Diuji
6 Mei 2025

Pembimbing I,	Pembimbing II,
Ahmad Fahmi Karami, M.Kom	Nur Fitriyah Ayu Tunjung Sari, M.Cs
NIP. 19870909 202012 1 001	NIP. 19911226 202012 2 001

Mengetahui,
Ketua Program Studi Teknik Informatiksa
Fakultas Sains dan Teknologi
Universitas Islam Negeri Maulana Malik Ibrahim Malang

Dr. Ir. Fachrul Kurniawan, M.MT., IPU
NIP. 19771020 200912 1 001

HALAMAN PENGESAHAN

OPTIMASI RUTE MENGGUNAKAN ALGORITMA A* DAN *BRUTE FORCE* PADA *GAME* SIMULASI TRAVELING SALESMAN PROBLEM

SKRIPSI

Oleh :
RIDWAN JAUHAR KAFABIHI
NIM. 210605110069

Telah Dipertahankan di Depan Dewan Penguji Skripsi
dan Dinyatakan Diterima Sebagai Salah Satu Persyaratan
Untuk Memperoleh Gelar Sarjana Komputer (S.Kom)
Tanggal: 21 Mei 2025

Susunan Dewan Penguji

Ketua Penguji	: Dr. Yunifa Miftachul Arif, M.T NIP. 19830616 201101 1 004	( )
Anggota Penguji I	: Ashri Shabrina Afrah, M.T NIP. 19900430 202012 2 003	( )
Anggota Penguji II	: Ahmad Fahmi Karami, M.Kom NIP. 19870909 202012 1 001	( )
Anggota Penguji III	: Nur Fitriyah Ayu Tunjung Sari, M.Cs NIP. 19911226 202012 2 001	( )

Mengetahui dan Mengesahkan,
Ketua Program Studi Teknik Informatika
Fakultas Sains dan Teknologi
Universitas Islam Negeri Maulana Malik Ibrahim Malang

Dr. Ir. Fachrul Kurniawan, M.MT., IPU
NIP. 19771020 200912 1 001

PERNYATAAN KEASLIAN TULISAN

Saya yang bertanda tangan di bawah ini:

Nama	: Ridwan Jauhar Kafabihi
NIM	: 210605110069
Fakultas / Prodi	: Sains dan Teknologi / Teknik Informatika
Judul Skripsi	: Optimasi Rute Menggunakan Algoritma A* dan *Brute Force* Pada *Game* Simulasi *Travelling Salesman Problem*

Menyatakan dengan sebenarnya bahwa Skripsi yang saya tulis ini benar-benar merupakan hasil karya saya sendiri, bukan merupakan pengambil alihan data, tulisan, atau pikiran orang lain yang saya akui sebagai hasil tulisan atau pikiran saya sendiri, kecuali dengan mencantumkan sumber cuplikan pada daftar pustaka.

Apabila dikemudian hari terbukti atau dapat dibuktikan skripsi ini merupakan hasil jiplakan, maka saya bersedia menerima sanksi atas perbuatan tersebut.

Malang, 21 Mei 2025
Yang membuat pernyataan,

Ridwan Jauhar Kafabihi
NIM.210605110069

MOTTO

...Though tears may fall and resolve may weaken, yet thou must rise and endure…

HALAMAN PERSEMBAHAN

Segala puji bagi Allah Subhanahu wa Ta'ala dengan segala limpahan rahmat dan karunia-Nya, sehingga penulis bisa menyelesaikan skripsi ini. Shalawat serta salam penulis curahkan kepada nabi Muhammad Shallallahu 'Alaihi Wasallam.

Karya ini penulis persembahkan kepada kedua orangtua tercinta Bapak Agus Susilo dan Ibu Dewi Sulastri. Terimakasih penulis ucapkan terhadap semua dukungan yang diberikan baik ilmu, materi, maupun doa di setiap harinya sehingga penulis mampu menyelesaikan penulisan karya tulis ini. Penulis juga mengucapkan terimakasih kepada kakak – kakak saya yang telah membantu memotivasi saya dalam perjalanan penulisan skripsi ini. Serta kepada seorang yang dengan sabar mendampingi dalam penulisan karya ini.

Semoga ilmu yang didapat penulis dapat berguna dan berkah bagi penulis dan orang-orang disekitarnya. Penulis juga mengucapkan terimakasih dan apresiasi yang sangat besar terhadap Bpk. Ahmad Fahmi Karami selaku dosen pembimbing yang memberikan arahan penulisan skripsi ini. Penulis turut mengucapkan terimakasih sebesar-besarnya terhadap pihak terkait yang membantu dalam penulisan ini termasuk Akhmad Badawi selaku tim pengerjaan projek, teman-teman "ASTER", teman-teman Prodi Teknik Informatika, dan pihak-pihak yang belum tersebut lainnya

KATA PENGANTAR

Assalamualaikum Warahmatullahi Wabarakatuh.

Puji syukur kehadirat Allah SWT atas segala limpahan rahmat, hidayah, dan karunia-Nya, sehingga penulis dapat menyelesaikan penyusunan skripsi dengan judul *"Optimasi Rute Menggunakan Algoritma A dan Brute Force pada Game Simulasi Traveling Salesman Problem"* sebagai salah satu syarat untuk memperoleh gelar Sarjana Komputer (S.Kom) pada Program Studi Teknik Informatika, Fakultas Sains dan Teknologi, Universitas Islam Negeri Maulana Malik Ibrahim Malang.

Skripsi ini tidak akan dapat terselesaikan tanpa dukungan dan bantuan dari berbagai pihak. Oleh karena itu, pada kesempatan ini penulis ingin menyampaikan ucapan terima kasih yang sebesar-besarnya kepada:

1. Allah SWT, atas segala nikmat dan kekuatan yang diberikan selama proses penyusunan skripsi ini.
2. Bapak dan Ibu tercinta, atas doa, cinta, dan dukungan moral maupun materiil yang tidak pernah berhenti.
3. Prof. Dr. H. M. Zainuddin, M.A., selaku Rektor Universitas Islam Negeri Maulana Malik Ibrahim Malang.
4. Prof. Dr. Hj. Sri Hariani, M.Si., selaku Dekan Fakultas Sains dan Teknologi Universitas Islam Negeri Maulana Malik Ibrahim Malang.
5. Dr. Fachrul Kurniawan, M.MT., IPM, selaku Ketua Program Studi Teknik Informatika Universitas Islam Negeri Maulana Malik Ibrahim Malang.
6. Bapak Ahmad Fahmi Karami, M.Kom selaku dosen pembimbing I dan Ibu Nur Fitriyah Ayu Tunjung Sari, M.Cs selaku dosen pembimbing II yang telah

memberikan bimbingan, arahan, dengan penuh kesabaran selama proses penyusunan skripsi ini.

7. Dr. Yunifa Miftachul Arif, M. T selaku dosen penguji I dan Ashri Shabrina Afrah, M.T, selaku dosen penguji II yang telah memberikan arahan perbaikan sehingga skripsi dapat diselesaikan dengan baik.
8. Seluruh dosen dan staf pengajar Teknik Informatika, atas ilmu yang telah diberikan selama masa perkuliahan.
9. Teman-teman seperjuangan dan sahabat-sahabat tercinta, atas kebersamaan, semangat, dan doa yang tak ternilai harganya.
10. Semua pihak yang tidak dapat disebutkan satu per satu, yang telah memberikan kontribusi dalam penyusunan skripsi ini.

Penulis menyadari bahwa skripsi ini masih jauh dari sempurna. Oleh karena itu, kritik dan saran yang membangun sangat penulis harapkan demi penyempurnaan di masa mendatang.

Akhir kata, semoga skripsi ini dapat memberikan manfaat dan kontribusi positif bagi pembaca, khususnya dalam pengembangan ilmu di bidang teknik informatika.

Malang, 16 Mei 2025

Ridwan Jauhar Kafabihi

DAFTAR ISI

HALAMAN PENGAJUAN ..... **ii**
HALAMAN PERSETUJUAN ..... **iii**
HALAMAN PENGESAHAN ..... **iv**
PERNYATAAN KEASLIAN TULISAN ..... **v**
MOTTO ..... **vi**
HALAMAN PERSEMBAHAN ..... **vii**
KATA PENGANTAR ..... **viii**
DAFTAR ISI ..... **x**
DAFTAR GAMBAR ..... **xii**
DAFTAR TABEL ..... **xiv**
ABSTRAK ..... **xv**
ABSTRACT ..... **xvi**
البحث مستخلص ..... **xvii**
BAB I PENDAHULUAN **18**
1.1 Latar Belakang 18
1.2 Rumusan Masalah 22
1.3 Batasan Masalah 22
1.4 Tujuan Penelitian 22
1.5 Manfaat Penelitian 22
BAB II STUDI PUSTAKA **23**
2.1 Penelitian Terkait 23
2.2 *Game* 25
2.3 *Finite State Machine* (FSM) 28
2.4 *Traveling Salesman Problem* 29
2.5 *Pathfinding* 30
2.6 Algoritma *Brute Force* 32
2.7 Algoritma A* 33
BAB III METODE PENELITIAN **36**
3.1 Objek Penelitian 36
3.2 *Finite State Machine* (FSM) 38
3.3 Rancangan Pengujian 39
3.4 Skenario Pengujian 39
3.4.1 Pengujian Algoritma Brute Force 39
3.4.2 Pengujian Algoritma A* 40
3.4.3 Pengujian Game User Experience Satisfaction Scale (GUESS) 41
3.5 Studi Kasus 43
3.6 Permasalahan TSP 44
3.7 Permasalahan A* 49
BAB IV HASIL DAN PEMBAHASAN **57**
4.1 Implementasi 57
4.2 Implementasi Algoritma *Brute Force* 57
4.3 Implementasi Algoritma A* 60
4.4 Pengujian 63
4.5 Pengujian *Brute Force* 63
4.6 Pengujian A* 76
4.7 Pengujian *Usability* 88

4.8 Integrasi dengan Islam.....92
BAB V KESIMPULAN DAN SARAN.....95
5.1 Kesimpulan 95
5.2 Saran 96

DAFTAR GAMBAR

Gambar 3. 1 *Flowchart Core Game*play *Loop* 36
Gambar 3. 2 Ilustrasi *Game* 37
Gambar 3. 3 FSM 38
Gambar 3. 4 Tatanan Skenario *Game* 43
Gambar 3. 5 Skenario objektif *game* 44
Gambar 3. 6 Hasil Sekenario TSP 48
Gambar 3. 7 Sekenario objektif *game* 49
Gambar 3. 8 Langkah Ekspansi Selanjutnya 51
Gambar 3. 9 Jalur telah ditemukan 52
Gambar 3. 10 Jalur E - D 54
Gambar 3. 11 Jalur D - C 54
Gambar 3. 12 Jalur C - B 55
Gambar 3. 13 Jalur B - A 55
Gambar 3. 14 Jalur selesai ditentukan 56
Gambar 4. 1 Penanaman Pohon 64
Gambar 4. 2 Pengecatan Bangunan 65
Gambar 4. 3 Pelebaran Sungai 65
Gambar 4. 4 Peninggian Bangunan 66
Gambar 4. 5 Penanaman Rumput 67
Gambar 4. 6 Hutan Kota 67
Gambar 4. 7 Urban Garden 68
Gambar 4. 8 Danau Buatan 69
Gambar 4. 9 Titik A ke B (Uji 1) 77
Gambar 4. 10 Titik A ke B (Uji 2) 77
Gambar 4. 11 Titik A ke B (Uji 3) 78
Gambar 4. 12 Titik B ke C (Uji 1) 78
Gambar 4. 13 Titik B ke C (Uji 2) 79
Gambar 4. 14 Titik B ke C (Uji 3) 79
Gambar 4. 15 Titik C ke D (Uji 1) 80
Gambar 4. 16 Titik C ke D (Uji 2) 80
Gambar 4. 17 Titik C ke D (Uji 3) 81
Gambar 4. 18 Titik D ke E (Uji 1) 81
Gambar 4. 19 Titik D ke E (Uji 2) 82
Gambar 4. 20 Titik D ke E (Uji 3) 83
Gambar 4. 21 Titik E ke F (Uji 1) 83
Gambar 4. 22 Titik E ke F (Uji 2) 84
Gambar 4. 23 Titik E ke F (Uji 3) 84
Gambar 4. 24 Titik F ke A (Uji 1) 85
Gambar 4. 25 Titik F ke A (Uji 2) 85

Gambar 4. 26 Titik F ke A (Uji 3) 86
Gambar 4. 27 Diagram *Ease Of Use* 90
Gambar 4. 28 Diagram *Navigation* 90
Gambar 4. 29 Diagram *Understanding* 91

DAFTAR TABEL

Tabel 3. 1 Skenario *Game* 39
Tabel 3. 2 Tabel Pengujian A* 41
Tabel 3. 3 Tabel kuisoner 42
Tabel 3. 4 Klasifikasi Skor *Usability* 43
Tabel 3. 5 Tabel Detail Objektif 45
Tabel 3. 6 Tabel jarak antar kota 46
Tabel 3. 7 Tabel *cost* rute optimal 47
Tabel 3. 8 Tabel *Open list* 1 50
Tabel 3. 9 Tabel *Closed list* 1 50
Tabel 3. 10 Tabel *Open list* 2 50
Tabel 3. 11 Tabel *Open list* 3 51
Tabel 3. 12 Tabel Closed list 2 52
Tabel 3. 13 Tabel *Closed list* Akhir 53
Tabel 4. 1 *Pseudocode* Membuat Permutasi 58
Tabel 4. 2 *Pseudocode* Membuat Matriks Jarak 59
Tabel 4. 3*Pseudocode* Slove TSP 59
Tabel 4. 4 *Pseudocode* Kondisi Jalan A* 61
Tabel 4. 5 *Pseudocode* Fungsi Move Along Path 61
Tabel 4. 6 *Pseudocode* Fungsi A* 62
Tabel 4. 7 Tabel Blackbox 70
Tabel 4. 8 Tabel Perbandingan Waktu 87
Tabel 4. 9 Hasil Pengujian *Usability* 89

ABSTRAK

Kafabihi, Ridwan Jauhar. 2025. **Optimasi Rute Menggunakan Algoritma A* Dan *Brute Force* Pada *Game* Simulasi Traveling Salesman Problem.** Skripsi. Prodi Teknik Informatika Fakultas Sains dan Teknologi Universitas Islam Negeri Maulana Malik Ibrahim Malang. Pembimbing: (I) Ahmad Fahmi Karami, M.Kom (II) Nur Fitriyah Ayu Tunjung Sari, M.Cs

.

Penelitian ini bertujuan untuk menerapkan Algoritma *Brute Force* dan A* dalam *game CoolCity*, sebuah *game* simulasi edukatif berbasis *Travelling Salesman Problem* (TSP), guna meningkatkan efisiensi pemain dalam menentukan jalur optimal dalam skenario multi-objektif. Algoritma *Brute Force* digunakan untuk menentukan urutan kunjungan lokasi dengan jarak total minimum, sedangkan algoritma A* diterapkan untuk mencari jalur terpendek antar lokasi. Hasil pengujian menunjukkan bahwa rute yang dihasilkan Algoritma *Brute Force* konsisten dengan perhitungan manual dan memberikan hasil optimal dari segi jarak. Sementara itu, pengujian algoritma A* terhadap tiga kali proses pengujian menunjukkan rata-rata selisih waktu tempuh sebesar 0,11 detik pada pengujian tanpa hambatan, 0,22 detik dengan hambatan festival, dan 9.55 detik pada pengujian dengan hambatan tambahan berupa lokasi festival dan lampu merah. Temuan ini menunjukkan bahwa keberadaan hambatan dinamis berdampak signifikan terhadap waktu tempuh aktual, sehingga jalur dengan biaya terendah tidak selalu menghasilkan waktu tercepat. Selain itu, hasil uji usability menggunakan pendekatan GUESS menunjukkan bahwa *game* memiliki tingkat keterpahaman, kemudahan navigasi, dan kepercayaan diri pengguna yang tinggi, dengan skor rata-rata 5,12 dari skala 7. Hal ini mengindikasikan bahwa *game CoolCity* berhasil memberikan pengalaman bermain yang baik dan efektif dalam menyampaikan tujuan edukatifnya.

Kata kunci: A*, *Brute Force*, *game* simulasi, travelling salesman problem

.

ABSTRACT

Kafabihi, Ridwan Jauhar. 2025. **Route Optimization Using A* Algorithm and Brute Force in Traveling Salesman Problem Simulation Game.** Thesis. Informatics Engineering Study Program, Faculty of Science and Technology, Maulana Malik Ibrahim State Islamic University, Malang. Promoter: (I) Ahmad Fahmi Karami, M.Kom (II) Nur Fitriyah Ayu Tunjung Sari, M.Cs

The purpose of this research is to implement the *Brute Force* and A* algorithms in *CoolCity*, an educational simulation *game* based on the Travelling Salesman Problem (TSP), to enhance players' efficiency in determining the optimal route in multi-objective scenarios. The *Brute Force* algorithm is employed to determine the optimal sequence of visits with the shortest total distance, while the A* algorithm is used to find the shortest path between locations. Test results indicate that the routes generated by the *Brute Force* algorithm are consistent with manual calculations and always yield the optimal distance. In contrast, A* algorithm tests across three scenarios show average travel time deviations of 0.11 seconds in the scenario without obstacles, 0.22 seconds in the scenario with static obstacles, and 9.55 seconds in the scenario with dynamic obstacles. These findings highlight that the presence of dynamic obstacles significantly affects actual travel time, indicating that a path with the lowest cost does not always result in the shortest travel time. Furthermore, the usability test using the GUESS method demonstrates that the *game* has a high level of user understanding, easy navigation, and strong user confidence, with an average score of 5.12 out of 7. These results suggest that *CoolCity* provides an effective and engaging educational experience.

Key words: A*, *Brute Force*, *game* simulasi, travelling salesman problem.

البحث مستخلص

كفابيهي، رضوان.2025.وخوارزمية القوة الغاشمة في لعبة محاكاة مشكلة البائع المتنقل (A*) تحسين المسار باستخدام خوارزمية . البحث الجامعي. قسم الهندسة المعلوماتية، كلية العلوم والتكنولوجيا بجامعة مولانا مالك إبراهيم الإسلامية الحكومية مالانج. المشرف الأول: أحمد فهمي كرامي ماجستير في الحاسب الآلي الثاني: نور فطرية أيو تونجونج ساري ، ماجستير في علوم الحاسب الآلي.

الكلمات الرئيسية: .القوة الغاشمة، لعبة المحاكاة، مشكلة البائع المتجول ،A*

وهي لعبة محاكاة تعليمية تعتمد ،(CoolCity) في (A*) يهدف هذا البحث إلى تطبيق خوارزمية القوة الغاشمة وخوارزمية لتحسين كفاءة اللاعب في تحديد المسار الأمثل في سيناريو متعدد الأهداف. تُستخدم خوارزمية ،(TSP) على مشكلة البائع المتنقل .لإيجاد أقصر مسار بين المواقع (A*) القوة الغاشمة لتحديد تسلسل زيارات المواقع بأقل مسافة إجمالية، بينما يتم تطبيق خوارزمية تُظهر نتائج الاختبار أن المسار الذي تم إنشاؤه بواسطة خوارزمية القوة الغاشمة يتوافق مع الحسابات اليدوية ويوفر النتائج المثلى من مقابل ثلاث عمليات اختبارية متوسط فرق في زمن السفر يبلغ (A*) حيث المسافة. وفي الوقت نفسه، أظهر اختبار خوارزمية ٠.١١ ثانية في الاختبار بدون عوائق، و٠.٢٢ ثانية مع وجود عوائق مهرجانية، و ٩.٥٥ ثانية في الاختبار مع وجود عوائق إضافية على شكل مواقع مهرجانات وإشارات ضوئية حمراء. تُظهر هذه النتيجة أن وجود العوائق الديناميكية له تأثير كبير على وقت السفر الفعلي، وبالتالي فإن المسار الأقل تكلفة لا يؤدي دائماً إلى أسرع وقت. بالإضافة إلى ذلك، أظهرت نتائج اختبار قابلية الاستخدام أن اللعبة تتمتع بمستوى عالٍ من قابلية الفهم وسهولة التصفح وثقة المستخدم، بمتوسط ٥.١٢ درجة (GUESS) باستخدام نهج .توفر بنجاح تجربة لعب جيدة وفعالة في تحقيق أهدافها التعليمية (CoolCity) على مقياس من ٧. وهذا يشير إلى أن لعبة

BAB I

PENDAHULUAN

1.1 Latar Belakang

Industri *game* saat ini mengalami perkembangan yang sangat pesat dengan beragamnya genre yang ditawarkan. Menurut data di *Steam chart* diakses pada 28 November pukul 11.30*,* genre *game* simulasi merupakan salah satu genre yang diminati oleh *player* dimana *game* seperti *Farming Simulator 2024, Euro Truck simulator 2,* dan *Stardew Valley, masuk* dalam 100 besar *game* yang paling banyak dimainkan di tahun 2024 (Steam Chart, 2024). Simulasi merupakan suatu proses peniruan terhadap hal yang sebenarnya beserta dengan keadaan di sekitarnya (Faisal dkk., 2015). Hal ini mendukung pernyataan Irawan (2019) *game* simulasi merefleksikan situasi-situasi dari kehidupan nyata yang telah dimodifikasi menjadi lebih sederhana, diambil sebagian atau dikeluarkan dari konteksnya. Salah satu jenis tantangan *game* simulasi adalah *game* dengan tantangan multi objektif, yang mengharuskan pemain melakukan berbagai aksi untuk mencapai tujuan yang telah ditetapkan.

Tantangan untuk menyelesaikan skenario multi objektif berbentuk kunjungan kebeberapa tujuan dalam satu rute perjalanan yang optimal dikenal dengan TSP atau kepanjangan dari *Traveling Salesman Problem*. TSP adalah masalah optimasi kombinatorial terkenal yang pertama kali diajukan pada tahun 1930 serta merupakan masalah yang menarik bagi matematikawan dan ilmuwan komputer karena mudah diungkapkan tetapi sulit dipecahkan (Liao, 2023). Menurut Hayes

(2017) masalah optimasi secara umum dapat didefinisikan sebagai solusi matematis terbaik dari semua pencarian kemungkinan solusi yang ada. Dalam konteks game simulasi TSP memiliki relevansi yang kuat dalam menghadapi tantangan multi objektif. Salah satu penerapannya terlihat pada *city simulation game*, di mana pemain dihadapkan pada kebutuhan untuk menentukan rute cost optimal yang mempertimbangkan jarak dari berbagai titik yang harus dikunjungi dengan sumber daya terbatas.

Pemain akan mencoba berbagai skenario untuk dapat memenuhi tantangan tersebut dengan tantangan waktu. Namun, seringkali pemain mengalami kesulitan dalam menemukan rute optimal sehingga mengakibatkan ketidakefisienan dan mempengaruhi performa permainan. Isu ini merupakan masalah yang signifikan, karena ketidakefisienan dalam strategi tidak hanya berdampak pada pengalaman bermain, tetapi juga menurunkan tingkat kepuasan pemain (Fan dkk., 2024) .

Untuk membantu pemain dalam menyelesaikan pemilihan rute yang cost optimal, biasanya *game* memiliki *In-Game Assistance* untuk membantu pemain seperti *hint*, *tooltips*, dan sistem rekomendasi berbasis algoritma. *In-Game Assistance* merupakan layanan bantuan yang disediakan dalam *game*, sehingga memungkinkan pengguna mendapat bantuan secara langsung saat berinteraksi dengan *game*. Pada penelitian yang dilakukan oleh Leuchter dan Kurtz, disampaikan bahwa *In-Game Assistance* meningkatkan performa pemain (Leuchter & Kurtz, 2022). *In-Game Assistance* yang berbentuk algortima merupakan bantuan yang lebih baik dibandingkan dengan bantuan lainya karena penggunaan sistem

rekomendasi algoritma akan lebih terfokus pada masalah - masalah pemilihan rute(Burnes & Siman, 2024).

Algoritma TSP dirancang untuk membantu menyelesaikan masalah optimasi rute dengan menemukan *cost* terendah untuk mengunjungi beberapa tujuan. Dalam penelitian ini, pendekatan yang digunakan menggabungkan dua algoritma, yaitu *Brute Force* dan A*. Algoritma *Brute Force* akan digunakan untuk menyelesaikan aspek multi-objektif dalam TSP untuk menentukan urutan dengan *cost* terkecil pada setiap kunjungan. Sedangkan, Algoritma A∗ diterapkan untuk menghitung jarak antar titik dalam setiap kemungkinan rute secara *cost* optimal sehingga menemukan jalur terpendek. Algoritma A* merupakan algoritma yang dapat menemukan jalur terpendek secara efektif, sehingga dapat menyelesaikan banyak permasalahan pencarian pada jaringan jalan statis (Yan & Dublin, 2023). Dengan menerapkan kedua algoritma ini, *game* simulasi diharapkan dapat meningkatkan performa pemain sehingga memberikan pengalaman yang lebih baik dan optimal dalam pemilihan jalur.

Pada QS Al-'Asr: 1-3:

وَالْعَصْرِ إِنَّ الإِنْسَانَ لَفِي خُسْرٍ إِلا الَّذِينَ آمَنُوا وَعَمِلُوا الصَّالِحَاتِ وَتَوَاصَوْا بِالْحَقِّ وَتَوَاصَوْا بِالصَّبْرِ

"*Demi masa. Sesungguhnya manusia itu benar-benar berada dalam kerugian, kecuali orang-orang yang beriman dan mengerjakan amal saleh dan saling menasihati untuk kebenaran dan kesabaran*." *(QS. Al-'Asr: 1-3).*

Ayat ini menekankan betapa berharganya waktu, dan bagaimana manusia yang tidak memanfaatkannya dengan baik akan berada dalam kerugian. Sama halnya dalam *game* simulasi, pemain dituntut untuk mengelola sumber daya dan waktu dengan bijaksana untuk mencapai tujuan yang diinginkan. Ketika pemain

gagal merencanakan strategi yang efisien, mereka akan mengalami kerugian, baik dari segi performa permainan maupun tingkat kepuasan. Seperti yang dijelaskan dalam Surah Al-Asr, orang yang tidak menggunakan waktu dengan baik akan kehilangan kesempatan untuk meraih hasil yang optimal.

Pada QS Al-Baqarah ayat 269:

يُؤْتِى الْحِكْمَةَ مَنْ يَّشَاۤءُۚ وَمَنْ يُّؤْتَ الْحِكْمَةَ فَقَدْ اُوْتِيَ خَيْرًا كَثِيْرًاۗ وَمَا يَذَّكَّرُ اِلَّآ اُولُوا الْاَلْبَابِ

"Dia (Allah) menganugerahkan hikmah kepada siapa yang Dia kehendaki. Siapa yang dianugerahi hikmah, sungguh dia telah dianugerahi kebaikan yang banyak. Tidak ada yang dapat mengambil pelajaran (darinya), kecuali ululalbab." (QS. Al-Baqarah: 269).

Pengembangan sistem In-Game Assistance berbasis algoritma mencerminkan upaya menghadirkan kebijaksanaan teknologi dalam membantu pemain membuat keputusan yang lebih tepat, efisien, dan cerdas. Pemanfaatan ilmu pengetahuan sebagai bentuk hikmah yang diberikan Allah dapat mendorong kemajuan teknologi dan peningkatan kualitas pengalaman bermain.

Penerapan TSP menggunakan Algoritma *Brute Force* dan A* dalam *game* simulasi bertujuan untuk membantu pemain mengoptimalkan waktu dan sumber daya mereka, sehingga pengalaman bermain menjadi lebih baik dan memberikan manfaat edukatif. Hal ini sejalan dengan pesan dalam QS Al-'Asr mengenai pentingnya memanfaatkan waktu secara maksimal, serta QS Al-Baqarah ayat 269 yang menekankan pentingnya hikmah (kebijaksanaan) dalam bertindak. Dengan demikian, pengembangan teknologi berbasis algoritma tidak hanya mencerminkan kemajuan ilmu pengetahuan, tetapi juga menjadi bentuk aktualisasi nilai-nilai Islam dalam kehidupan modern.

1.2 Rumusan Masalah

Bagaimana mendapatkan rute dengan *cost* optimal yang mempertimbangkan jarak terpendek, dalam menyelesakan tantangan pada *game* simulasi berbasis multi objektif?

1.3 Batasan Masalah

Batasan masalah yang penulis kemukakan dalam pengimplementasian algoritma ini adalah:

1. Jumlah tujuan yang digunakan untuk permasalahan TSP tidak lebih dari 5 titik.
2. Penelitian hanya dilakukan pada *game* cool city.
3. Karakter hanya bergerak pada sumbu x dan z secara bergantian.

1.4 Tujuan Penelitian

Tujuan penelitian ini adalah menerapkan Algoritma *Brute Force* dan A* dalam *game* simulasi berbasis TSP untuk mendapatkan rute dengan *cost* optimal pada *game* simulasi berbasis multi objektif.

1.5 Manfaat Penelitian

Manfaat penelitian dari penelitian ini adalah sebagai berikut:

1. Memberikan alternatif edukasi optimasi rute manajemen sumber daya dengan *game* simulasi.
2. Mengembangkan sarana edukasi bagi setiap orang mengenai optimasi rute untuk manajemen sumberdaya.

BAB II

STUDI PUSTAKA

2.1 Penelitian Terkait

Pathfinding adalah algoritma yang digunakan dalam berbagai bidang seperti robot, *game*, dan transportasi. Seperti yang dilakukan Hermawan dan Setiyani melakukan implementasi algoritma A* dalam subgenre *game roguelike*, yang memfokuskan pada pencarian jalur terpendek dalam proses pengejaran musuh terhadap karakter pemain. Hal ini menunjukan bahwa A* merupakan salah satu metode yang tepat untuk pencarian terpendek dalam proses pengejaran musuh terhadap karakter pemain (Hermawan & Setiyani, 2019).

Penelitian oleh Zuhdi juga mengimpelementasikan algoritma A* dalam *game Great Elephant*. Algoritma A* digunakan untuk menemukan jalur terpendek di dalam labirin guna meningkatkan efisiensi waktu dalam mencari makanan. Hasil penelitian yang diuji aspek fungsionalitasnya menggunakan ISO 25010 dan menunjukkan tingkat keberhasilan 100% (Zuhdi dkk., 2023).

Selain itu, penelitian oleh Iskandar melakukan implementasi algoritma - algoritma *Pathfinding* dan menemukan bahwa A* adalah algoritma yang paling cocok untuk *game platformer*. Hasil dari penelitian ini menunjukan jika waktu eksekusi, jumlah *node*, dan jarak tempuh serta jumlah *node* yang dihitung oleh algoritma A* lebih efisien dari algoritma lain (Iskandar dkk., 2020).

Selain algoritma *Pathfinding* terdapat masalah lain yang menyangkut dengan waktu dan keefisienan yaitu TSP. TSP adalah suatu masalah dengan tujuan untuk

menemukan rute terpendek dari sejumlah tujuan yang nantinya akan dikunjungi tanpa harus melewati lokasi yang sudah dikunjungi. Penelitian oleh Wawan pada studi kasus JNE: Balige menghasilkan solusi optimum dari multi objektif sejumlah 6 titik dengan waktu tempuh 36 menit lebih efisien 25% dari yang tidak menggunakan TSP yakni 48 menit lokasi (Wawan, 2022).

Penelitian oleh Samah menunjukkan implementasi algoritma TSP berupa *Brute Force* pada sistem rekomendasi perjalanan yang membantu wisatawan merencanakan rute optimal dengan mempertimbangkan waktu tempuh dan estimasi waktu kedatangan. Dengan menggunakan konsep Traveling Salesman Problem (TSP), Algoritma *Brute Force* berhasil menentukan urutan tempat dengan durasi waktu terpendek, serta terbukti *reliable* melalui *reliability testing* dengan hasil yang dapat diterima (t-test = 0,00067) (Samah dkk., 2019).

Tabel 2. 1 Penelitian Terdahulu

No	**Judul**	***Pathfin-ding***	**TSP**	***Subject***	**Hasil Penelitian**
1.	Implementasi Algoritma A* Pada Permainan Komputer Roguelike Berbasis Unity. (2019)	Ya (A*)	Tidak	Pengejaran karakter musuh terhadap karakter pemain.	Hasil penelitian ini menunjukan bahwa A* merupakan salah satu metode yang tepat untuk pencarian terpendek dalam proses pengejaran musuh terhadap karakter pemain.
2.	Implementation Of A* Algorithm In A Great Elephant *Game* With Unity 2D. (2023)	Ya (A*)	Tidak	Pergerakan karakter gajah pada maze untuk makanan	Hasil penelitian yang diuji menggunakan ISO 25010 pada aspek fungsi menunjukkan tingkat keberhasilan 100%.
3.	Identifying Artificial Intelligence Pathfinding Algorithms for	Ya (A*, BFS, Djikstra, DFS)	Tidak	Pergerakan karakter pada *game* platformer	Hasil dari penelitian ini menunjukan jika waktu eksekusi, jumlah *node*, dan jarak tempuh serta jumlah *node* yang

No	Judul	*Pathfin-ding*	TSP	*Subject*	Hasil Penelitian
	Platformer Games. (2020)				dihitung oleh algoritma A* lebih efisien dari algoritma lain.
4.	Optimalisasi Rute Distribusi Kurir Menggunakan Metode Traveling Salesman Problem (Studi Kasus: JNE Balige) (2022)	Tidak	Ya (Branch and Bound)	Pergerakan kurir dalam distribusi	Solusi 6 titik dengan waktu 25% lebih cepat.
5.	Brute Force algorithm implementation for traveljoy travelling recommendation system	Tidak	Ya (*Brute Force*)	Sistem rekomendasi Travel Joy	*Brute Force* berhasil menentukan urutan tempat dengan durasi waktu terpendek, serta terbukti *reliable* melalui *reliability testing* dengan hasil yang dapat diterima (t-test = 0,00067)
6.	Optimasi Rute Menggunakan Algoritma A* dan Brute Force Pada Game Simulasi TSP	Ya (A*)	Ya (*Brute Force*)	*Game* Cool City	

2.2 *Game*

Game didefinisikan sebagai bentuk keahlian di mana pemain membuat keputusan untuk mengelola sumber daya dalam permainan guna mencapai tujuan tertentu (Sehang dkk., 2019).

Menurut (Bahtiar dkk., 2019) ada 8 macam *basic genre game*, antara lain:

1. *Action*

   Action Game adalah genre permainan dimana pemain dapat berinteraksi dengan *environment* secara langsung untuk melakukan aksi seperti menembak, menghindar, menebas, melompat atau aksi lainnya. Biasanya

game dengan genre ini memerlukan kecepatan dan ketangkasan dalam melakukan aksi. Contoh dari *gamenya* adalah *game Valorant, Counter Strike, dan Tekken*

2. *Adventure*

   Adventure Game adalah *game* yang berfokus pada eksplorasi dan pemecahan teka - teki. Pada *game* bergenre *adventure* sering dijumpai *environment* yang sangat luas dan dapat berinteraksi dengan *environment.* Genre ini seringkali melibatkan pengumpulan key item untuk pemecahan teka - teki. Contoh dari *gamenya* adalah *Sekiro, dan Black Myth Wukong.*

3. *Educational*

   Game edukasi adalah *game* yang khusus dirancang untuk mengajarkan user suatu pembelajaran tertentu, pengembangan konsep dan pemahaman dan membimbing mereka dalam melatih kemampuan mereka, serta memotivasi mereka untuk memainkannya (Diharjo dkk., 2020). *Game* ini biasanya menggabungkan elemen permainan dengan konten pendidikan untuk meningkatkan pengalaman belajar. Contoh dari *gamenya* adalah *Kahoot, dan Duolingo.*

4. *Racing*

   *Racing Game a*dalah genre permainan yang menantang pemain untuk berlomba melawan waktu atau lawan lain menggunakan kendaraan. Dalam *game* ini, keterampilan mengemudi dan strategi balapan sangat penting, dengan berbagai lintasan dan jenis kendaraan untuk dipilih. Contoh *gamenya* adalah *Need for Speed, dan Forza Horizon, The Crew 2.*

5. *Role Playing*

   RPG adalah genre *game* dimana pemain diharuskan untuk mengembangkan karakter yang dimiliki dengan mendapatkan EXP, Item, dan Uang dari eksplorasi dunia atau menyelesaikan quest. Biasanya dalam *game* RPG pemain dapat menyelesaikan banyak quest yang tersebar di dunia *game* tersebut. contoh *gamenya* adalah *Elden Ring, dan Genshin Impact.*

6. *Simulation*

   Simulation game adalah *game* yang merefleksikan situasi-situasi dari kehidupan nyata yang telah dimodifikasi menjadi lebih sederhana, diambil sebagian atau dikeluarkan dari konteksnya. Genre ini berusaha menciptakan pengalaman realistis dari berbagai aspek kehidupan atau aktivitas. Contoh *gamenya* adalah *Farming Simulator, dan Top Eleven Football Manager.*

7. *Sports*

   Sport game adalah genre *game* yang meniru berbagai cabang olahraga di dunia asli. Genre ini biasanya berfokus pada strategi, dan ketangkasan bermain sehingga memberikan pengalaman bermain yang menegangkan seperti aslinya. Contoh *gamenya* adalah *FIFA 2024, dan NBA2K24.*

8. *Strategy*

   Genre s*trategy* adalah *game* tentang perencanaan dan pengambilan keputusan. Dalam *game* ini, pemain dituntut untuk menggerakkan unit atau mengelola sumber daya untuk mencapai tujuan tertentu. Pemain harus berpikir untuk melakukan perencanaan untuk tujuan mereka baik bertahan

hidup, atau menyerang lawan. Contoh *gamenya* adalah *Clash of Clans, dan Chess.*

2.3 *Finite State Machine* (FSM)

Menurut Jagdale *Finite-State Machine* (FSM) merupakan salah satu teknik tertua dalam permainan yang digunakan dalam permainan lama seperti PACMAN dan permainan baru seperti TOMB RAIDER (Jagdale, 2021). Dalam permainan tujuan utama dari FSM adalah membuat NPC lebih cerdas. Secara luas FSM digunakan dalam pengembangan perangkat lunak, khususnya dalam *game* dan aplikasi interaktif, untuk mengatur transisi antar keadaan berdasarkan *input* atau kejadian tertentu. Dalam konsep dasar FSM, terdapat beberapa elemen penting, yaitu state, *event*, dan action. *State* merupakan kondisi atau keadaan di mana sistem berada pada waktu tertentu, yang dalam konteks *game* dapat mencakup keadaan seperti "Idle" (diam), "Attacking" (menyerang), atau "Dead" (mati). Sementara itu, *event* adalah *input* atau sinyal yang memicu perubahan dari satu *state* ke *state* lainnya. Sebagai contoh, dalam *game*, *event* bisa berupa perintah dari pemain atau perubahan kondisi lingkungan, yang dapat mempengaruhi perilaku karakter dalam *game*. *Action* adalah tindakan yang diambil oleh sistem sebagai respons terhadap *event* yang terjadi, seperti animasi karakter, perubahan posisi, atau interaksi dengan objek lain.

FSM telah terbukti efektif dalam mengatur perilaku Non-*Play*able Characters (NPC) di dalam *game*, yang memungkinkan NPC untuk beradaptasi dan merespons situasi yang dihadapi oleh pemain secara dinamis. Hal tersebut didukung oleh penelitian Satrio dkk, (2022) yang menerapkan FSM dalam *game* "Lost

Civilization", dimana FSM digunakan untuk mengontrol perilaku musuh dengan mendefinisikan state, *event*, dan action, sehingga NPC dapat merespons situasi dalam *game* secara lebih realistis Satrio dkk, (2022). FSM juga merupakan metode yang kuat untuk mendesain skenario game (Arif dkk., 2023). Pada *game* jenis Tower Defense, FSM membantu menentukan tindakan NPC saat diserang, yang membuat skenario game semakin menarik (Hasan Syu'aibi dkk., 2023). Selain itu, dalam *game* RPG "Kisah Tjepoe", FSM digunakan untuk meningkatkan variasi perilaku NPC, sehingga alur cerita menjadi lebih dinamis dan menarik (Nizhar, dkk., 2023).

2.4 *Traveling Salesman Problem*

Traveling Salesman Problem (TSP) adalah sebuah masalah dimana seorang salesman harus mengunjungi semua titik tujuan yang telah ditentukan dan kembali ke titik awal, dengan syarat bahwa setiap titik hanya dikunjungi sekali (Paillin & Tupan, 2020). TSP banyak digunakan dalam bidang logistik, distribusi, dan penjadwalan rute perjalanan, sehingga penting untuk menemukan solusi yang optimal untuk mengurangi biaya dan waktu perjalanan. Menurut (Osaba, dkk., 2020) terdapat beberapa pendekatan untuk mengatasi masalah TSP ini antara lain:

1. *Exact Solution Approaches* (pendekatan eksak)

   Penyelesaian TSP dengan pendekatan eksak akan selalu mendapatkan hasil dengan tipe yang optimal pada permasalahan yang diberikan. Pendekatan ini menjamin solusi optimal dengan *cost* yaitu *resource* komputasi yang intens dan waktu yang lama. Menurut Wawan (2022) metode ini dibagi menjadi dua bagian yakni:

a. *Brute Force Method*

Metode ini melibatkan menghitung dan membandingkan semua kemungkinan permutasi rute.

b. *Branch and Bound*

Algoritma ini bekerja secara rekursif dengan cara membagi masalah menjadi bagian-bagian kecil dan mencatat biaya minimum untuk memperkirakan nilai optimal (Triyanto dkk., 2015).

2. *Approximate approaches* (pendekatan perkiraan)

Penyelesaian TSP untuk ukuran yang besar memerlukan banyak *resource* komputasi dan waktu, maka dari itu masih ada ruang untuk pengembangan dan penerapan solusi *approximate*, atau heuristik. Pendekatan ini tidak menjamin solusi optimal, akan tetapi memberikan solusi yang mendekati optimal dengan kebutuhan *resource* komputasi yang wajar.

2.5 *Pathfinding*

Pathfinding adalah metode yang terkenal dan telah digunakan sejak lama mulai dari *game*, logistik, dan robot *autonomous*. Tujuan dari *Pathfinding* adalah untuk mendapatkan rute yang optimum atau mendekati optimum untuk entitas yang ingin di otomasi untuk mencapai tujuan (Kherrour, dkk., 2023). *Pathfinding* biasanya ditemukan pada pergerakan NPC (Non-*Player* Character) seperti pada *shooter game* (Hasniati dkk., 2023), dan *game adventure* lainnya untuk NPC mengejar *player* (Hendy, 2021). Namun *Pathfinding* juga bisa ditemukan pada *play*able *character* seperti pada *game* Dota 2 dan League Of Legends (Huu Le,

2018). Menurut penelitian (Kherrour, dkk., 2023) Ada empat algoritma *Pathfinding* yang terkenal:

1. Algoritma A* (A Star) dikenal sebagai salah satu algoritma yang paling sering digunakan untuk pencarian jalur (*pathfinding*) dan penerusan grafis (*graph traversal*), yaitu proses ploting jalur yang paling efisien antar titik, yang disebut dengan *node*s (Syukriyah, dkk., 2016) Metode A* adalah metode yang merupakan hasil pengembangan dari metode dasar *Best First Search* (BFS).
2. Theta* adalah salah satu algoritma multi sudut *Pathfinding*. Algoritma ini dikmebangkan berdasarkan algoritma pendahulunya yaitu A*. A* bisa menemukan jalur terpendek akan tetapi tidak jalur terpendek sebenarnya. Theta* tidak terbatas untuk masalah ini, karena algoritma ini mampu menyimpang dari jalur grid memungkinkan untuk menemukan jalur yang lebih pendek dan tampak lebih alami (Krafft, 2019).
3. *Dijkstra* merupakan algoritma routing jalur terpendek klasik yang terkenal dalam penelitian perencanaan jalur robot bergerak 2D. Algoritma ini merupakan algoritma sederhana untuk masalah jalur terpendek sumber tunggal, yang secara efektif dapat menghitung jalur terpendek ke semua tujuan. Pada intinya adalah algoritma yang menangani jarak terpendek antara dua titik yang ada pada graf (Luo dkk., 2016).
4. *Breadth-First Search* adalah sebuah algoritma graph traversal yang mendasar. BFS mulai dari titik root dan mengeksplor semua *node* pada kedalaman ini sebelum melanjutkan ke kedalaman selanjutnya. Langkah ini kemudian

diulang - ulang hingga menemukan yang paling optimal (Indriyono & Widyatmoko, 2021).

2.6 Algoritma *Brute Force*

Salah satu dasar algoritma yang digunakan di pencarian adalah *Brute Force*. *Brute Force* melakukan pemilihan untuk *cost* perjalanan yang paling murah dengan cara mengkomparasi berbagai hasil dari permutasi (kemungkinan) yang telah dieksplorasi. Hasil dari algoritma ini pasti optimal, tapi kurang efektif jika terdapat titik lebih dari 10, karena kita harus mencari permutasi dari 10 titik tersebut (Amelia dkk., 2024). Algoritma *Brute Force* dapat digunakan dalam skala kecil karena kompleksitas dari algoritma ini adalah O(n!) (Baidoo & Oppong, 2016). Berikut ini adalah tahapan untuk menggunakan pendekatan *Brute Force* untuk menyelesaikan TSP:

1. Menghitung total dari permutasi dari seluruh *node* yang ada dengan (n!) di mana n merepresentasikan sebuah *node*.

$$P = (n-1)! \tag{2.1}$$

Keterangan:
P adalah permutasi.
n adalah *node*.

2. Menghitung jarak untuk seluruh permutasi menggunakan heuristic.
3. Bandingkan jarak setiap jalur.
4. Memilih jalur dengan jarak atau *cost* yang terkecil.

2.7 Algoritma A*

Algoritma A* (A Star) dikenal sebagai salah satu algoritma yang kompeten, terutama pada kemampuannya untuk memberikan jalur optimal atau jalur yang mendekati optimal *cost* terendah (Brown Requist & Momayez, 2024). Algoritma A* adalah algoritma yang merupakan hasil pengembangan dari algoritma dasar *Best First Search* (BFS). Seperti halnya pada BFS, untuk menemukan solusi, A* juga dituntun oleh fungsi heuristik, yang menentukan urutan titik mana yang akan dikunjungi terlebih dahulu. Heuristik merupakan penilai yang memberi harga pada tiap titik yang memandu A* mendapatkan solusi yang diinginkan (Ayu Nugraeni dkk., 2015). Heuristik dibagi menjadi dua:

1. *Euclidean Distance*:

   Euclidean Distance adalah metode perhitungan jarak antara dua titik dalam ruang *Euclidean* yang diperkenalkan oleh matematikawan Yunani bernama Euclid. Menurut pendapat Setiawan dalam (Susilawati, dkk., 2020) metode ini sering digunakan sebagai fungsi heuristik dalam berbagai aplikasi. Hubungannya dengan Teorema Pythagoras sangat signifikan. *Euclidean Distance* Digunakan ketika pergerakan bisa dilakukan dalam segala arah, termasuk diagonal, seperti dalam ruang 2D atau 3D di mana tidak ada pembatas *grid* yang kaku.

$$d = \sqrt{(x_2 - x_1)^2 + (y_2 + y_1)^2} \tag{2.2}$$

Keterangan:
d adalah jarak euclidean
x_1 adalah titik awal
x_2 adalah titik tujuan
y_1 adalah titik awal
y_2 adalah titik tujuan

2. *Manhattan Distance*

Manhattan Distance, juga dikenal sebagai "City Block Distance," mengukur jumlah jarak antara dua titik dengan cara menjumlahkan selisih absolut antara setiap atribut pada kedua titik tersebut. Menurut pendapat Januardi dalam (Shafira, dkk., 2023) teknik ini sering digunakan untuk menentukan kesamaan atau perbedaan antara dua objek. Pengukurannya didasarkan pada perhitungan jarak vertikal dan horizontal antara dua titik. *Manhattan Distance* digunakan ketika pergerakan dibatasi oleh *grid* yang hanya mengizinkan pergerakan horizontal atau vertikal, seperti pada peta kotak-kotak atau lingkungan dengan jalan-jalan berbentuk persegi.

$$d = |x_2 - x_1| + |y_2 + y_1| \quad (2.3)$$

Keterangan:
d adalah jarak euclidean
x_1 adalah titik awal
x_2 adalah titik tujuan
y_1 adalah titik awal
y_2 adalah titik tujuan

Kedua rumus heuristik bisa diterapkan dalam algoritma A* dalam konteks *game*. Penggunaan heuristik disesuaikan dengan konteks *game* yang akan dibuat. Jika *game* yang dibuat merupakan *game* berbasis *grid* dan pergerakan terbatas pada sumbu x dan y maka heuristik *manhattan* lebih baik daripada *Euclidean.* Sedangkan *Euclidean* baik digunakan pada *game* yang tidak memiliki batasan *grid*, dan pergerakan bisa dilakukan dalam segala arah. Pada penerapannya, Algoritma A* dapat diimplementasikan dengan mengikuti rumus yang dijelaskan pada bagian 2.3.

$$f(n) = g(n) + h(n) \quad (2.4)$$

Keterangan:
$f(n)$ adalah total jarak dari titik asli dan jarak heuristiknya
$g(n)$ adalah harga dari titik awal ke titik yang ingin dituju
$h(n)$ adalah jarak heuristik

Langkah - langkah untuk menyelesaikan perhitungan A* adalah dengan menggunakan konsep *Open list* dan *Closed list.*

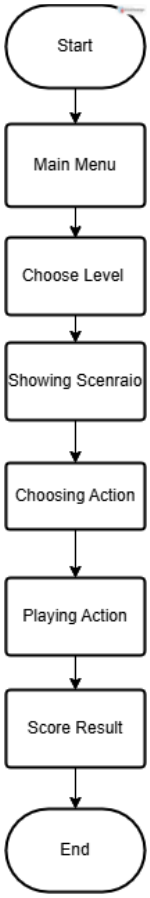
1. Masukkan *node* awal ke dalam *Closed list.*
2. Masukan tetangga dari *node* awal kedalam *Open list.*
3. Hitung nilai $f(n)$ yang terdapat pada *Open list*.
4. Ambil *node* dengan nilai $f(n)$ terkecil dari *Open list*.
5. Pindahkan *node* yang sedang dievaluasi dari *Open list* ke *Closed list.*
6. Ulangi langkah $2-5$ sampai *node* tujuan ditemukan

BAB III

METODE PENELITIAN

3.1 Objek Penelitian

Objek *game* yang digunakan pada penelitian ini adalah *game* berjudul *CoolCity*. *CoolCity* adalah *game* edukasi simulasi berbasis *puzzle* tentang manajemen kota. *Game* ini dirancang untuk platform PC (*Personal Computer*) yang ditujukan untuk semua umur. Pada *game CoolCity* pemain akan diberikan peran sebagai *stakeholder* yang bertugas untuk menurunkan suhu dengan mengelola kota melalui aksi pembangunan atau penataan banganunan.



Gambar 3. 1 *Flowchart Core Game*play *Loop*

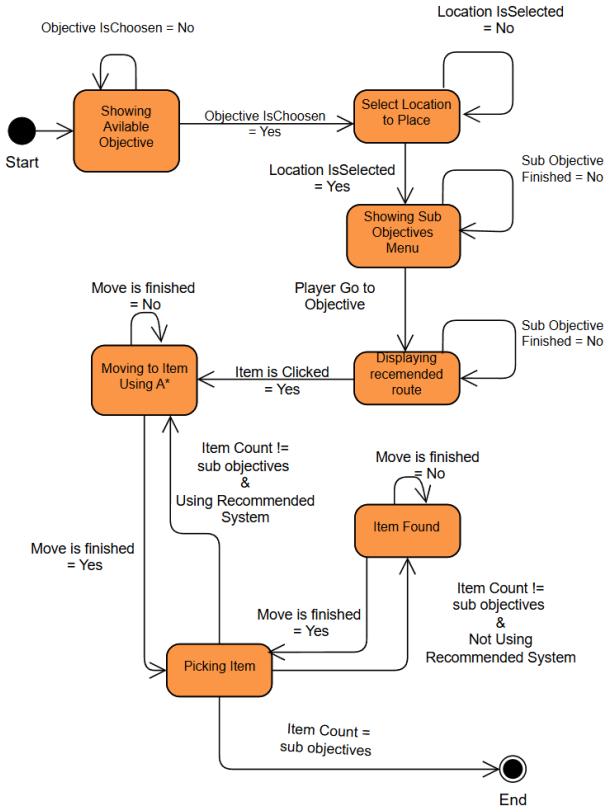
Berdasarkan *Flowchart* pada Gambar 3.1 dapat dijelaskan *core gameplay loop* dari *CoolCity* diawali dengan tampilan dasar yaitu tampilan menu yang memiliki tiga pilihan yaitu *play*, *option*, dan *exit*. Kemudian apabila pemain memilih menu *option* akan ditampilkan beberapa pilihan untuk mengatur pengaturan *game* dan tombol untuk kembali ke menu utama. Akan tetapi, saat pemain memilih menu *play*, maka akan ditampilkan menu dari skenario (*stage*) dan pemain dapat memilih skenario atau kembali ke menu utama. Setelah disajikan dua pilihan tersebut dan pemain memilih menu skenario, pemain akan ditampilkan skenario dari *game*. Saat permainan berlangsung pemain dapat menyelesaikan permainan tersebut dengan cara melakukan berbagai aksi yang disediakan dan pemain dapat memperoleh skor saat permainan telah berakhir. Pada menu *game over* pemain dapat memilih *next level* untuk pergi ke menu *play* awal dan pemain juga dapat memilih keluar dari serta kembali ke menu utama.

Gambar 3. 2 Ilustrasi *Game*

Dapat dilihat pada Gambar 3.2 merupakan ilustrasi *game CoolCity*, pemain diharuskan untuk mengelola berbagai sumber daya seperti energi dan waktu. Saat

pemain melakukan pembangunan diperlukan berbagai material dari toko yang berbeda di seluruh penjuru kota hal ini dapat menjadi suatu permasalahan karena adanya ketidakefisienan dalam memilih jalur sehingga dapat mempengaruhi skor akhir. Masalah tersebut merupakan TSP yang dapat diselesaikan dengan pendekatan Algoritma *Brute Force*, selain itu untuk lebih mengefisienkan perencanaan jalur dapat digunakan algoritma A*.

3.2 *Finite State Machine* (FSM)



Gambar 3. 3 FSM

FSM digunakan untuk mengelola perilaku dan interaksi berbagai elemen dalam permainan berpindah dari satu *state* ke *state* lain berdasarkan *input* atau kondisi tertentu. Gambar 3.3 menggambarkan alur interaksi pemain dengan sistem

saat di dalam permainan. Skenario dimulai dengan menampilkan objektif utama yang dapat dipilih oleh pemain untuk dibangun, ketika pemain memilih objektif utama sistem akan memberikan instruksi untuk menentukan pembangunan. Ketika lokasi telah ditentukan, sistem akan menyediakan rute optimal yang memungkinkan pemain bergerak menuju item menggunakan algoritma A* untuk mengambil item tersebut. Proses ini akan berulang hingga jumlah item yang dikumpulkan sama dengan jumlah sub-tujuan (subobjektif) yang tersedia.

3.3 Rancangan Pengujian

Tujuan pengujian adalah untuk mengevaluasi keefektifan algoritma A* dalam menemukan jalur optimal pada peta permainan, menentukan efisiensi penyelesaian masalah TSP menggunakan Algoritma *Brute Force*, serta mengevaluasi tingkat kepuasan pengguna terhadap *game* secara keseluruhan, termasuk antarmuka, navigasi, dan performa.

3.4 Skenario Pengujian

Pengujian akan dilakukan dengan beberapa skenario dalam beberapa frekuensi untuk menguji performa algoritma pada berbagai tujuan yang harus dicapai Tabel 3. 1. Setiap skenario memiliki alur kunjungan yang berbeda untuk menguji keefektifan dan efisiensi algoritma yang digunakan.

3.4.1 Pengujian Algoritma Brute Force

Tabel 3. 1 Skenario *Game*

Skenario	Jumlah Tujuan	Frekuensi Pengujian
Skenario 1	1	3
Skenario 2	2	3
Skenario 3	3	3

Skenario	Jumlah Tujuan	Frekuensi Pengujian
Skenario 4	4	3
Skenario 5	5	3

Pengujian dilakukan menggunakan metode *blackbox* yaitu pengujian yang memeriksa fungsionalitas sistem tanpa mengintip ke dalam struktur (Nugroho dkk., 2022). Dengan memberikan masukan berupa skenario yang berbeda dan jumlah tujuan yang bervariasi. Saat karakter melakukan kliking pada objektif utama system akan menampilkan multi objektif yang harus diselesaikan dengan cara mengunjungi objektif tersebut. pengujian dilakukan dengan cara:

1) Melakukan perhitungan terhadap objektif TSP.
2) Memberikan input ke sistem dan membandingkan *output* dengan hasil perhitungan.

3.4.2 Pengujian Algoritma A*

Karakter harus menemukan jalur tercepat dari titik awal ke tujuan dalam peta yang memiliki ukuran 150 x 150 *node*. Kemudian pengujian dilakukan dengan cara:

1) Karakter memulai dari posisi awal yang telah ditentukan.
2) Algoritma A* dijalankan untuk mencari jalur optimal menuju tujuan.
3) Hasil waktu perjalanan, panjang jalur, dan kecepatan karakter.
4) Hasil akan dibandingkan dengan waktu ideal yang didapatkan dari persamaan:

$$s = v \times t \tag{3.1}$$

Kemudian dari persamaan tersebut didapatkan persamaan untuk mencari waktu ideal :

$$t = \frac{s}{v} \tag{3.2}$$

Keterangan:
s : Jarak
v : Kecepatan
t: Waktu

Pengujian dilakukan dengan mengambil salah satu hasil rekomendasi jalur yang diperoleh melalui Algoritma *Brute Force*. Setiap segmen jarak antar titik, seperti titik A–B, B–C, C–D, dan D–A, akan diuji sebanyak tiga kali pengujian yang tertera pada Tabel 3. 2.

Tabel 3. 2 Tabel Pengujian A*

Uji	Var 1	Var 2	Var 3	Var 4
1	Jarak	Waktu	*No Obstacle*	*No Obstacle*
2	Jarak	Waktu	*Static Obstacle*	*No Obstacle*
3	Jarak	Waktu	*Static Obstacle*	*Dynamic Obstacle*

3.4.3 Pengujian Game User Experience Satisfaction Scale (GUESS)

Pengujian dilakukan dengan melibatkan peserta yang belum pernah mencoba *game* sebelumnya untuk memastikan hasil yang objektif. Peserta diberikan waktu 20 menit untuk memainkan *game* tanpa panduan yang mendetail, sehingga mereka bebas mengeksplorasi fitur *game* secara mandiri. Setelah sesi bermain, peserta diminta untuk mengisi kuesioner GUESS yang terdiri dari 11 pernyataan yang berfokus pada aspek *usability* dinilai menggunakan skala Likert 1-7, di mana 1 menunjukkan "Sangat Tidak Setuju" dan 7 menunjukkan "Sangat Setuju." Data yang terkumpul dari kuesioner kemudian dianalisis untuk menghasilkan skor rata-rata pada setiap pernyataan, yang akan digunakan untuk mengidentifikasi kekuatan dan kelemahan *game*. Aspek dengan skor tinggi menunjukkan keunggulan *game*, sedangkan aspek dengan skor rendah akan menjadi fokus perbaikan.

Tabel 3. 3 Tabel kuisoner

Bagian	Kode	Pertanyaan
EOUS	Q1	I think it is easy to learn how to play the *game*
EOUS	Q2	I find the controls of the *game* to be straightforward.
EOUS	Q5	I do not need to go through a lengthy tutorial or read a manual to play the *game.*
NAV	Q3	I always know how to achieve my goals/objectives in the *game*.
NAV	Q4	I find the *game*'s interface to be easy to navigate
NAV	Q6	I find the *game*'s menus to be user friendly
NAV	Q7	I feel the *game* trains me well in all of the controls.
NAV	Q8	I always know my next goal when I finish an event in the *game*.
NAV	Q9	I feel the *game* provides me the necessary information to accomplish a goal within the *game*.
UNS	Q10	I think the information provided in the *game* (e.g., onscreen messages, help) is clear.
UNS	Q11	I feel very confident while playing the *game*.

Pertanyaan - pertanyaan diatas dapat dibagi menjadi beberapa bagian seperti *Ease of use* (EOUS), *Navigation* (NAV), dan yang terakhir adalah *Understanding* (UNS) (Phan dkk., 2016) . *Ease Of Use* (EOUS) mengacu pada Kemudahan Belajar dan Penggunaan awal *game* bagian ini dapat diinterpretasikan bahwa Pengguna merasa *game* ini mudah dipelajari secara intuitif, tanpa harus membaca instruksi rumit dan menandakan desain antarmuka dan tutorial awal efektif (Sattu & Fairul Filza, 2025). *Navigation* (NAV) mengacu pada Navigasi dan informasi yang ada dalam *game* bagian ini dapat diinterpretasikan bahwa Pengguna merasa bantuan visual, teks, dan navigasi sudah jelas. *Understanding* (UNS), mengacu pada sejauh mana pemain merasa paham dan percaya diri saat bermain. Berikut tabel klasifikasi untuk skor rata – rata dari pengisian kuisioner GUESS

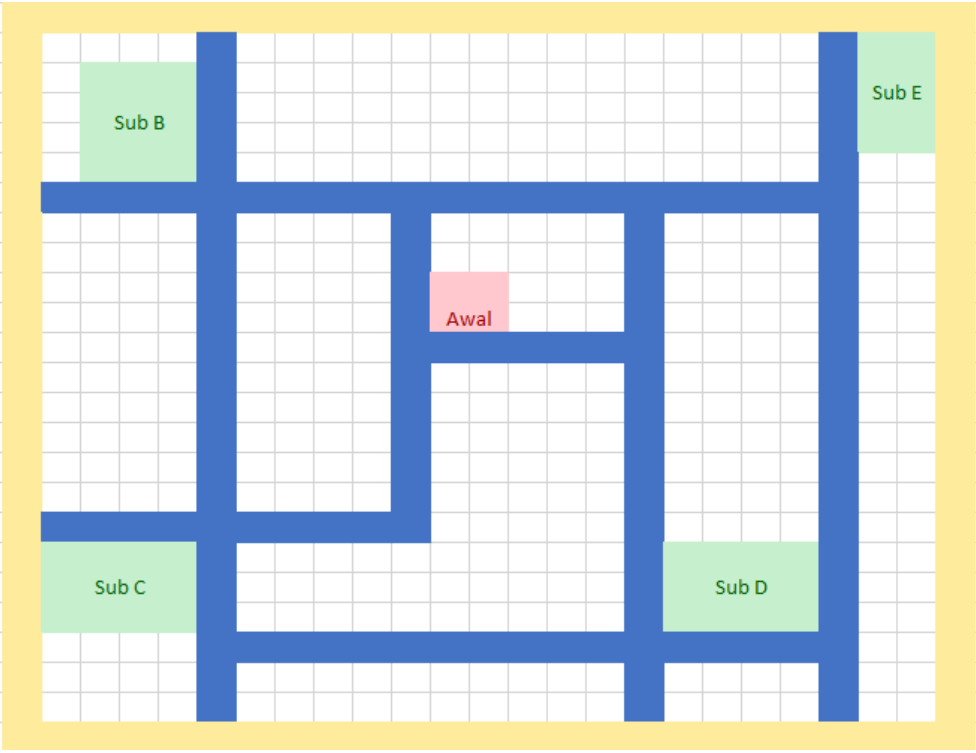
Tabel 3. 4 Klasifikasi Skor *Usability*

Skor	Klasifikasi	Hasil
85 – 100%	Sangat Baik	Berhasil
65 – 84 %	Baik	Berhasil
55 – 64 %	Cukup Baik	Tidak Berhasil
0 – 54%	Kurang Baik	Tidak Berhasil

(diadaptasi dari Danil, dkk (2024))

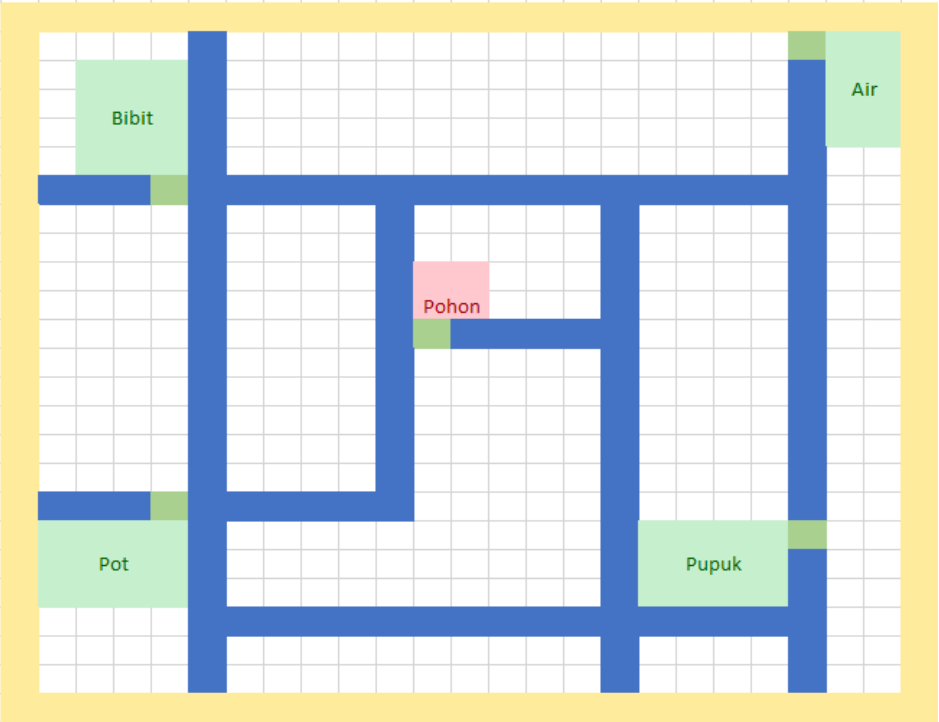
3.5 Studi Kasus

Seperti pada Gambar 3. 3 *State action* penanaman pohon besar dimulai dengan menampilkan objektif utama dimana *player* harus menggerakkan *character* ke tempat penanaman pohon besar. Ketika tujuan di klik oleh *player* algoritma A* akan mulai dijalankan oleh sistem. Saat algortima A* telah selesai di kalkulasi *character* akan bergerak ke tempat penanaman pohon. Ketika character sampai pada tempat penanaman pohon maka sub objektif akan muncul di beberapa tempat seperti toko - toko lain.



Gambar 3. 4 Tatanan Skenario *Game*

3.6 Permasalahan TSP



Gambar 3. 5 Skenario objektif *game*

Pada Gambar 3. 5 terdapat contoh kasus skenario *game CoolCity* dengan lima titik yang memiliki dua warna yang berbeda yaitu merah dan hijau. Untuk titik merah sebagai titik awal sekaligus sebagai objektif utama. Sedangkan untuk titik yang berwarna hijau sebagai sub-objektif. Pada *game CoolCity* akan diilustrasikan *action* saat penanaman pohon akan ditandai dengan blok titik berwarna merah. Kemudian untuk blok dengan titik berwarna hijau dalam hal ini akan mengilustrasikan beberapa toko misalnya untuk toko bibit sebagai Sub B, lalu untuk toko pot sebagai Sub C, toko pupuk sebagai Sub D, dan yang terakhir yaitu toko air sebagai Sub E.

Langkah - langkah perhitungan untuk menyelesaikan *problem* di atas diawali dengan:

a. Hitung permutasi atau kemungkinan yang mungkin muncul dengan rumus 2.4

$$P = (n-1)!$$

$$P = (5-1)!$$

$$P = 4!$$

$$P = 24$$

dapat disimpulkan dengan bahwa total jumlah permutasi adalah 24.

b. Hitung jarak setiap permutasi

Karena gerakan yang valid hanyalah vertikal dan horizontal, maka digunakan heuristik *Manhattan* untuk menghitung jarak antar titik. Rumus yang digunakan adalah rumus 2.2

Diketahui koordinat titik-titik adalah sebagai berikut:

Tabel 3. 5 Tabel Detail Objektif

Kode	Nama Objektif	Koordinat
A	Pohon besar	(12, 15)
B	Toko bibit	(4, 19)
C	Toko Pot	(4, 6)
D	Toko Pupuk	(20, 6)
E	Toko Air	(23, 23)

Perhitungan jarak antara A dan B dengan rumus *Manhattan*:

$$d = |x_2 - x_1| + |y_2 - y_1|$$

$$d_{ab} = |4 - 12| + |19 - 15|$$

$$d_{ab} = |8| + |4|$$

$$d_{ab} = 12$$

Dilakukan perhitungan yang sama untuk seluruh pasangan titik. Hasil perhitungan jarak antara semua titik kemudian dimasukkan ke dalam Tabel 3. 6 berikut:

Tabel 3. 6 Tabel jarak antar kota

	A	**B**	**C**	**D**	**E**
A	0	12	17	17	19
B	12	0	13	29	23
C	17	13	0	16	36
D	17	29	16	0	20
E	19	23	36	20	0

Setelah proses hitung jarak antara titik selesai. dapat dilakukan penjumlahan untuk semua permutasi.

1. B-C-D-E

   A-B-C-D-E-A berarti terdiri dari

$$d = A - B \; + \; B - C \; + \; C - D \; + \; D - E \; + \; E - A$$

$$d = 12 \; + \; 13 \; + \; 16 \; + \; 20 + 19$$

$$d = \; 80$$

2. C-B-D-E
3. D-B-C-E
4. B-D-C-E
5. ….

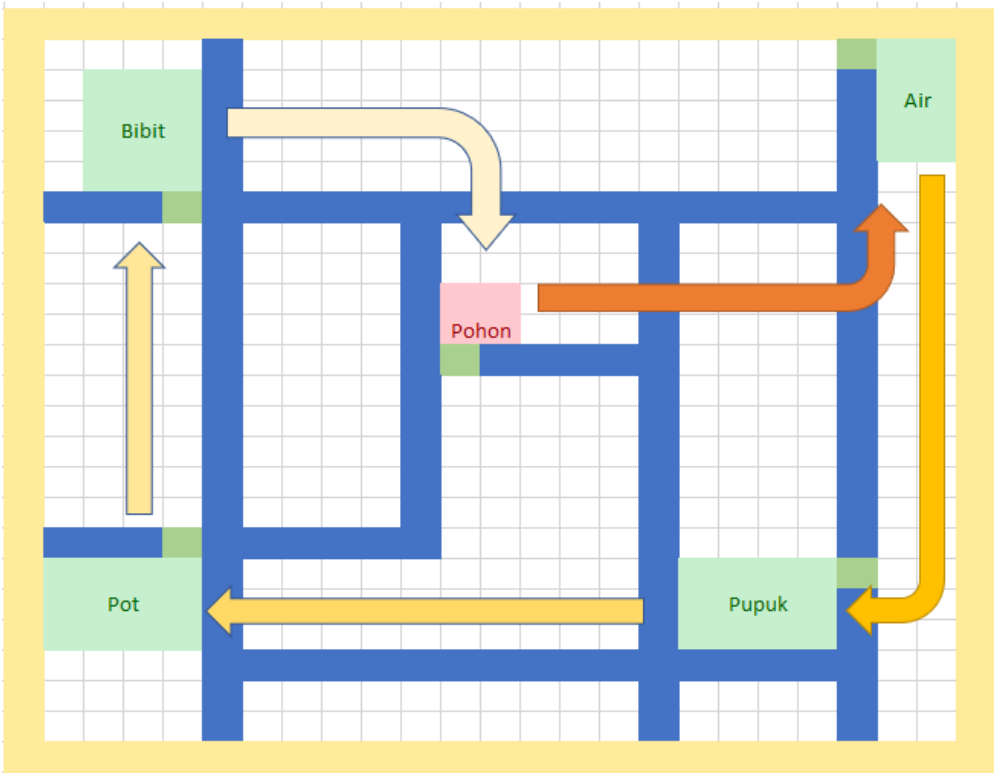
24. E-D-C-B

c. Setelah semua permutasi dihitung, rute dengan *cost* terkecil dipilih sebagai rute optimal.

Tabel 3. 7 Tabel *cost* rute optimal

Rute	**Harga**
A - B - C - D - E – A	80
A - B - C - E - D – A	98
A - B - D - C - E – A	112
A - B - D - E - C – A	114
A - B - E - C - D – A	104
A - B - E - D - C – A	88
A - C - B - D - E – A	98
A - C - B - E - D – A	90
A - C - D - B - E – A	104
A - C - D - E - B – A	88
A - C - E - B - D – A	122
A - C - E - D - B – A	114
A - D - B - C - E – A	114
A - D - B - E - C – A	122
A - D - C - B - E – A	88
A - D - C - E - B – A	104
A - D - E - B - C – A	90
A - D - E - C - B – A	98
A - E - B - C - D – A	88
A - E - B - D - C – A	104
A - E - C - B - D – A	114
A - E - C - D - B – A	112
A - E - D - B - C – A	98
A - E - D - C - B – A	80

Berdasarkan Tabel 3. 7 terdapat dua hasil yang mempunyai *cost* terkecil yaitu jalur A - B - C - D - E - A dan jalur A - E - D - C - B - A dengan *cost* 80. Jika terdapat lebih dari 1 solusi maka dapat dipilih secara acak, atau bisa mempertimbangkan faktor lain seperti prioritas sub objectvies. Dalam kasus ini akan diambil jalur A - E - D - C - B - A.

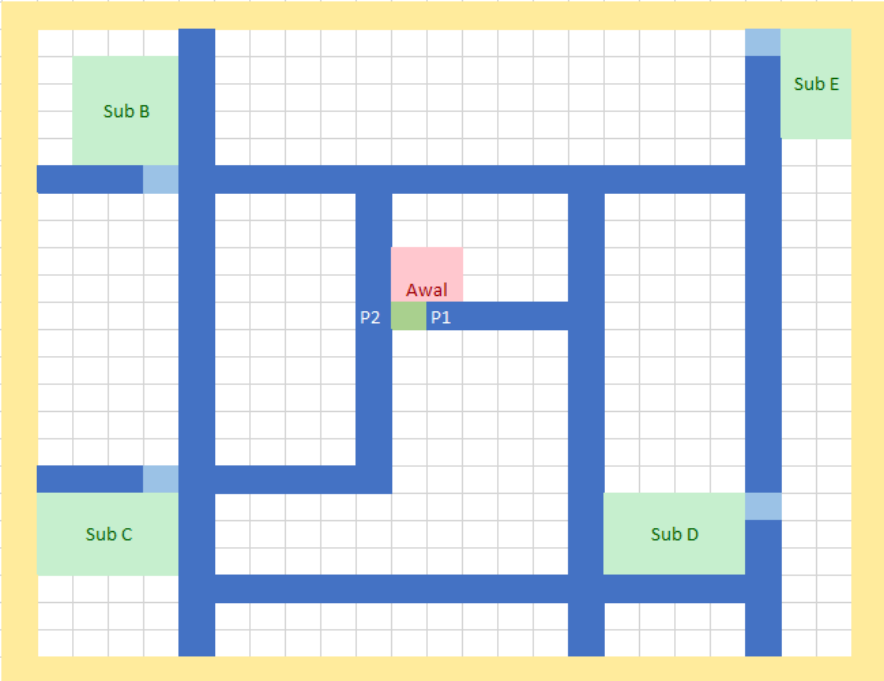


Gambar 3. 6 Hasil Sekenario TSP

Hasil perhitungan TSP dapat dilihat pada Gambar 3. 6 yang memperlihatkan visualisasi hasil skenario TSP demgam menunjukkan urutan kunjungan dari titik awal ke titik tujuan secara berurutan. Urutan tersebut ditandai melalui gradasi warna, dimulai dari warna yang lebih pekat sebagai titik awal perjalanan, menuju warna yang lebih terang sebagai titik akhir. Representasi visual ini mempermudah

pemahaman jalur optimal yang dihasilkan oleh algoritma, serta memperjelas pola rute yang dipilih berdasarkan perhitungan *cost* terkecil.

3.7 Permasalahan A*



Gambar 3. 7 Sekenario objektif *game*

Pada Gambar 3. 7 skenario permainan adalah pergi dari *node* awal ke *node* E. Untuk mencapai *node* tersebut, ada dua pilihan jalur, yaitu melalui titik P1 atau titik P2. Cara menentukan jarak terbaiknya adalah menghitung masing masing dari titik tersebut heuristiknya dan kita masukan kedalam open list. Perhitungan P1 dapat menggunakan rumus 2.3:

$$f(P1) = g(P1) + h(P1)$$

$$f(P1) = 1 + |21 - 12| + |23 - 13|$$

$$f(P1) = 20$$

Tabel 3. 8 Tabel *Open list* 1

Open list	f(n)
P1	20
P2	22

Setelah ditemukan dari Tabel 3. 8 nilai mana yang lebih kecil kita dapat masukan ke *Closed list* Tabel 3. 9. Jalur kita dibuat dengan berulang kali menelusuri *Open list* dan memilih *grid* dengan skor f(n) terendah.

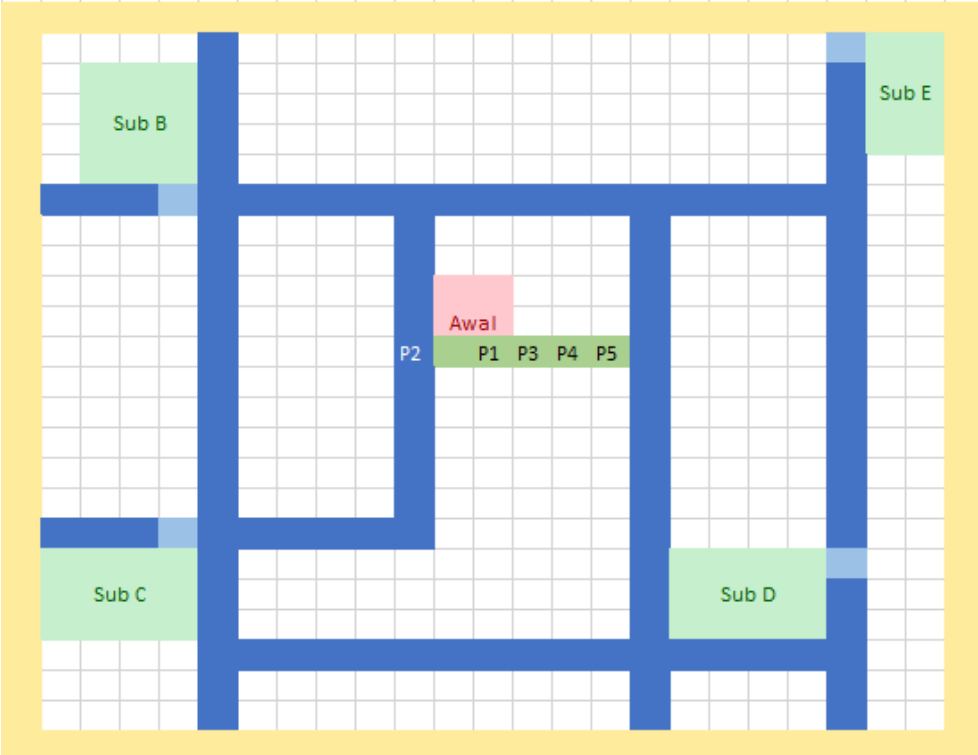
Tabel 3. 9 Tabel *Closed list* 1

Closed list	f(n)
A	20
P1	20

Pada tahap ini, ditunjukan pada Tabel 3. 10 terdapat dua titik, P2 dan P5, karena P5 memiliki jarak yang lebih kecil dari *node* P2, maka P5 akan diperluas terlebih dahulu.

Tabel 3. 10 Tabel *Open list* 2

Open list	f(n)
P2	22
P5	20



Gambar 3. 8 Langkah Ekspansi Selanjutnya

Titik P5 diekspansi, sehingga mempunyai tetangga yaitu titik P6 yang akan dimasukkan ke dalam *open list*. Hasil perhitungan *cost* P6 dapat dilihat pada Tabel 3. 11. P5 kemudian dimasukkan ke dalam *closed list* karena telah diperluas. Hasil perhitungan *cost* dari *closed list* yang telah diteruskan dapat dilihat pada Tabel 3. *12*. Langkah-langkah ini diteruskan dan di rangkum hingga seperti pada Gambar 3. 9

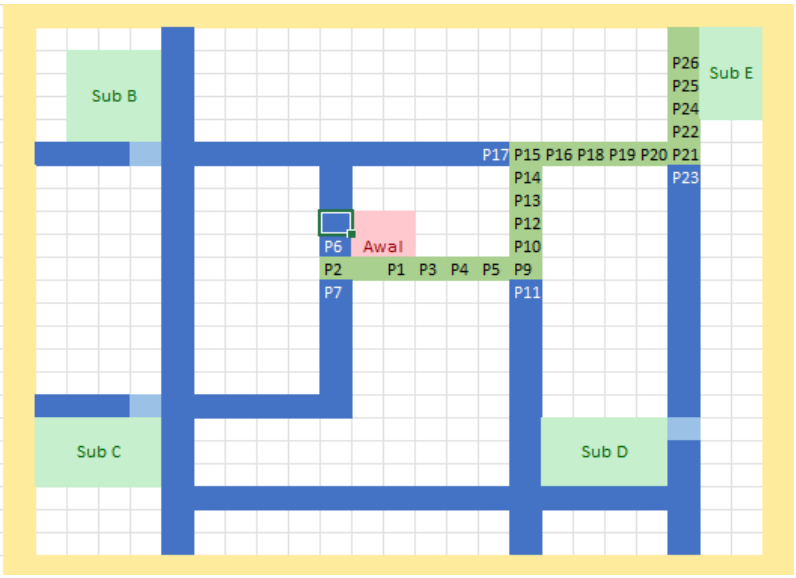
Tabel 3. 11 Tabel *Open list* 3

Open list	**f(n)**
P2	22
P6	20

Tabel 3. 12 Tabel *Closed list 2*

Closed list	$f(n)$
A	20
P1	20
P3	20
P4	20
P5	20

Gambar 3. 9 menunjukkan keseluruhan hasil perhitungan yang telah menemukan jalur terpendek dari titik awal hingga titik tujuan. Jalur yang ditemukan hanya satu, dimulai dari titik "Awal" hingga mencapai titik tujuan tanpa adanya percabangan. Oleh karena itu, proses *backtracking* tidak diperlukan dalam kasus ini karena tidak ada jalur alternatif yang perlu dievaluasi. Jalur yang teridentifikasi sudah langsung menuju tujuan secara optimal, sehingga setiap langkah dari titik awal hingga titik tujuan sudah pasti dan tidak membutuhkan koreksi lebih lanjut.



Gambar 3. 9 Jalur telah ditemukan

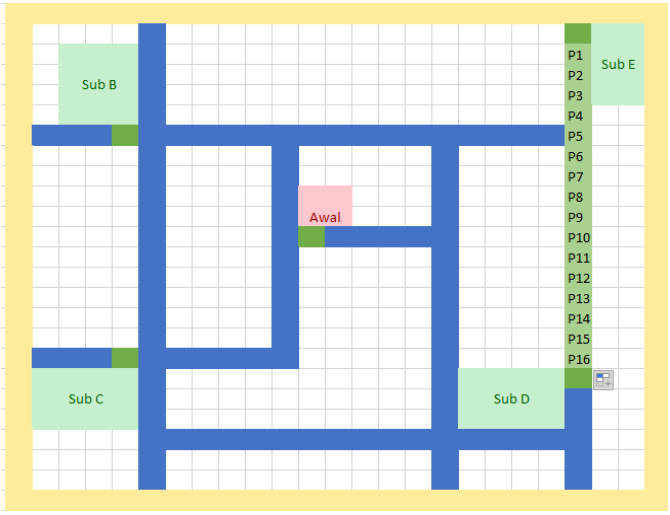
Pada Tabel 3. 13 merupakan detail dari hasil perhitungan *cost* setiap titik yang telah diekspansi menggunakan heuristik Manhattan. Hasil tersebut menunjukan P1 sampai P27 telah masuk ke dalam *closed list.* Sebagian besar titik yang telah diekspansi memiliki nilai *cost* yang sama. Hal ini terjadi karena contoh jalur yang disimulasikan simetris sehingga hanya memiliki satu kemungkinan jalur terpendek.

Tabel 3. 13 Tabel *Closed list* Akhir

Closed list	$\boldsymbol{f(n)}$
Awal	20
P1	20
P3	20
P4	20
P5	20
P9	20
P10	20
P12	20
P13	20
P14	20
P15	20
P16	20
P18	20
P19	20
P20	20
P21	20
P22	20
P24	20
P25	20
P26	20

Closed list	$f(n)$
P27 / E	20

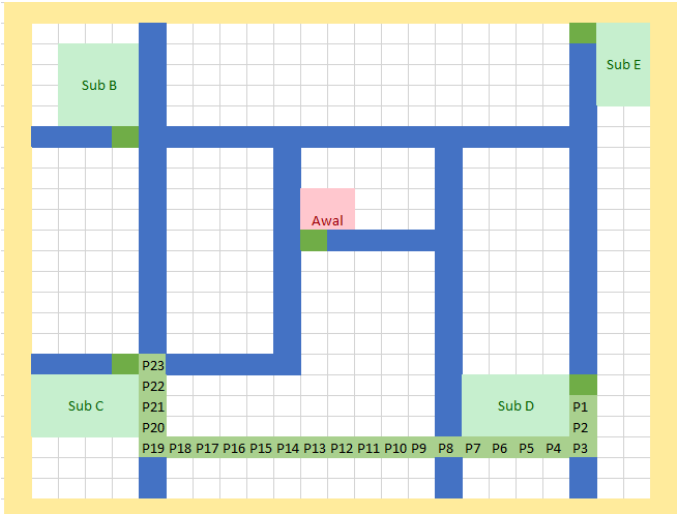
– Jalur E - D



Gambar 3. 10 Jalur E – D

Kemudian Jalur dari titik E ke D akan dilanjutkan dihitung, yang ditampilkan pada Gambar 3. 10.

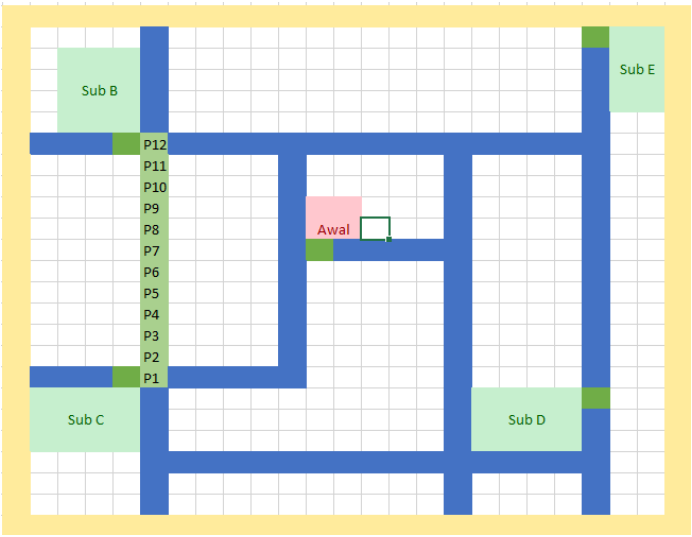
– Jalur D - C



Gambar 3. 11 Jalur D – C

Selanjutnya, perhitungan dilanjutkan dari titik D ke C, seperti yang divisualisasikan pada Gambar 3. 11.

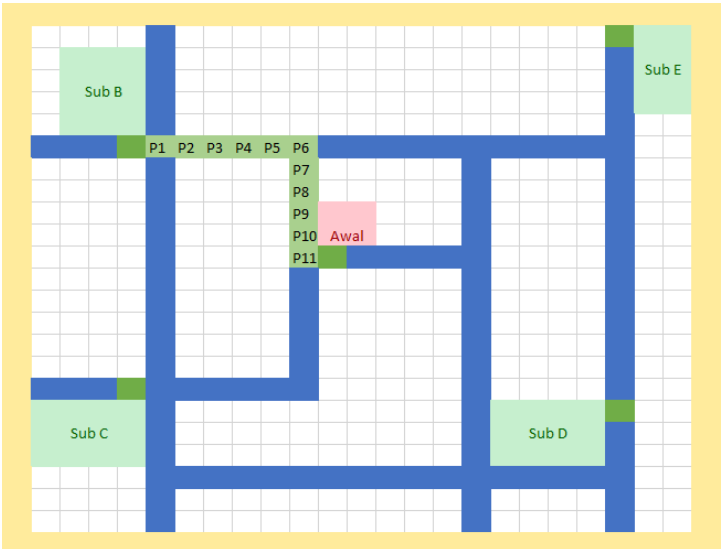
- Jalur C - B



Gambar 3. 12 Jalur C – B

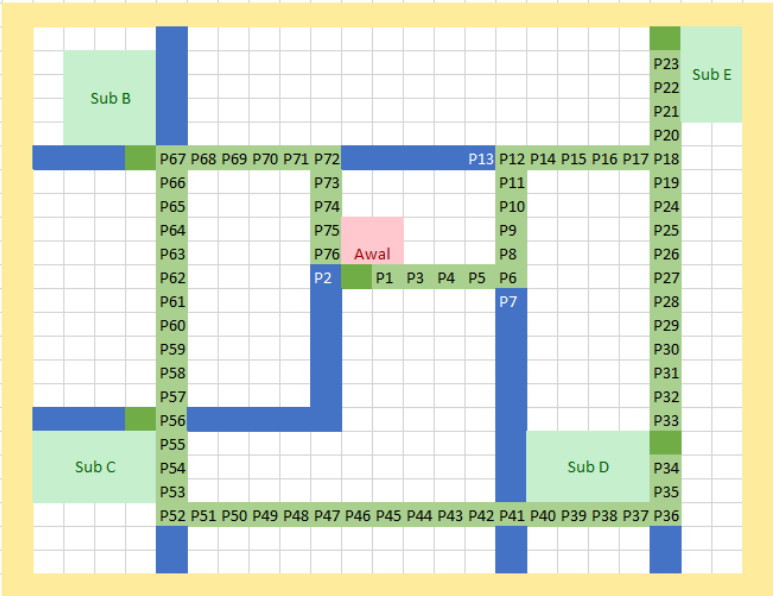
Langkah berikutnya adalah perhitungan jarak dari titik C ke B, sebagaimana ditunjukkan dalam Gambar 3. 12.

- Jalur B - A



Gambar 3. 13 Jalur B – A

Terakhir, jarak dihitung dari titik B ke A, dan hasilnya ditampilkan pada Gambar 3. 13.



Gambar 3. 14 Jalur selesai ditentukan

Setelah seluruh jalur antar titik dihitung dan digabungkan, maka terbentuklah jalur lengkap dari awal hingga akhir. Hasil akhir dari perhitungan ini divisualisasikan pada Gambar 3. 14, yang menunjukkan rute lengkap yang telah ditentukan berdasarkan urutan dan jarak optimal antar titik.

BAB IV

HASIL DAN PEMBAHASAN

4.1 Implementasi

Pada bab ini dijelaskan mengenai proses implementasi sistem yang telah dirancang pada bab sebelumnya. Implementasi dilakukan pada *game Cool City* yang dikembangkan menggunakan Unity Engine 2022.3.47f1 dan Visual Studio Code. Fokus utama implementasi adalah pada sistem pergerakan karakter yang menggunakan algoritma pencarian jalur, yaitu A* dan *Brute Force*.

Pada *game Cool City*, *player* memiliki tugas untuk membangun lokasi tujuan, untuk mencapainya *player* harus mengunjungi beberapa lokasi penting untuk mengambil item dalam peta kota untuk kemudian digunakan untuk pembangunan lokasi tersebut. Untuk menentukan rute terbaik yang harus dilalui karakter, digunakan dua metode algoritma yaitu *Brute Force* dan A*. Sistem ini dirancang agar *player* dapat berpindah tempat menuju beberapa lokasi secara mudah tanpa harus mempertimbangkan jalur terpendek dan waktu tempuh tercepat.

4.2 Implementasi Algoritma *Brute Force*

Metode *Brute Force* digunakan untuk mencari semua kemungkinan urutan perjalanan ke lokasi-lokasi penting. Tujuannya adalah menemukan rute dengan total jarak paling pendek yang harus dilalui oleh karakter. Hasil dari algoritma ini adalah daftar urutan lokasi yang harus dikunjungi karakter. Lokasi awal yang merupakan lokasi skenario dimulai, ditunjukan dengan angka 0 dan angka 1-9 digunakan sebagai output penunjuk lokasi penting lainnya untuk mengambil item

yang perlu dikoleksi untuk melakukan pembangunan. Garis bantu berwarna putih dan angka akan muncul untuk membantu *player* menentukan lokasi yang perlu dikunjungi serta urutan kunjungan.

Algoritma *Brute Force* berjalan ketika *player* memilih untuk melakukan pembangunan. Dalam aksi pembangunan *player* perlu mengunjungi area pembangunan layaknya survei tempat pembangunan kemudian sistem akan menyediakan berbagai lokasi penting untuk mengambil item untuk dikunjungi. Sebelum ditampilkan akan dilakukan perhitungannya mulai dari membuat setiap kombinasi yang bisa dilakukan.

Tabel 4. 1 *Pseudocode* Membuat Permutasi

```
fungsi Membuat Urutan Tujuan(daftarTujuan):
    daftar Urutan = list kosong untuk menyimpan semua urutan
    Susun Urutan(daftarTujuan, mulai dari posisi pertama, daftar Urutan)
    kembalikan daftar Urutan

fungsi Susun Urutan(daftar Tujuan, posisi Saat Ini, daftar Urutan):
    jika posisi Saat Ini adalah posisi terakhir dalam daftar:
        salin urutan sekarang dan tambahkan ke daftar Urutan
        keluar dari prosedur

    untuk setiap posisi saat ini hingga akhir daftar:
        tukar posisi titik saat ini dengan titik yang sedang dicek
        panggil kembali Susun Urutan dengan posisi berikutnya
        kembalikan posisi titik seperti semula (tukar kembali)
```

Setelah semuanya kemungkinan jalan ditemukan kemudian dibuat tabel matriks jarak untuk jarak setiap titik agar seperti Tabel 3.3 tabel jarak.

Tabel 4. 2 *Pseudocode* Membuat Matriks Jarak

```
Prosedur BuatMatriksJarak(daftarTujuan, titikAwalAkhir):
   jumlahTitik = jumlah dari daftarTujuan + 1  // tambah titik awal/akhir
   matriksJarak = matriks dua dimensi [jumlahTitik][jumlahTitik]

   untuk setiap baris dari 0 sampai jumlahTitik - 1:
      untuk setiap kolom dari baris sampai jumlahTitik - 1:
         jika baris dan kolom adalah titik yang sama:
            isi matriksJarak[baris][kolom] dengan 0
         jika tidak:
            jika baris adalah titik terakhir:
               posisiA = posisi titikAwalAkhir
            jika tidak:
               posisiA = posisi dari daftarTujuan pada indeks baris

            jika kolom adalah titik terakhir:
               posisiB = posisi titikAwalAkhir
            jika tidak:
               posisiB = posisi dari daftarTujuan pada indeks kolom

            jarak = hitung jarak Euclidean antara posisiA dan posisiB
            isi  matriksJarak[baris][kolom]  dan  matriksJarak[kolom][baris]
dengan jarak

   kembalikan matriksJarak
```

Kemudian lakukan evaluasi semua rute permutasi untuk mencari yang jaraknya paling pendek.

Tabel 4. 3*Pseudocode* Slove TSP

```
Prosedur TemukanJalurTerpendek():
   totalBangunan = jumlah seluruh bangunan tujuan
   daftarTujuan = daftar indeks semua bangunan tujuan (tanpa bangunan
awal/akhir)

   semuaKombinasiJalur() = buat semua kemungkinan urutan kunjungan
dari daftarTujuan

//inisialisasi
   jalurTerbaik = belum ada
   jarakTerpendek = nilai maksimum (tak hingga)

   untuk setiap kombinasiJalur dalam semuaKombinasiJalur:
```

```
            jarakTotal = 0
            titikSebelumnya = indeks bangunan awal/akhir

            untuk setiap bangunan dalam kombinasiJalur:
                tambahkan jarak dari titikSebelumnya ke bangunan saat ini ke
jarakTotal
                titikSebelumnya = bangunan saat ini

            tambahkan jarak kembali ke bangunan awal/akhir ke jarakTotal

            jika jarakTotal lebih kecil dari jarakTerpendek:
                simpan kombinasiJalur sebagai jalurTerbaik
                simpan jarakTotal sebagai jarakTerpendek

        tambahkan bangunan awal/akhir di awal dan akhir jalurTerbaik
        tampilkan jalur terbaik, jarak terpendek, jumlah kemungkinan
kombinasi, dan jumlah bangunan

        kembalikan jalurTerbaik dan jarakTerpendek
```

4.3 Implementasi Algoritma A*

Setelah mendapatkan urutan kunjungan, *player* dapat melakukan kliking pada jalan disekitar karakter sehingga karakter berjalan sesuai dengan titik yang diklik. Hal ini dibuat sedemikian rupa untuk menambahkan aspek kebebasan dan aspek kesenangan. Namun *player* bisa langsung klik lokasi tujuan, dan sistem akan secara otomatis mencari jalur tercepat menggunakan algoritma A*.

Algoritma A* akan berjalan jika mouse di klik saat kursor berada pada jalan dan tidak saat diatas UI. Maka *open list* dan *closed list* akan dikosongkan terlebih dahulu, kemudian sistem akan memanggil Algoritma A* dan apabila kalkulasi telah ditemukan maka sistem akan memanggil fungsi *MoveAlongPath*.

Tabel 4. 4 *Pseudocode* Kondisi Jalan A*

```
Update:
   Jika mouse di klik:
      Simpan posisi mouse klik
      konversi mouse klik dan posisi karakter menjadi koordinat
      Kosongkan Open list dan Close List
      Panggil Metode Astar(lokasi awal, target lokasi)
      jika jalan ditemukan:
         Aktifkan status bergerak
      Atur indeks jalur saat ini kembali ke awal

   Jika status bergerak aktif dan jalur tidak kosong:
   Panggil fungsi untuk bergerak mengikuti jalur yang sudah ditemukan
```

Karakter bergerak ketika indeks masih dalam batas jalur, jika kondisi tersebut tidak terpenuhi maka semua titik sudah dilewati dan pergerakan telah selesai.

Tabel 4. 5 *Pseudocode* Fungsi Move Along Path

```
Fungsi MoveAlongPath:
Jika indeks saat ini masih dalam batas jalur:
   Ambil posisi titik tujuan saat ini
   Gerakkan karakter menuju titik tersebut
   Hadapkan karakter ke arah titik tersebut
   Jika sudah dekat dengan titik:
      Lanjut ke titik berikutnya (tambah indeks)

Jika semua titik sudah dilewati:
   Hentikan pergerakan
   Aktifkan regenerasi
   Jalankan proses tunggu regenerasi (coroutine)
   Kosongkan jalur
   Tampilkan log "Pergerakan Selesai"
```

Algoritma A* diawali dengan inisialisasi nilai g, h, dan f pada *node* awal, kemudian *node* tersebut dimasukkan ke dalam *open list*. Selama *open list* tidak kosong dan iterasi tidak melebihi batas, algoritma akan memilih *node* dengan nilai f terendah. Jika *node* tersebut adalah tujuan, maka jalan akan direkonstruksi. Akan

tetapi jika *node* tersebut bukan tujuan, maka *node* akan dipindahkan ke *closed list*, dan semua tetangganya diperiksa. Tetangga yang sudah ada di *closed list* akan dilewati, sedangkan yang lain akan dihitung ulang nilai g, h, dan f-nya. Jika tetangga belum ada di *open list* atau nilai g-nya lebih baik, maka diperbarui dan dimasukkan ke *open list*. Proses ini berulang hingga tujuan ditemukan atau tidak ada jalur yang memungkinkan.

Tabel 4. 6 *Pseudocode* Fungsi A*

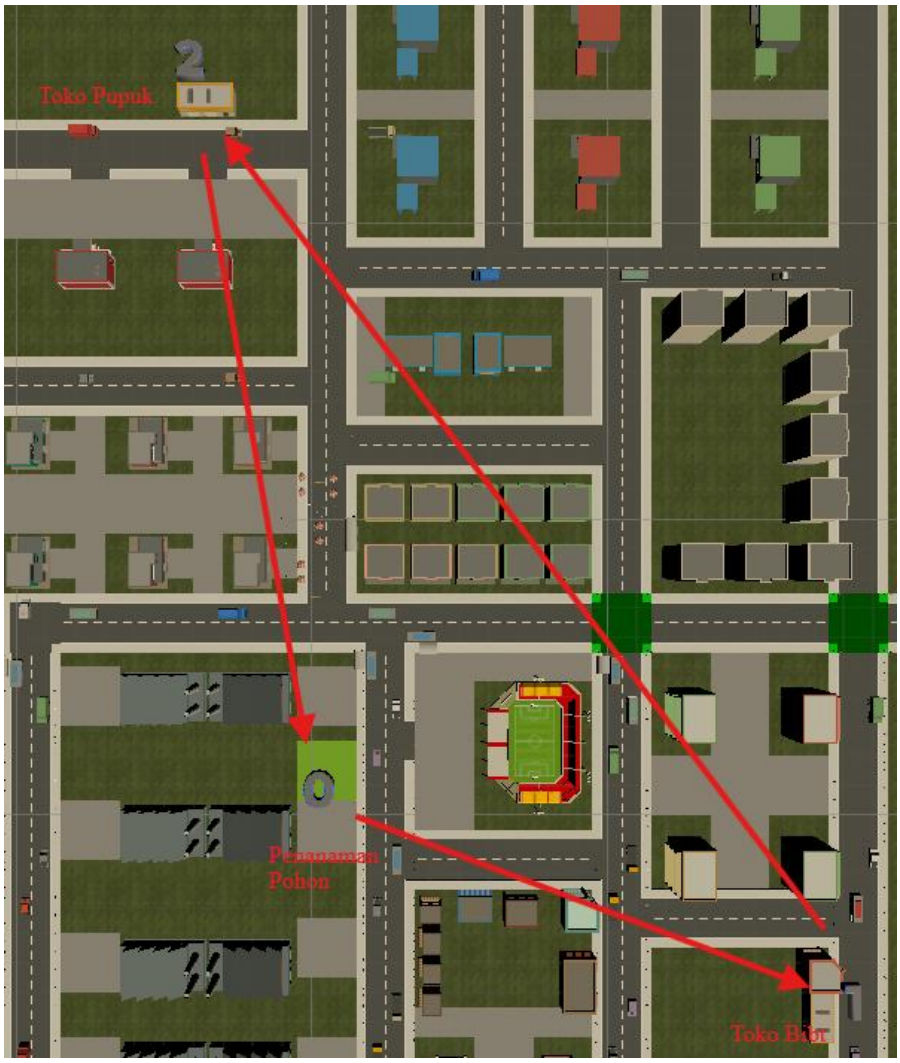
```
Fungsi Astar(titikAwal, titikTujuan):
  Inisialisasi titik awal:
    g = 0 (jarak dari awal)
    h = estimasi jarak ke tujuan (heuristik)
    f = g + h
  Masukkan titik awal ke daftar terbuka (OpenList)

  Selama OpenList tidak kosong dan belum melebihi batas perulangan:
    Ambil titik dengan nilai f terkecil dari OpenList sebagai titik saat ini

    Jika titik saat ini adalah tujuan:
      Kembalikan jalur dari awal ke tujuan //Return

    Pindahkan titik saat ini ke daftar tertutup (ClosedList)

    Untuk setiap tetangga dari titik saat ini:
      Jika tetangga sudah ada di ClosedList:
        Lewati
      Hitung jarak baru dari awal ke tetangga (g sementara)
      Jika tetangga belum ada di OpenList atau jarak baru lebih baik:
        Perbarui informasi tetangga:
          - Tetapkan titik saat ini sebagai induk
          - Hitung ulang g, h, dan f
        Masukkan tetangga ke OpenList (jika belum)

  Jika tidak ditemukan jalur:
    Kembalikan jalur kosong //Return
```

4.4 Pengujian

Pengujian dilakukan untuk mengevaluasi performa sistem dalam menentukan jalur tercepat karakter pada *game Cool City*, dengan menggunakan kombinasi Algoritma *Brute Force* dan A*. Pengujian dilakukan menggunakan metode blackbox, dengan memberikan masukan berupa skenario yang berbeda dan jumlah tujuan yang bervariasi. Selain itu, sistem juga diuji langsung oleh pengguna (*user*) yang belum pernah mencoba *game* sebelumnya untuk memastikan hasil yang objektif. Peserta diberikan waktu 20 menit untuk memainkan *game* tanpa panduan yang mendetail, sehingga mereka bebas mengeksplorasi fitur *game* secara mandiri. Setelah sesi bermain, peserta diminta untuk mengisi kuesioner *GUESS* yang akan berfokus pada aspek *usability*.

4.5 Pengujian *Brute Force*

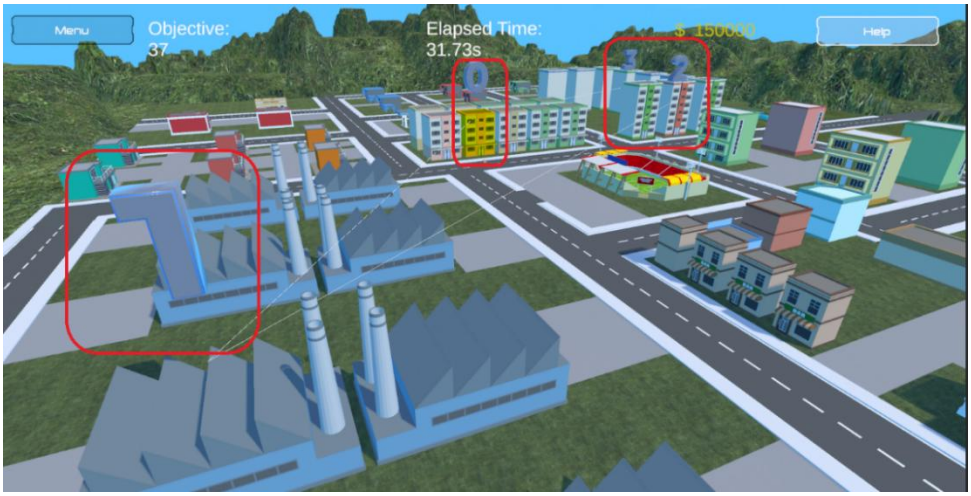
Dalam pengujian Algoritma *Brute Force*, dilakukan simulasi dengan memainkan *game* secara langsung. Gambar 4.1, menunjukan perjalanan dimulai dari lokasi penanaman pohon yang berindikasikan 0 lalu ke Toko Bibit yang berindikasikan angka 1 dan Toko Pupuk (angka 2) untuk memperoleh semua bahan yang dibutuhkan dalam proses pembangunan pohon lalu kembali.



Gambar 4. 1 Penanaman Pohon

Selanjutnya pada Gambar 4.2 pengecatan bangunan berada pada titik 0, kemudian dilakukan perjalanan ke beberapa lokasi sesuai urutan tugas yang ada. Pertama, dimulai dengan menuju ke Pabrik pada titik 1 untuk mengambil bahan cat dan alat pengecatan yang diperlukan. Pabrik ini berfungsi sebagai titik awal untuk persiapan pengecatan. Kemudian, perjalanan dilanjutkan ke Kantor Bangunan 1 pada titik 2 untuk memeriksa kualitas cat yang digunakan dan memastikan persediaan alat-alat pengecatan mencukupi. Selanjutnya, perjalanan menuju ke Kantor Bangunan 2 pada titik 3 adalah untuk mendapatkan persetujuan akhir dari

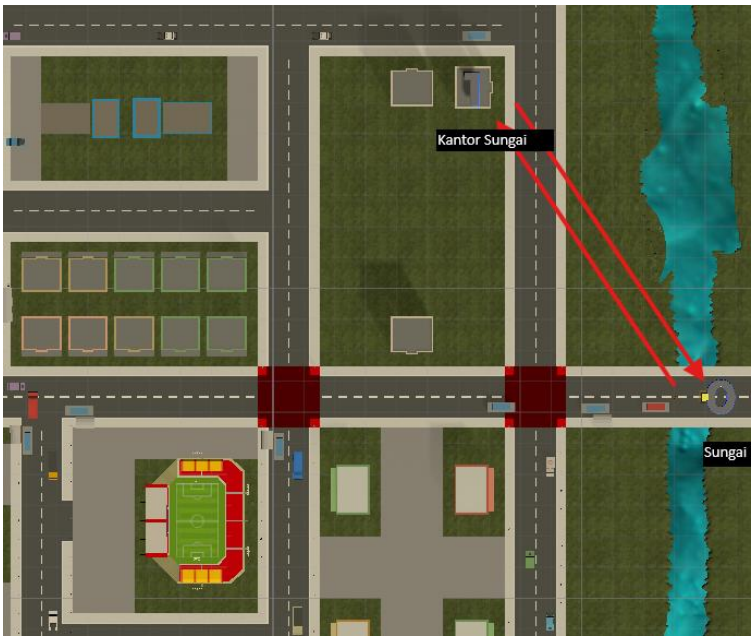
pihak pengawas proyek, dan terakhir kembali ke tempat pengecatan untuk memulai pengecatan.



Gambar 4. 2 Pengecatan Bangunan

– Pelebaran Sungai

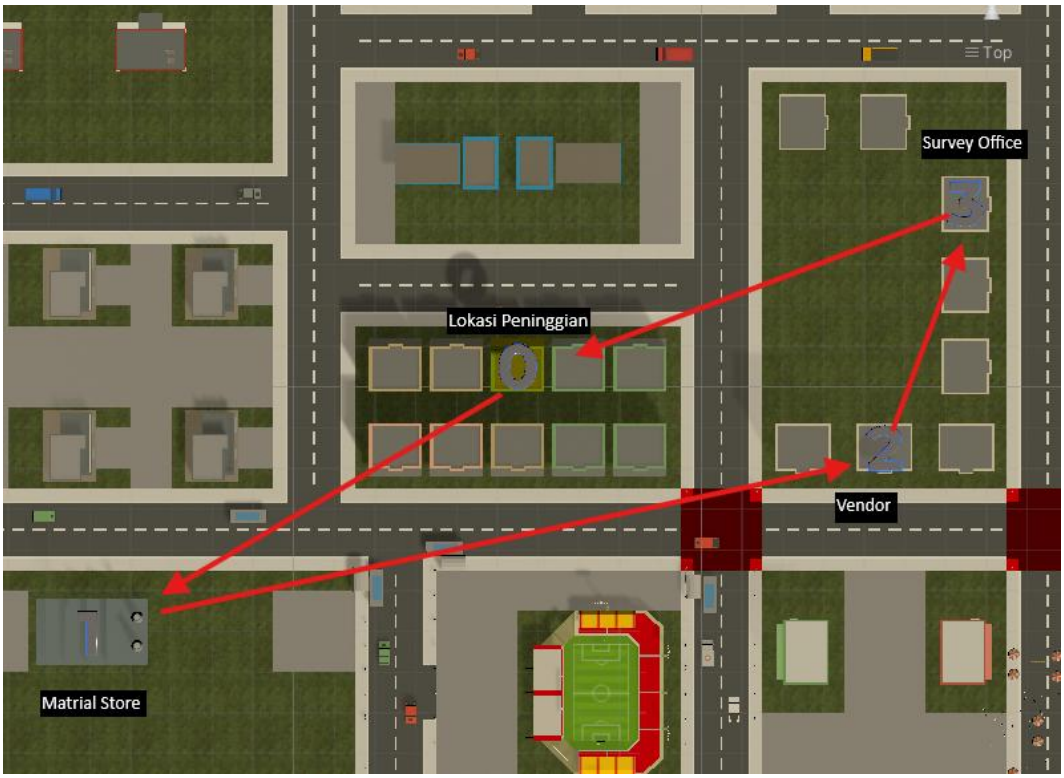
Gambar 4. 3 Pelebaran Sungai Gambar 4. 3 merupakan rute yang harus di tempuh untuk menyelesaikan objektif pelebaran sungai yang terdiri dari Sungai – Kantor Sungai – Sungai.



Gambar 4. 3 Pelebaran Sungai

– Peninggian Bangunan

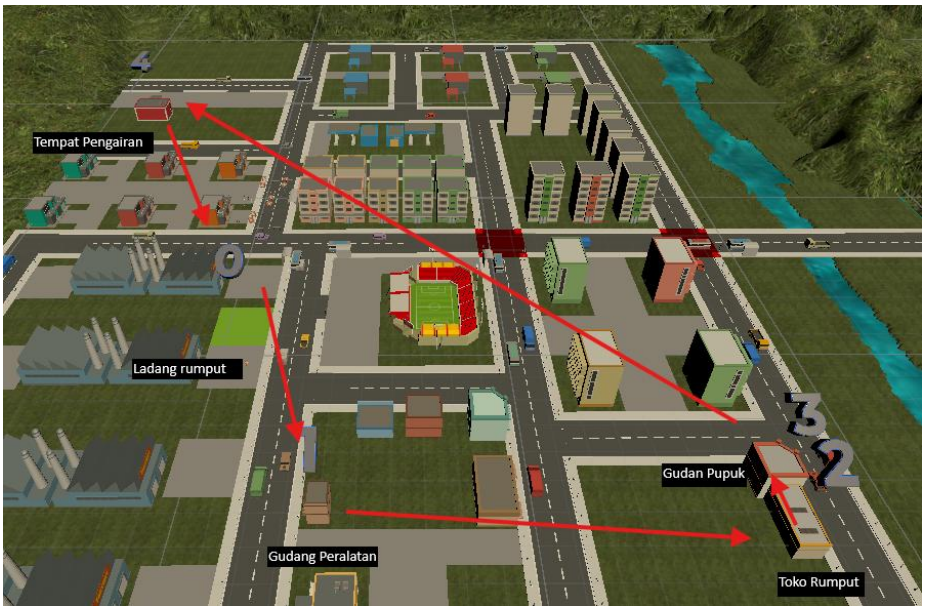
Gambar 4. 4 merupakan gambar untuk rute peninggian bangunan, karakter harus mengunjungi Lokasi Bangunan - *Material Store* - Rumah Walikota – area survey - Lokasi Bangunan.



Gambar 4. 4 Peninggian Bangunan

– Penanaman Rumput

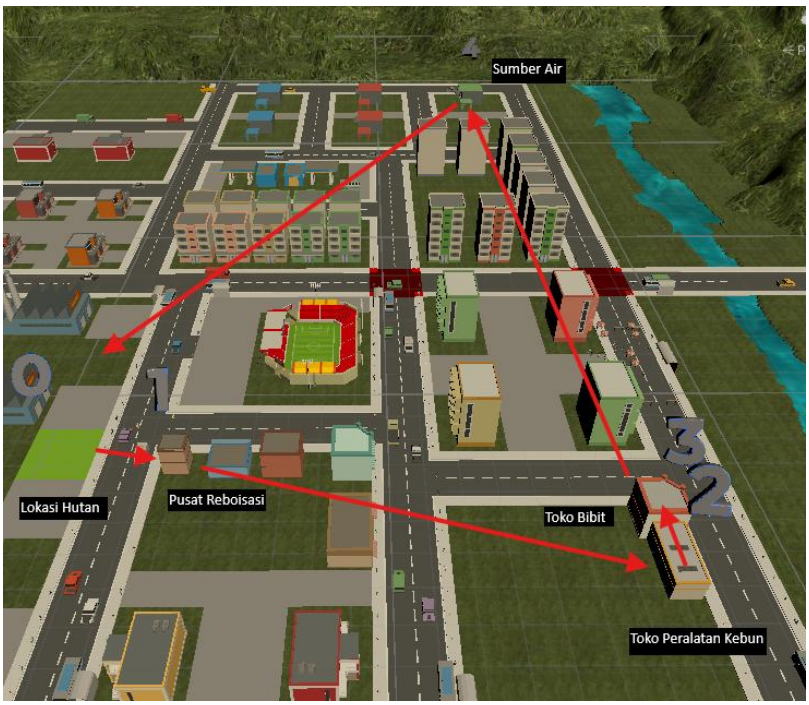
Objektif penanaman rumput pada Gambar 4. 5 mengharuskan karakter pergi ke Area Rumput, Gudang Pupuk - Toko Rumput - Gudang Peralatan - Tempat Pengairan - Area Rumput. Rute ini cukup panjang dan memiliki banyak titik perhentian.



Gambar 4. 5 Penanaman Rumput

– Hutan Kota

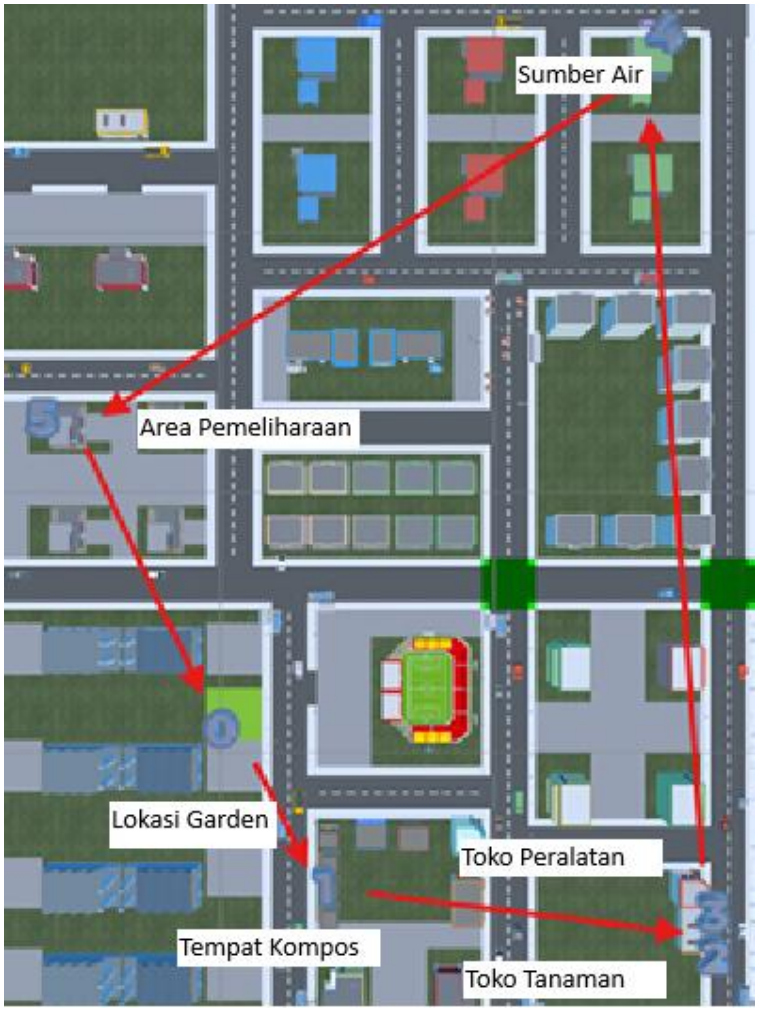
Objektif Hutan Kota mengharuskan karakter mengunjungi Hutan Kota - Pusat Reboisasi, Toko Peralatan Kebun - Sumber Air - Toko Bibit - Hutan Kota seperti yang ditunjukkan pada Gambar 4. 6.



Gambar 4. 6 Hutan Kota

– *Urban Garden*

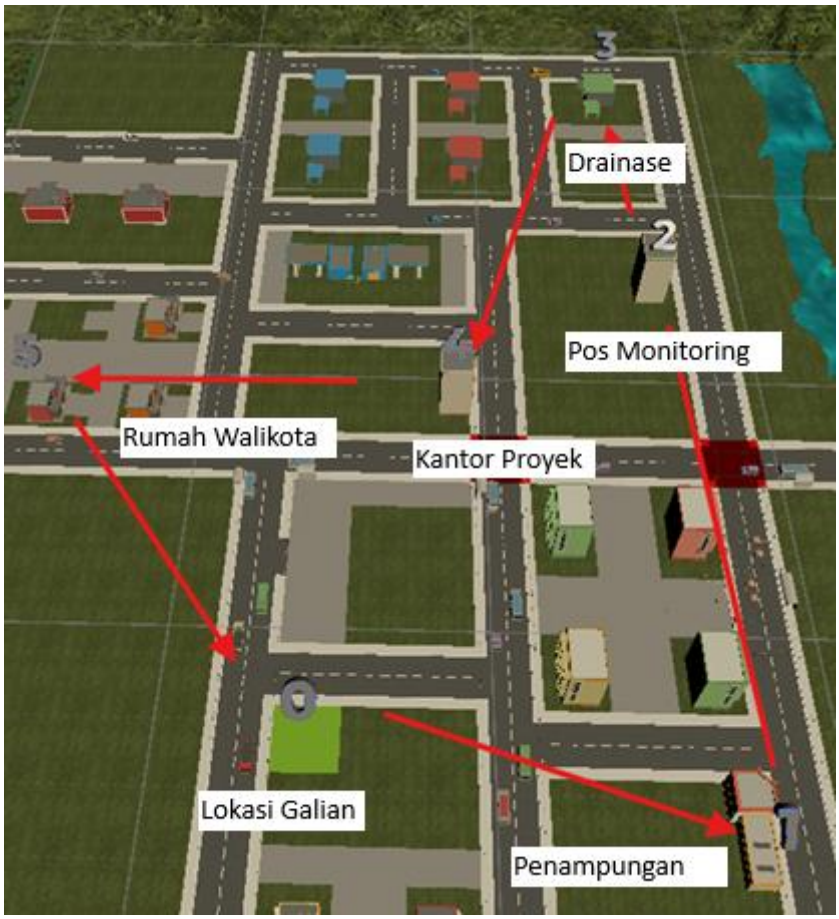
Gambar 4. 7 menunjukan objektif *Urban Garden*, *player* harus mengunjungi Lahan Taman - Toko Peralatan Taman - Area Pemeliharaan - Tempat Kompos - Sumber Air Toko tanaman - Lahan Taman.



Gambar 4. 7 Urban Garden

– Danau Buatan

Gambar 4. 8 menunjukan objektif Danau Buatan melibatkan perjalanan ke Titik Galian - Rumah Walikota - Pos Monitoring - Kantor Proyek - Penampungan Drainase - Titik Galian.



Gambar 4. 8 Danau Buatan

Rangkuman dari hasil pengujian ini dapat dilihat pada Tabel 4.7 berikut. Dalam tabel ini, dilakukan pengujian dengan berbagai kasus seperti jumlah total tujuan yang berbeda, yang mempengaruhi jarak dan waktu yang diperlukan untuk mencapai tujuan. Tabel 4.7 menggambarkan urutan kunjungan yang ditemukan oleh algoritma dan ekspektasi urutan kunjungan

Tabel 4. 7 Tabel *Blackbox*

Skenario	Nama Objektif	Lokasi yang harus dikunjungi	Rute Tersedia	Cost	Output *Brute Force*	Ket
1a	Pelebaran Sungai	– Sungai (A) – Kantor Sungai (B)	**A-B-A (Rute tercepat)**	256.12	A-B-A	sesuai
1b	Pelebaran Sungai	– Sungai (A) – Kantor Sungai (B)	**A-B-A (Rute tercepat)**	256.12	A-B-A	sesuai
1c	Pelebaran Sungai	– Sungai (A) – Kantor Sungai (B)	**A-B-A (Rute tercepat)**	256.12	A-B-A	sesuai
2a	Penanaman Pohon	– Toko Bibt (A) – Toko Pupuk (B) – Penanaman Pohon (C)	**C-A-B-C (Rute tercepat)**	**761.01**	C-A-B-C	sesuai
			C-B-A-C (Rute tercepat)	**761.01**		
2b	Penanaman Pohon	– Toko Bibt (A) – Toko Pupuk (B) – Penanaman Pohon (C)	**C-A-B-C (Rute tercepat)**	**761.01**	C-A-B-C	sesuai
			C-B-A-C (Rute tercepat)	**761.01**		
2c	Penanaman Pohon	– Toko Bibt (A) – Toko Pupuk (B) – Penanaman Pohon (C)	**C-A-B-C (Rute tercepat)**	**761.01**	C-A-B-C	sesuai
			C-B-A-C (Rute tercepat)	**761.01**		
3a	Pengecatan Bangunan	– Kantor Kontraktor (A) – Gudang Cat (B) – Toko Peralatan (C)	D-A-B-C-D	570,82	D-B-A-C-D	Sesuai
			D-A-C-B-D	406,31		
			D-B-A-C-D	**405,75**		

Skenario	Nama Objektif	Lokasi yang harus dikunjungi	Rute Tersedia	Cost	Output *Brute Force*	Ket
		– Lokasi pengecatan (D)	D-B-C-A-D	406,31		
			D-C-A-B-D	**405,75**		
			D-C-B-A-D	570,82		
3b	Peninggian Bangunan	– Vendor (A) – Material Store (B) – Gedung Survey (C) – Lokasi Peninggian (D)	D-A-B-C-D	648,65	D-B-A-C-D	Sesuai
			D-A-C-B-D	518,21		
			D-B-A-C-D	**503,92**		
			D-B-C-A-D	518,21		
			D-C-A-B-D	**503,92**		
			D-C-B-A-D	648,65		
3c	Peninggian Bangunan	– Vendor (A) – Material Store (B) – Gedung Survey (C) – Lokasi Peninggian (D)	D-A-B-C-D	648,65	D-B-A-C-D	Sesuai
			D-A-C-B-D	518,21		
			D-B-A-C-D	**503,92**		
			D-B-C-A-D	518,21		
			D-C-A-B-D	**503,92**		
			D-C-B-A-D	648,65		
4a	Penanaman Rumput	– Gudang Pupuk (A) – Toko Rumput (B) – Gudang Peralatan (C) – Tempat Pengairan (D)	E-A-B-C-D-E	778,35	E-C-B-A-D-E	Sesuai
			E-A-B-D-C-E	902,24		
			E-A-C-B-D-E	973,12		
			E-A-C-D-B-E	1115,91		
			E-A-D-B-C-E	1086,17		

Skenario	Nama Objektif	Lokasi yang harus dikunjungi	Rute Tersedia	Cost	Output *Brute Force*	Ket
		– Tempat Penanaman (E)	E-A-D-C-B-E	1105,07		
			E-B-A-C-D-E	785,01		
			E-B-A-D-C-E	898,06		
			E-B-C-A-D-E	968,94		
			E-B-C-D-A-E	1105,07		
			E-B-D-A-C-E	1092,83		
			E-B-D-C-A-E	1115,91		
			E-C-A-B-D-E	766,11		
			E-C-A-D-B-E	1092,83		
			E-C-B-A-D-E	**755,27**		
			E-C-B-D-A-E	1086,17		
			E-C-D-A-B-E	898,06		
			E-C-D-B-A-E	902,24		
			E-D-A-B-C-E	**755,27**		
			E-D-A-C-B-E	968,94		
			E-D-B-A-C-E	766,11		
			E-D-B-C-A-E	973,12		
			E-D-C-A-B-E	785,01		
			E-D-C-B-A-E	778,35		
4b	Hutan Kota	– Pusat Reboisasi (A) – Sumber Air (B) – Toko Bibit (C) – Toko Peralatan Kebun (D) – Hutan Kota (E)	E-A-B-C-D-E	854,51	E-A-D-C-B-E	Sesuai
			E-A-B-D-C-E	866,6		
			E-A-C-B-D-E	973,95		
			E-A-C-D-B-E	830,87		
			E-A-D-B-C-E	973,05		
			E-A-D-C-B-E	**817,88**		

Skenario	Nama Objektif	Lokasi yang harus dikunjungi	Rute Tersedia	Cost	Output *Brute Force*	Ket
			E-B-A-C-D-E	958,51		
			E-B-A-D-C-E	957,61		
			E-B-C-A-D-E	1064,96		
			E-B-C-D-A-E	**817,88**		
			E-B-D-A-C-E	1077,05		
			E-B-D-C-A-E	830,87		
			E-C-A-B-D-E	1113,68		
			E-C-A-D-B-E	1077,05		
			E-C-B-A-D-E	1100,69		
			E-C-B-D-A-E	973,05		
			E-C-D-A-B-E	957,61		
			E-C-D-B-A-E	866,6		
			E-D-A-B-C-E	1100,69		
			E-D-A-C-B-E	1064,96		
			E-D-B-A-C-E	1113,68		
			E-D-B-C-A-E	973,95		
			E-D-C-A-B-E	958,51		
			E-D-C-B-A-E	854,51		
4c	Hutan Kota	– Pusat Reboisasi (A) – Sumber Air (B) – Toko Bibit (C) – Toko Peralatan Kebun (D) – Hutan Kota (E)	E-A-B-C-D-E	854,51	E-A-D-C-B-E	Sesuai
			E-A-B-D-C-E	866,6		
			E-A-C-B-D-E	973,95		
			E-A-C-D-B-E	830,87		
			E-A-D-B-C-E	973,05		

Skenario	Nama Objektif	Lokasi yang harus dikunjungi	Rute Tersedia	Cost	Output *Brute Force*	Ket
			E-A-D-C-B-E	817,88		
			E-B-A-C-D-E	958,51		
			E-B-A-D-C-E	957,61		
			E-B-C-A-D-E	1064,96		
			E-B-C-D-A-E	817,88		
			E-B-D-A-C-E	1077,05		
			E-B-D-C-A-E	830,87		
			E-C-A-B-D-E	1113,68		
			E-C-A-D-B-E	1077,05		
			E-C-B-A-D-E	1100,69		
			E-C-B-D-A-E	973,05		
			E-C-D-A-B-E	957,61		
			E-C-D-B-A-E	866,6		
			E-D-A-B-C-E	1100,69		
			E-D-A-C-B-E	1064,96		
			E-D-B-A-C-E	1113,68		
			E-D-B-C-A-E	973,95		
			E-D-C-A-B-E	958,51		
			E-D-C-B-A-E	854,51		
5a	Urban Garden	– Toko Peralatan Taman (A) – Sumber Air (B) – Tempat Kompos (C)	F-A-B-C-D-E-F	1361,16	F-C-D-A-B-E-F	Sesuai
			….	….		
			F-C-D-A-B-E-F	**897,88**		
			…	…		

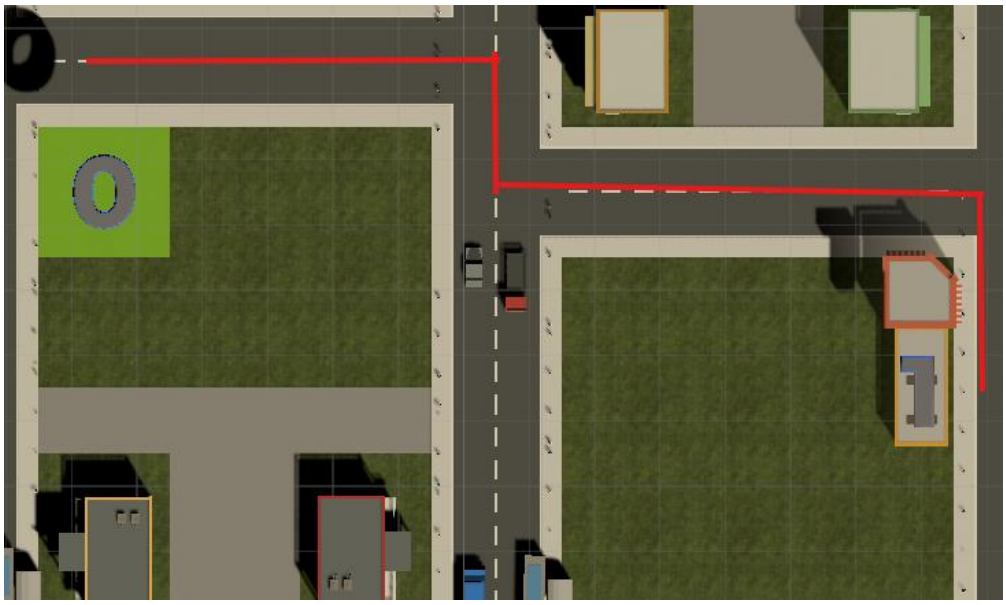
Skenario	Nama Objektif	Lokasi yang harus dikunjungi	Rute Tersedia	Cost	Output *Brute Force*	Ket
		– Toko tanaman (D) – Area Pemeliharaan (E) – Lokasi Garden (F)	**F-E-B-A-D-C-F**	**897,88**		
			…	…		
			F-E-D-C-B-A-F	1361,16		
5b	Danau Buatan	– Pos Monitoring (A) – Drainase (B) – Rumah Walikota (C) – Penampungan (D) – Kantor Proyek (E) – Lokasi Penggalian (F)	F-A-B-C-D-E-F	1152,31	F-A-D-B-E-C-F	Sesuai
			…	…		
			F-A-D-B-E-C-F	**876,4**		
			…	…		
			F-C-E-B-D-A-F	**876,4**		
			…	…		
			F-E-D-C-B-A-F	1152,31		
5C	Danau Buatan	– Pos Monitoring (A) – Drainase (B) – Rumah Walikota (C) – Penampungan (D) – Kantor Proyek (E) – Lokasi Penggalian (F)	F-A-B-C-D-E-F	1152,31	F-A-D-B-E-C-F	Sesuai
			…	…		
			F-A-D-B-E-C-F	**876,4**		
			…	…		
			F-C-E-B-D-A-F	**876,4**		
			…	…		
			F-E-D-C-B-A-F	1152,31		

Pada Tabel 4. 7 ditampilkan hasil pengujian rute dengan cost terkecil ditandai dengan font bold. Rute tersebut merupakan hasil perhitungan secara manual yang kemudian dibandingkan dengan hasil *output* Algoritma *Brute Force* yang dihasilkan. Berdasrkan hasil pengujian terlihat bahwa rute yang dihasilkan oleh Algoritma Brute Force selalu sesuai dengan rute yang memiliki *cost* terkecil dari seluruh kemungkinan rute yang tersedia, sebagaimana diperoleh melalui perhitungan manual. Contohnya dalam melaksanakan objektif penanaman pohon pemain diharuskan untuk pergi ke Toko Bibit, dan Toko Pupuk dengan lokasi awal berupa Lokasi Penanaman, maka urutan kunjungan yang tersedia ada dua yang sejalan dengan hasil *output* Algoritma *Brute Force*.

Hal ini membuktikan bahwa algoritma yang di implementasikan telah bekerja dengan baik dan dapat disimpulkan bahwa Algoritma *Brute Force* telah menghasilkan rute yang paling optimal, hal ini sejalan dengan penelitian Wawan Saputra, 2022 yang membuktikan bahwa Algoritma *Brute Force* merupakan solusi eksak yang pasti menghasilkan rute optimal.

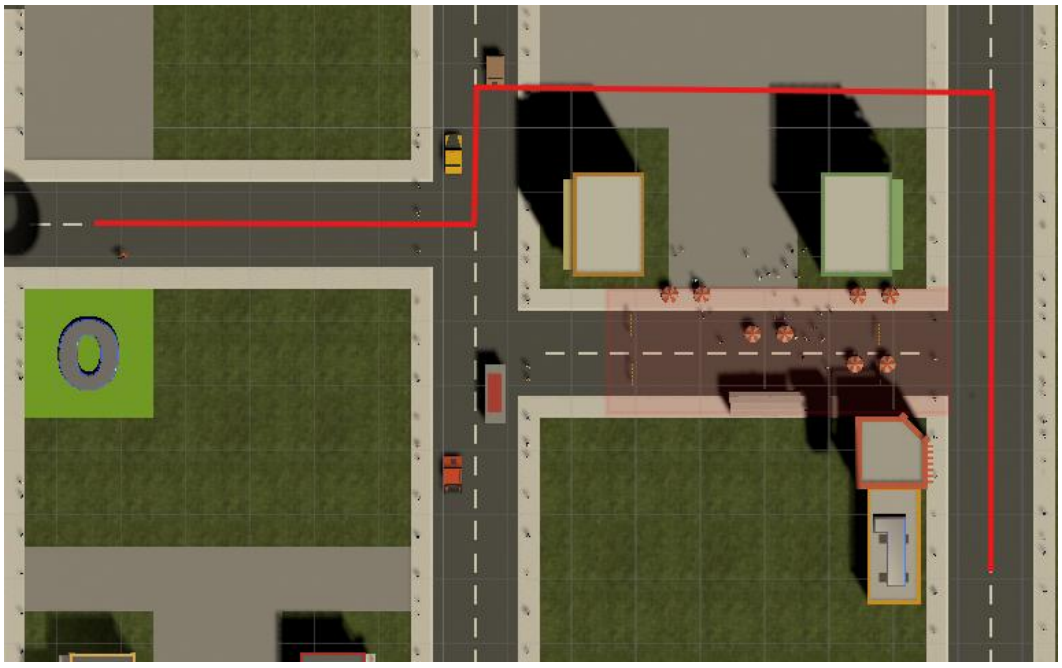
4.6 Pengujian A*

Pengujian Algoritma A* dilakukan dengan mengambil salah satu skenario yang telah berhasil diselesaikan oleh algoritma pada tahap sebelumnya, lebih tepatnya skenario yang diambil adalah pembuatan danau buatan. Skenario kemudian dianalisis berdasarkan rancangan pengujian yang telah ditentukan. Pengujian dilakukan tiga kali untuk setiap titiknya, ditunjukkan pada Gambar 4. 9 jalur dari titik 0 ke titik 1 (dikonversi menjadi titik A ke B).



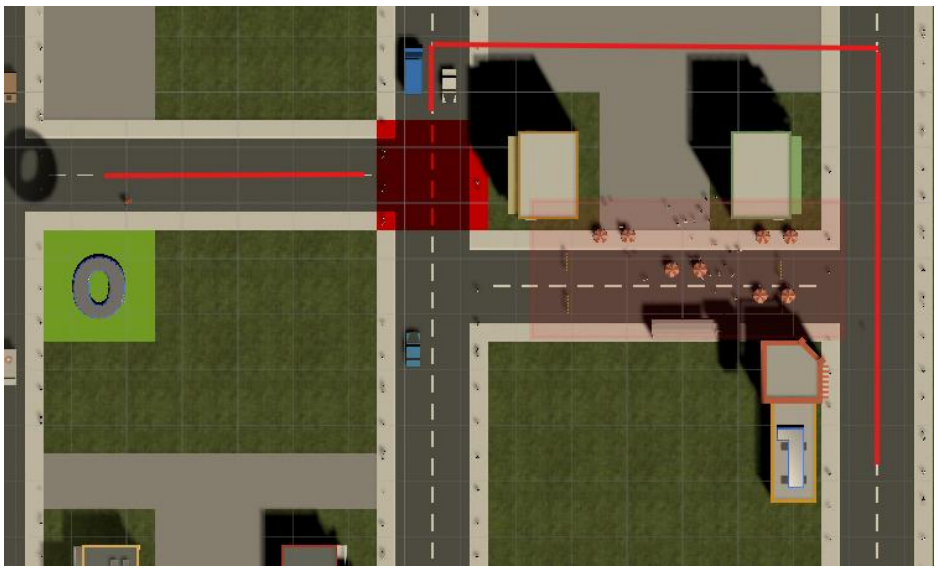
Gambar 4. 9 Titik A ke B (Uji 1)

Selanjutnya pada Gambar 4. 10 diberikan *static obstacle* berupa festifal yang menyebabkan rute optimal pada pengujian pertama tidak dapat dilalui lagi karena adanya penutupan jalan.



Gambar 4. 10 Titik A ke B (Uji 2)

Dalam Gambar 4. 11 ditunjukan penambahan *dynamic obstacle* berupa lampu merah disamping festifal. Pengujian dilakukan untuk mengetahui efek halangan yang bersifat dinamis terhadap algoritma.



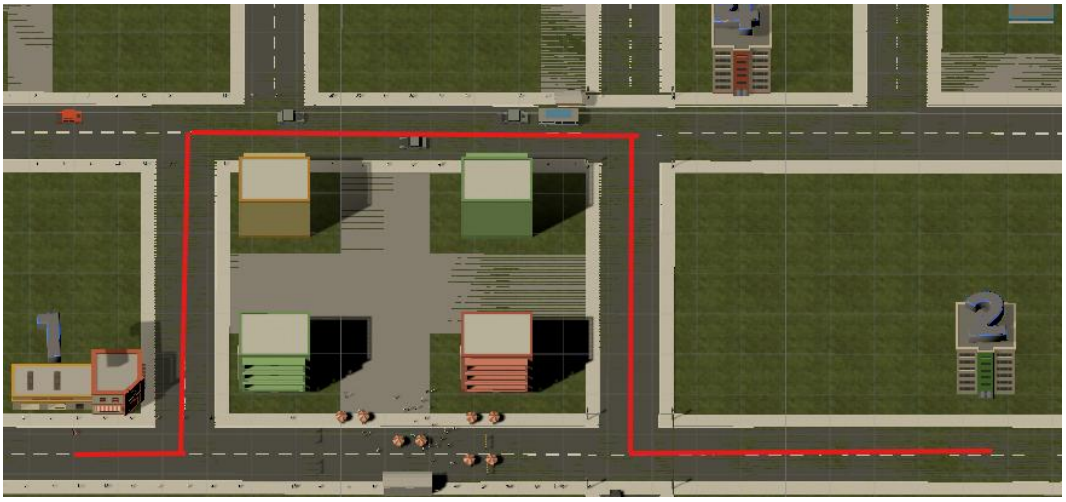
Gambar 4. 11 Titik A ke B (Uji 3)

Pada Gambar 4. 12, pengujian pertama untuk jalur dari titik 1 ke titik 2 (B ke C) dilakukan tanpa adanya rintangan di sepanjang jalur. Tujuan dari pengujian ini adalah untuk mengevaluasi performa dasar algoritma A* dalam kondisi ideal.

Gambar 4. 12 Titik B ke C (Uji 1)

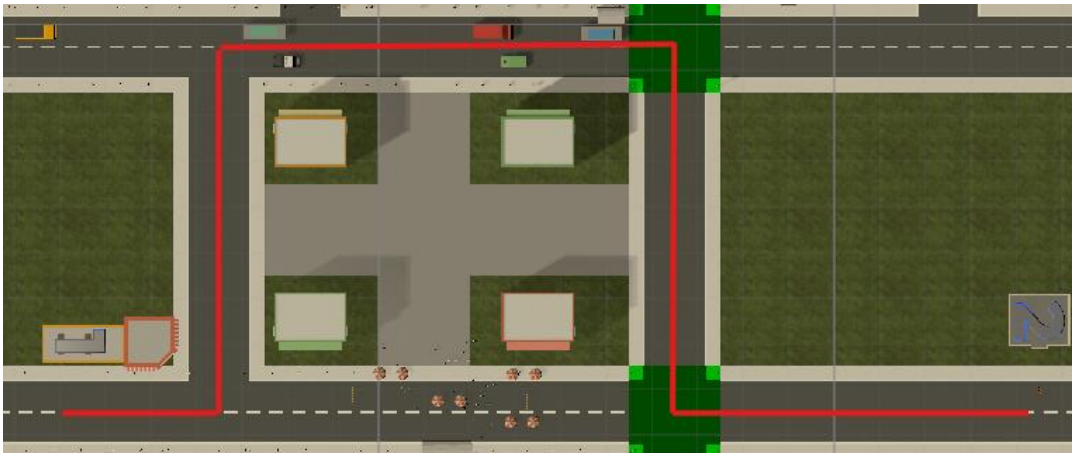
Gambar 4. 13 menunjukkan jalur yang sama, yakni dari B ke C, namun dengan kondisi berbeda pada pengujian kedua. Dalam pengujian ini, terdapat

rintangan berupa *obstacle festival* di sepanjang jalur, yang menjadi simulasi penutupan jalan.



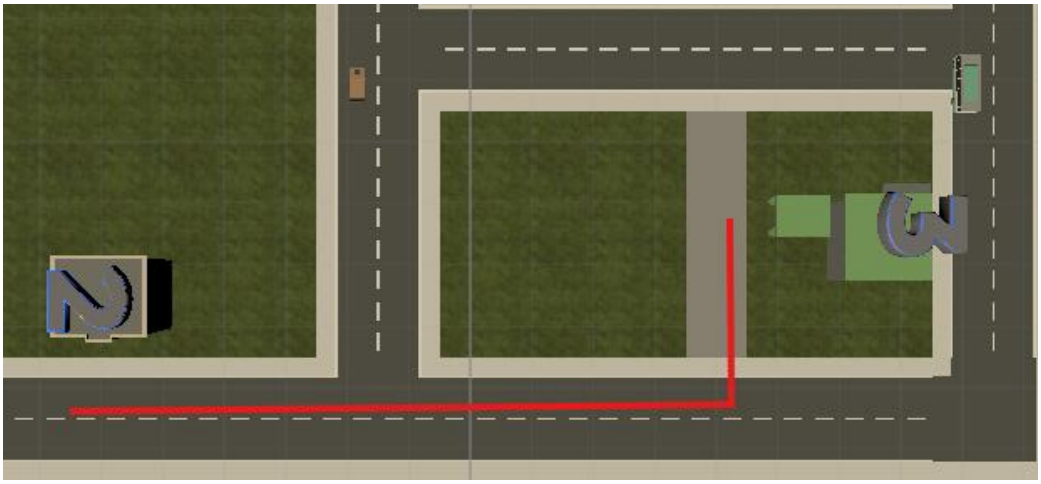
Gambar 4. 13 Titik B ke C (Uji 2)

Pada Gambar 4. 14, pengujian ketiga pada jalur B ke C memperkenalkan hambatan tambahan, yaitu lampu merah yang ditunjukkan pada gambar persegi berwarna hijau dipersimpangan jalan serta lampu merah tersebut disimulasikan bersamaan dengan *obstacle festival*.



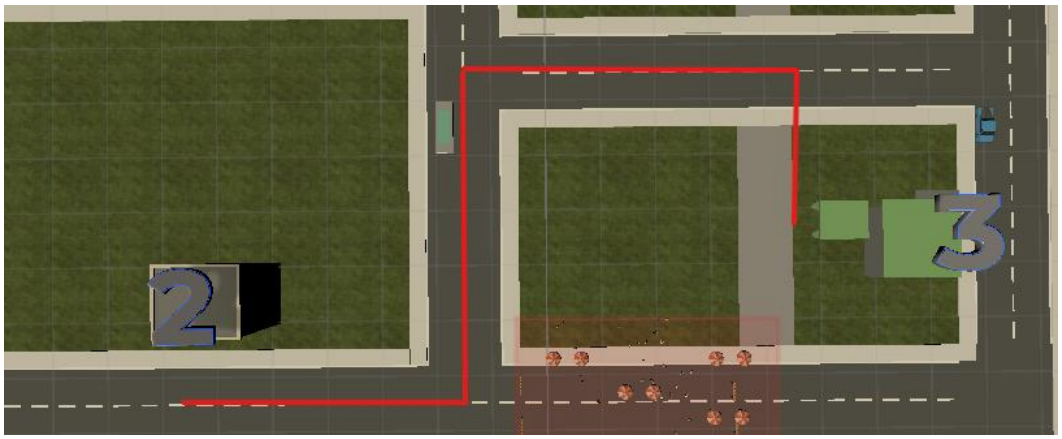
Gambar 4. 14 Titik B ke C (Uji 3)

Kemudian dapat dilihat pada Gambar 4. 15, pengujian jalur dari titik C ke D dilakukan tanpa adanya rintangan di sepanjang jalur sehingga tidak ada hambatan yang dihadapi pemain ketika hendak mencapai titik D.



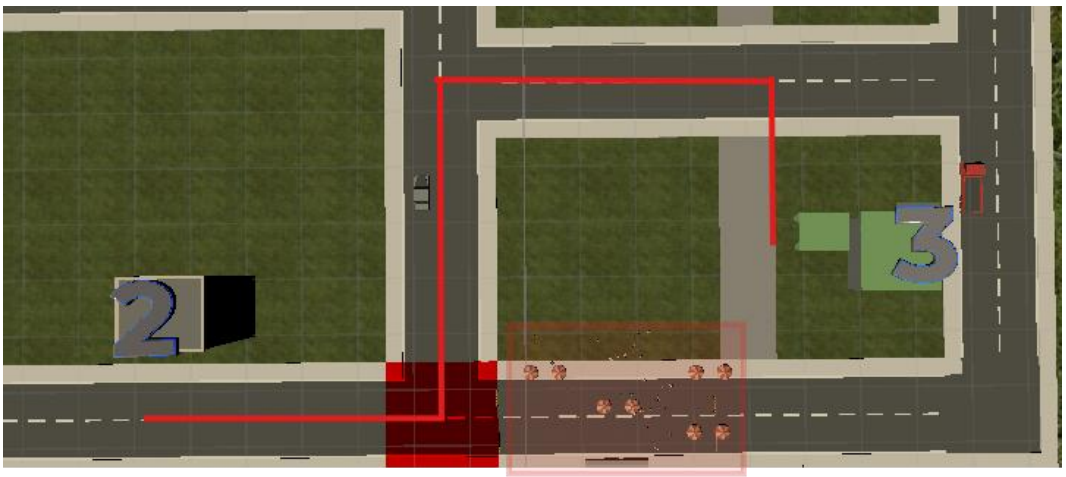
Gambar 4. 15 Titik C ke D (Uji 1)

Gambar 4. 16, menunjukkan rute yang sama dengan Gambar 4. 15 akan tetapi pada Gambar 4. 16 terdapat tambahan *static obstacle* berupa festival yang menyebabkan rute yang dilalui berbeda.



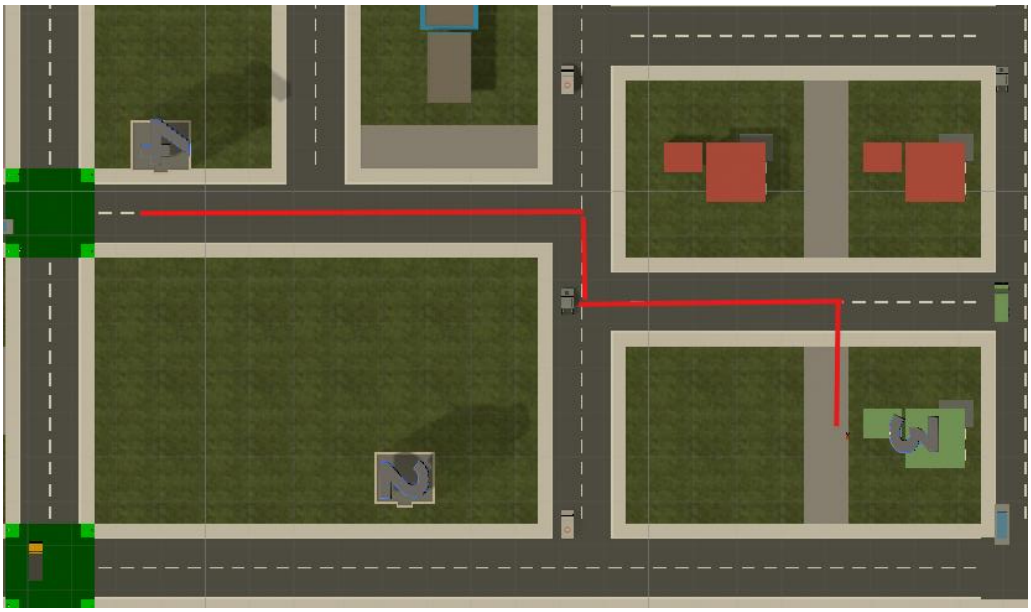
Gambar 4. 16 Titik C ke D (Uji 2)

Selanjutnya untuk Gambar 4. 17, menunjukkan rute yang sama dengan Gambar 4. 16 akan tetapi terdapat tambahan berupa *dynamic obstacle* berupa lampu merah yang berada pada persimpangan jalan.



Gambar 4. 17 Titik C ke D (Uji 3)

Gambar 4. 18 menunjukkan jalur dari titik D ke E dalam kondisi tanpa hambatan. Pengujian pertama ini bertujuan mengevaluasi rute optimal yang dihasilkan oleh A* pada kondisi ideal.



Gambar 4. 18 Titik D ke E (Uji 1)

Dalam Gambar 4. 19, jalur D ke E diuji dengan adanya *obstacle festival* yang mensimulasikan jalan yang ditutup. Pengujian ini dibuat untuk melihat efektivitas perencanaan jalur algoritma.

Gambar 4. 19 Titik D ke E (Uji 2)

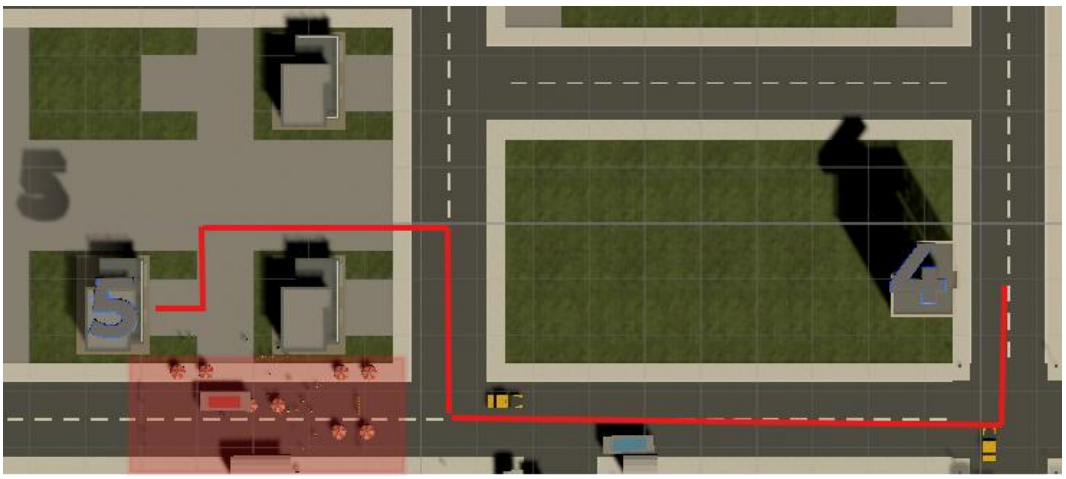
Gambar 4. 20 melanjutkan pengujian dengan menambahkan hambatan *lampu merah* bersama *obstacle festival* pada jalur D ke E. Pengujian ini dirancang untuk mengkaji bagaimana kombinasi hambatan memengaruhi efektivitas perencanaan jalur dan waktu.

Gambar 4. 20 Titik D ke E (Uji 3)

Gambar 4. 21 menggambarkan pengujian awal untuk jalur dari titik E ke F, pengujian 1 dilakukan untuk melihat kondisi ideal pencarian jalur.

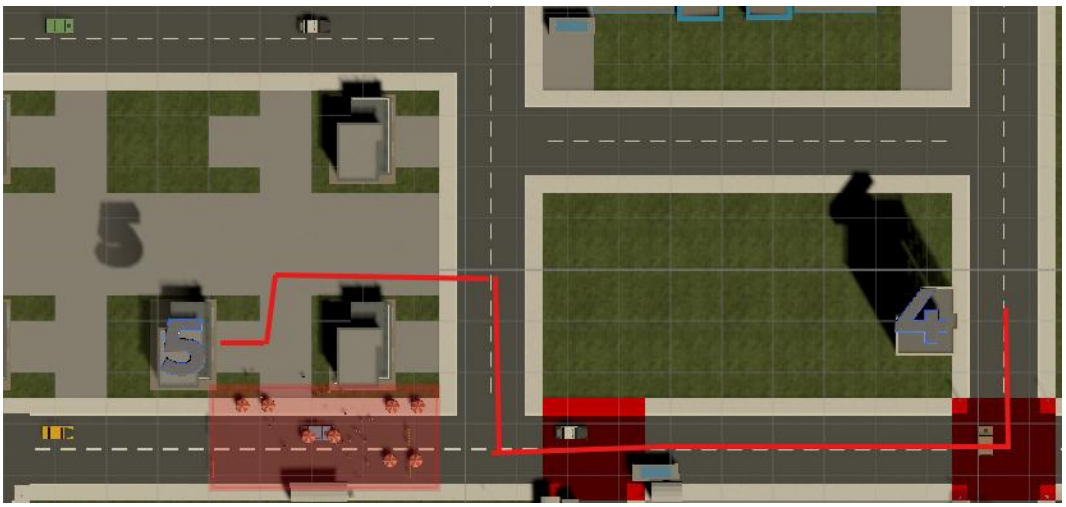
Gambar 4. 21 Titik E ke F (Uji 1)

Gambar 4. 22 menggambarkan pengujian awal untuk jalur dari titik E ke F, pengujian 2 dilakukan dengan adanya *obstacle* festifal.



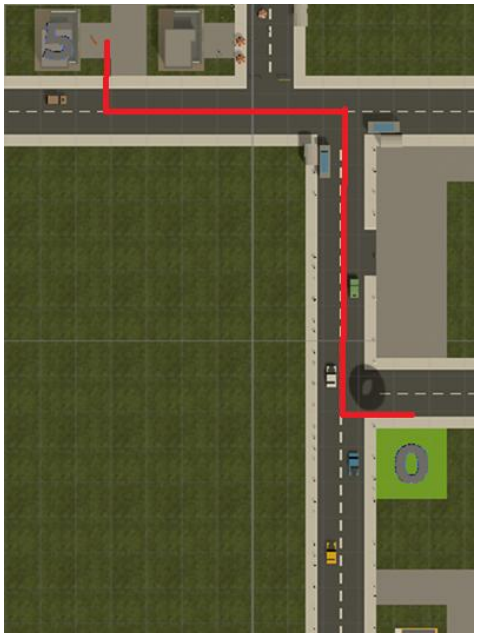
Gambar 4. 22 Titik E ke F (Uji 2)

Pada Gambar 4. 23, jalur E ke F diuji dengan adanya hambatan tambahan. Hambatan yang digunakan berupa *dynamic obstacle* yaitu lampu merah serta *obstacle static* yaitu *festival*.



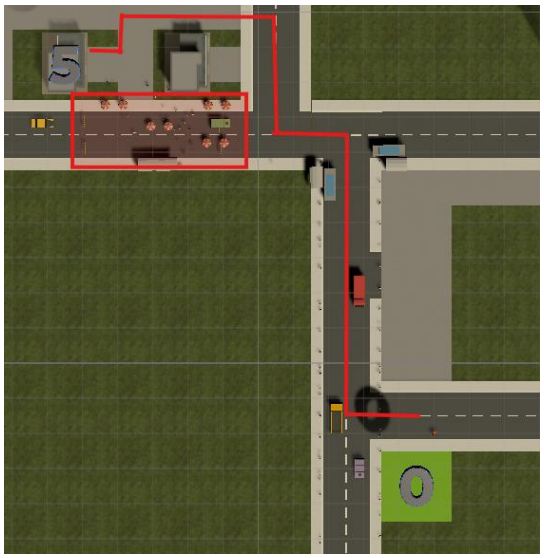
Gambar 4. 23 Titik E ke F (Uji 3)

Gambar 4. 24 memperlihatkan jalur dari titik F kembali ke titik A tanpa hambatan apapun untuk mengetahui efisiensi rute yang dihasilkan dalam kondisi optimal.



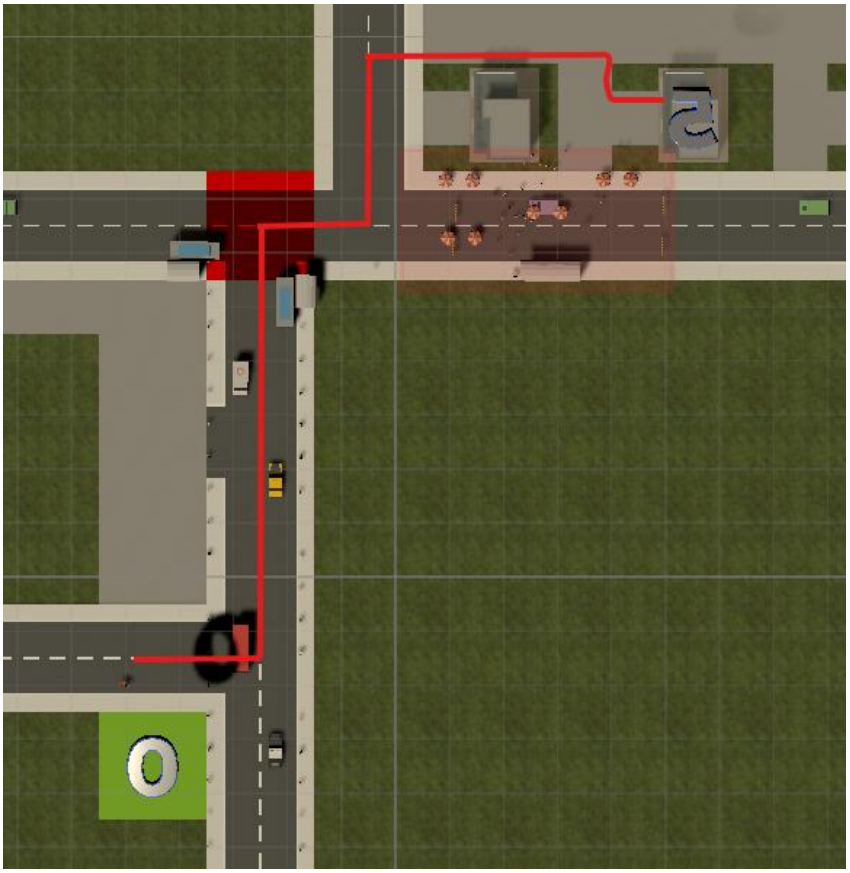
Gambar 4. 24 Titik F ke A (Uji 1)

Gambar 4. 25 Titik F ke A (Uji 2) merupakan pengujain titik F ke A dengan tambahan berupa *static obstacle* yaitu *festival*.



Gambar 4. 25 Titik F ke A (Uji 2)

Gambar 4. 26 Titik F ke A (Uji 3) tambahan kedua *obstacle* yaitu lampu merah dan penutupan jalan karena festival.



Gambar 4. 26 Titik F ke A (Uji 3)

Pengambilan data pengujian dilakukan menggunakan skrip yang dirancang untuk menghitung waktu tempuh, biaya perjalanan (*cost*), serta mempertahankan kecepatan konstan sebesar 10 *node* per detik. Tabel 4. 8 menyajikan ringkasan hasil pengujian yang mencakup total jarak tempuh dan perbandingan antara waktu tempuh aktual dengan waktu estimasi untuk setiap skenario yang diuji. total jarak dan perbandingan waktu tempuh terhadap waktu estimasi untuk setiap kasus yang diuji. Dengan demikian, dapat dievaluasi apakah algoritma berhasil menentukan jalur yang optimal.

Tabel 4. 8 Tabel Perbandingan Waktu

No	Pengujian	Jarak (*node*)	Waktu Tercatat (detik)	Waktu Ideal (t= $\frac{s}{v}$)	Selisih (detik)
A-B	UJI 1	185	18.59	18.50	+0.09
	UJI 2	210	21.15	21.00	+0.15
	UJI 3	220	28.97	22.00	+6.97
B-C	UJI 1	205	20.61	20.50	+0.11
	UJI 2	350	35.22	35.00	+0.22
	UJI 3	345	39.54	34.50	+5.04
C-D	UJI 1	130	13.12	13.00	+0.12
	UJI 2	165	16.53	16.50	+0.03
	UJI 3	175	25.96	17.50	+8.46
D-E	UJI 1	215	21.65	21.50	+0.15
	UJI 2	285	28.75	28.50	+0.25
	UJI 3	290	39.25	29.00	+10.25
E-F	UJI 1	180	18.10	18.00	+0.10
	UJI 2	205	20.59	20.05	+0.54
	UJI 3	200	37.16	20.00	+17.16
F-A	UJI 1	190	19.10	19.00	+0.10
	UJI 2	215	21.65	21.50	+0.15
	UJI 3	220	31.39	22.00	+9.39

Ket:
UJI 1 : Pengujian tanpa *obstacle.*
UJI 2 : Pengujian dengan *static obstacle*
UJI 3 : Pengujian dengan *dynamic* dan *static obstacle*

Menurut (Numeroso, dkk., 2022) algoritma A* yang menggunakan *heuristik admissible* dan *consistent* memenuhi syarat untuk menjadi algoritma dengan biaya jarak optimal (*cost-optimal*). Namun, optimalitas biaya ini tidak selalu berbanding lurus Costa dkk., (2015) dengan optimalitas waktu, terutama dalam konteks lingkungan dunia yang dinamis. Oleh karena itu, penelitian ini juga mempertimbangkan evaluasi terhadap waktu tempuh aktual yang diperoleh dari pengujian.

Berdasarkan hasil pengujian pada enam segmen jalur (A–B, B–C, C–D, D–E, E–F, dan F–A), ditemukan pada pengujian 1 terdapat rata-rata selisih waktu tempuh sebesar 0.11 detik dibanding waktu ideal, sedangkan pada pengujian 2

terdapat rata-rata selish 0.22 dan pada pengujian 3 memiliki rata-rata selisih 9.55 detik.

Kondisi uji kedua dan ketiga menunjukkan bahwa hambatan seperti lampu merah atau rintangan buatan berdampak signifikan terhadap waktu tempuh actual. Hal ini menunjukkan bahwa meskipun algoritma A* menjamin *cost-optimality* (Numeroso, dkk., 2022), adanya hambatan seperti lampu merah menyebabkan hasilnya tidak selalu *time-optimal*. Hal ini sejalan dengan temuan Costa dkk., (2015) yang menyatakan bahwa dalam jaringan jalan *time-dependent*, jalur dengan biaya terendah tidak selalu menghasilkan waktu tempuh tercepat. Oleh karena itu, penting untuk membedakan antara solusi *cost-optimal* dan *time-optimal* dalam perencanaan rute di lingkungan yang dinamis.

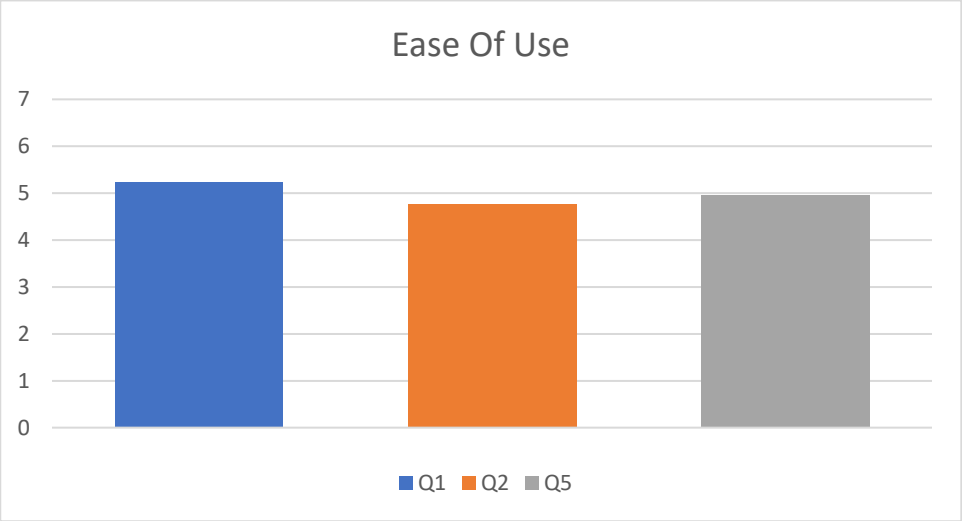
4.7 Pengujian *Usability*

Pengujian *Usability* dilakukan dengan menggunakan kuesioner dari GUESS yang difokuskan hanya pada aspek *usability*, hal tersebut dikarenakan pengembangan mekanik pergerakan dalam *game* cenderung memengaruhi aspek *usability* saja, seperti kemudahan navigasi dan kontrol, tanpa memberikan dampak signifikan terhadap aspek lain seperti *enjoyment, creativity freedom, personal gratification.* Pengujian dilakukan pada 25 orang yang belum pernah memainkan *game*. Setiap pemain diminta untuk memainkan *game* selama 20 menit yang selanjutnya dilakukan pengisian kuesioner yang disusun berdasarkan pertanyaan metode GUESS yang telah ditunjukkan pada Tabel 3. 3

Tabel 4. 9 Hasil Pengujian *Usability*

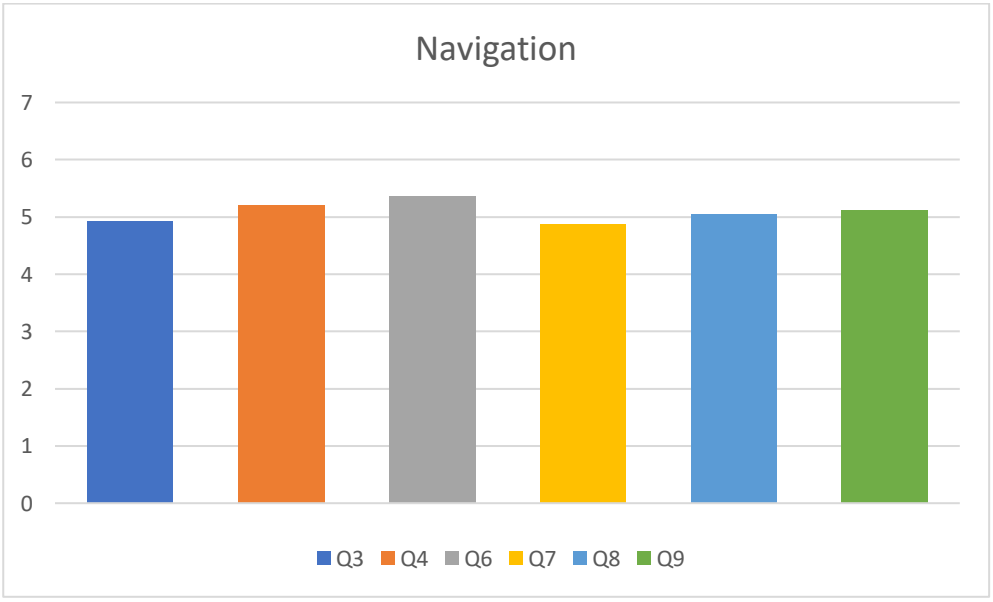
Kode	Pertanyaan	Rata Rata	Klasifikasi
Q1	I think it is easy to learn how to play the *game*	5,24	EOUS
Q2	I find the controls of the *game* to be straightforward.	4,76	EOUS
Q3	I always know how to achieve my goals/objectives in the *game*.	4,92	NAV
Q4	I find the *game*'s interface to be easy to navigate	5,2	NAV
Q5	I do not need to go through a lengthy tutorial or read a manual to play the *game*.	4,96	EOUS
Q6	I find the *game*'s menus to be user friendly	5,36	NAV
Q7	I feel the *game* trains me well in all of the controls.	4,88	NAV
Q8	I always know my next goal when I finish an event in the *game*.	5,04	NAV
Q9	I feel the *game* provides me the necessary information to accomplish a goal within the *game*.	5,12	NAV
Q10	I think the information provided in the *game* (e.g., onscreen messages, help) is clear.	5,52	UNS
Q11	I feel very confident while playing the *game*.	5,28	UNS

Kuesioner GUESS pada aspek *usability* mencakup 11 pernyataan dengan skala Likert 1–7. Hasil pengukuran menunjukkan bahwa skor rata-rata untuk setiap item berada di atas angka 4,5. Hal ini menunjukkan bahwa pengguna merasa puas terhadap kemudahan penggunaan sistem dalam *game*, serta menandakan bahwa antarmuka dan mekanisme interaksi telah dinilai intuitif dan mudah dipahami.



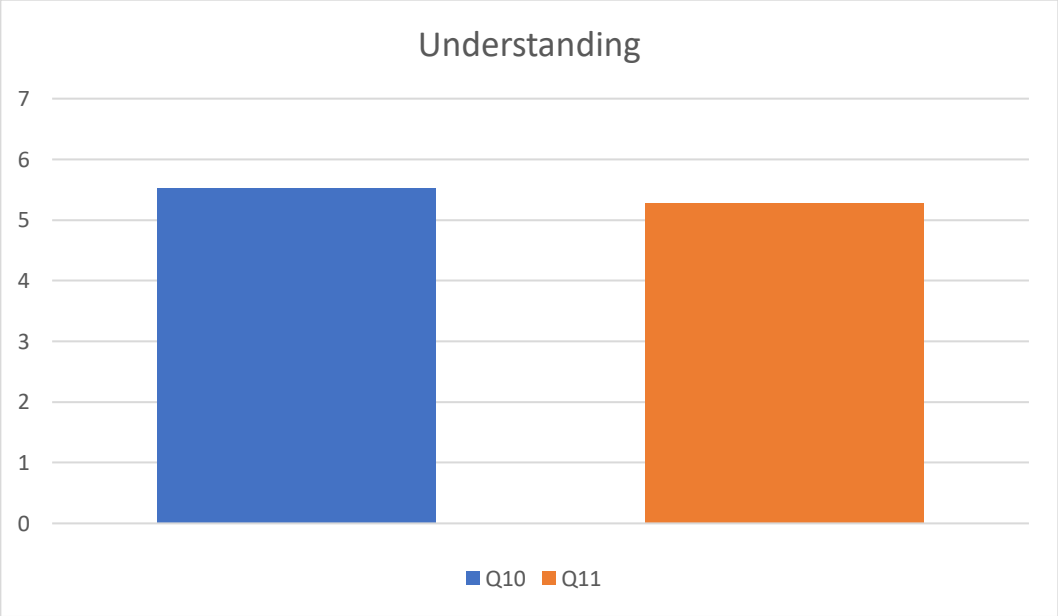
Gambar 4. 27 Diagram *Ease Of Use*

Sebagaimana ditampilkan pada Gambar 4. 27, aspek *Ease of Use* (EOUS) memperoleh skor yang tinggi. Hal ini ditunjukkan oleh pernyataan Q1: "*I think it is easy to learn how to play the game*" yang mendapatkan skor rata-rata sebesar 5,24. Namun, skor terendah pada kategori ini terdapat pada Q2: "*I find the controls of the game to be straightforward*" dengan nilai 4,76. Nilai ini dapat menjadi perhatian untuk perbaikan sistem kontrol di masa mendatang.



Gambar 4. 28 Diagram *Navigation*

Selanjutnya, aspek *Navigation* (NAV) sebagaimana diperlihatkan pada Gambar 4. 28, juga menunjukkan hasil yang baik. Skor tertinggi diperoleh dari pernyataan Q6: "*I find the game's menus to be user friendly*" dengan rata-rata 5,36, sementara skor terendah berasal dari Q3: "*I always know how to achieve my goals/objectives in the game*" dengan nilai 4,92. Hal ini menunjukkan bahwa sistem navigasi secara umum sudah membantu pemain, meskipun masih ada ruang untuk peningkatan kejelasan tujuan.



Gambar 4. 29 Diagram *Understanding*

Sebagaimana diperlihatkan pada Gambar 4. 28 Diagram *Understanding* (UNS), skor tertinggi juga dapat dilihat dari pernyataan Q10: "*I think the information provided in the game (e.g., onscreen messages, help) is clear*" dengan skor 5,52. Nilai ini memperkuat bahwa informasi yang disajikan dalam *game* mampu meningkatkan rasa percaya diri pemain selama bermain.

Berdasarkan skor rata-rata keseluruhan dari GUESS diperoleh skor sebesar 5,12 dari 7 sehingga diperoleh persentase skor sebesar 73%, menurut Tabel 3. 4 persentase skor tersebut termasuk ke dalam klasifikasi “Baik”, Hal ini menunjukkan bahwa sistem navigasi dalam game telah berhasil memberikan pengalaman pengguna yang positif, terutama dalam aspek kemudahan penggunaan, kejelasan tujuan, dan kepercayaan diri pemain. Hasil pengujian usability juga menunjukkan bahwa pengguna merasa game mudah digunakan, navigasi jelas, dan informasi yang disajikan mudah dipahami. Selain itu penggabungan Algoritma *Brute Force* dan A* mampu diimplementasikan secara efektif dalam skenario *gameplay* tanpa mengganggu kenyamanan pengguna.

4.8 Integrasi dengan Islam

Penelitian ini disusun dengan mengintegrasikan nilai-nilai Islam, khususnya dalam konsep muamalah, yang mencakup hubungan manusia dengan Allah (muamalah ma’a Allah) dan hubungan manusia dengan sesama (muamalah ma’a an-nas). Integrasi ini bertujuan untuk memberikan dimensi spiritual dan etis terhadap pengembangan teknologi, khususnya dalam merancang sistem yang bermanfaat dan memberikan kemudahan bagi penggunanya. Dalam Islam, setiap perbuatan manusia akan dipertanggungjawabkan kepada Allah SWT. Hal ini tergambar dalam firman Allah dalam QS. Al-Baqarah ayat 185:

شَهْرُ رَمَضَانَ الَّذِيْٓ اُنْزِلَ فِيْهِ الْقُرْاٰنُ هُدًى لِّلنَّاسِ وَبَيِّنٰتٍ مِّنَ الْهُدٰى وَالْفُرْقَانِۚ فَمَنْ شَهِدَ مِنْكُمُ الشَّهْرَ فَلْيَصُمْهُۗ وَمَنْ كَانَ مَرِيْضًا اَوْ عَلٰى سَفَرٍ فَعِدَّةٌ مِّنْ اَيَّامٍ اُخَرَۗ يُرِيْدُ اللّٰهُ بِكُمُ الْيُسْرَ وَلَا يُرِيْدُ بِكُمُ الْعُسْرَۖ وَلِتُكْمِلُوا الْعِدَّةَ وَلِتُكَبِّرُوا اللّٰهَ عَلٰى مَا هَدٰىكُمْ وَلَعَلَّكُمْ تَشْكُرُوْنَ

“Bulan Ramadan adalah (bulan) yang di dalamnya diturunkan Al-Qur’an sebagai petunjuk bagi manusia dan penjelasan-penjelasan mengenai petunjuk itu serta pembeda (antara yang hak dan yang batil). Oleh karena itu, siapa di antara kamu

hadir (di tempat tinggalnya atau bukan musafir) pada bulan itu, berpuasalah. Siapa yang sakit atau dalam perjalanan (lalu tidak berpuasa), maka (wajib menggantinya) sebanyak hari (yang ditinggalkannya) pada hari-hari yang lain. Allah menghendaki kemudahan bagimu dan tidak menghendaki kesukaran. Hendaklah kamu mencukupkan bilangannya dan mengagungkan Allah atas petunjuk-Nya yang diberikan kepadamu agar kamu bersyukur." (QS. Al-Baqarah: 185).

Tafsir Wajiz, "Bulan Ramadhan adalah bulan yang di dalamnya untuk pertama kali diturunkan Al-Qur'an pada lailatul qadar, yaitu malam kemuliaan, sebagai petunjuk bagi manusia dan penjelasan-penjelasan mengenai petunjuk itu dan pembeda antara yang benar dan yang salah. Karena itu, barangsiapa di antara kamu ada, yakni hidup, di bulan itu dalam keadaan sudah akil baligh, maka berpuasalah. Dan barangsiapa yang sakit di antara kamu atau dalam perjalanan lalu memilih untuk tidak berpuasa, maka ia wajib menggantinya sebanyak hari yang ditinggalkannya itu pada hari-hari yang lain. Allah menghendaki kemudahan bagimu dengan membolehkan berbuka, dan tidak menghendaki kesukaran bagimu dengan tetap mewajibkan puasa dalam keadaan sakit atau dalam perjalanan. Hendaklah kamu mencukupkan bilangannya dengan berpuasa satu bulan penuh dan mengakhiri puasa dengan bertakbir mengagungkan Allah atas petunjuk-Nya yang diberikan kepadamu, agar kamu bersyukur atasnya."(*Surat Al-Baqarah Ayat 185: Arab, Latin, Terjemah Dan Tafsir Lengkap*, n.d.)

Ayat tersebut menekankan bahwa Allah SWT menginginkan kemudahan bagi hamba-Nya. Hal ini selaras dengan tujuan dari penelitian ini, yaitu menciptakan sistem yang memberikan kemudahan bagi pengguna dalam menyelesaikan permasalahan dalam game simulasi, dengan pendekatan algoritma seperti TSP dan A* untuk efisiensi dan efektivitas.

Selain itu, Islam juga sangat menekankan pentingnya membantu sesama manusia. Hal ini termasuk kedalam konsep Muammalah Ma'a An-Nas. Seperti ditegaskan oleh Rasulullah SAW dalam sebuah hadits yang diriwayatkan oleh Imam Muslim dari Abu Hurairah:

مَنْ نَفَّسَ عَنْ مُؤْمِنٍ كُرْبَةً مِنْ كُرَبِ الدُّنْيَا نَفَّسَ اللَّهُ عَنْهُ كُرْبَةً مِنْ كُرَبِ يَوْمِ الْقِيَامَةِ وَمَنْ يَسَّرَ عَلَى مُعْسِرٍ يَسَّرَ اللَّهُ عَلَيْهِ فِي الدُّنْيَا وَالآخِرَةِ

"*Siapa yang menyelesaikan kesulitan seorang mukmin dari berbagai kesulitan-kesulitan dunia, niscaya Allah akan memudahkan kesulitan-kesulitannya pada hari kiamat. Siapa yang memudahkan orang yang sedang kesulitan niscaya Allah mudahkan baginya di dunia dan akhirat*." (Khutbah Jumat, Kemenag, 2023)

Hadis ini memperkuat nilai bahwa memberikan kemudahan kepada orang lain adalah tindakan mulia yang akan dibalas oleh Allah, baik di dunia maupun di akhirat. Dalam konteks penelitian ini, hal tersebut menjadi landasan moral untuk merancang sistem yang tidak hanya efisien secara teknis, tetapi juga mencerminkan nilai-nilai kebaikan dan keikhlasan dalam membantu orang lain.

Dengan demikian, penggabungan antara nilai spiritual dan sosial dalam Islam menjadi kerangka etik yang mendasari pengembangan sistem ini. Sistem yang dikembangkan diharapkan tidak hanya memberikan solusi teknis terhadap tantangan dalam game, tetapi juga membawa manfaat yang lebih luas dengan semangat kemudahan dan pertolongan sebagaimana diajarkan dalam Islam.

BAB V

KESIMPULAN DAN SARAN

5.1 Kesimpulan

Penelitian ini telah berhasil menerapkan Algoritma *Brute Force* dan A* pada *Traveling Salesman Problem* dalam *game CoolCity*, *game* simulasi dengan tantangan multi-objektif. Penerapan algoritma ini menunjukkan hasil yang sesuai dengan perhitungan manual dan dapat meningkatkan efisiensi pemain dalam menyelesaikan permainan. Efek dari implementasi kedua algoritma ini adalah rekomendasikan rute dengan cost optimal kepada pemain, sehingga pemain dapat menyelesaikan misi dengan perencanaan yang lebih baik dan strategi yang terarah. Algoritma *Brute Force* digunakan untuk menentukan urutan kunjungan lokasi dengan cost total minimum melalui eksplorasi seluruh kemungkinan rute. Berdasarkan hasil pengujian, rute yang dihasilkan Algoritma *Brute Force* sesuai dengan hasil hitung manual.

Algoritma A* digunakan untuk menghitung jalur terpendek antar lokasi dalam setiap skenario permainan. Hasil pengujian menunjukkan rata-rata selisih waktu tempuh aktual terhadap waktu ideal sebesar 0,11 detik pada skenario pertama (tanpa hambatan), 0,22 detik pada skenario kedua (dengan hambatan statis), dan 9,55 detik pada skenario ketiga (dengan rintangan statis dan dinamis). Temuan ini menunjukkan bahwa meskipun algoritma A* menjamin *cost-optimality*, kehadiran hambatan dinamis menyebabkan hasilnya tidak selalu *time-optimal*.

Berdasarkan hasil uji usability menggunakan pendekatan GUESS, diperoleh skor rata-rata sebesar 5,12 dari skala 7. Nilai ini menunjukkan bahwa *game CoolCity* memiliki tingkat keterpahaman, kemudahan navigasi, dan kepercayaan diri pengguna yang tinggi, serta termasuk dalam kategori "baik" dan "berhasil" dalam aspek kenyamanan dan pengalaman pengguna

5.2 Saran

Sebagai tindak lanjut dari penelitian ini, disarankan agar pengembangan sistem ke depan mempertimbangkan graf dinamis dimana waktu merupakan salah satu varibel seperti, adanya rintangan atau perubahan kondisi jalan sehingga integrasi algoritma optimasi berbasis waktu seperti *Time-Dependent A** atau algoritma heuristik lainnya dapat mendapatkan solusi *Time-Optimal*. Selain itu, Algoritma *Brute Force* memiliki keterbatasan dalam skalabilitas karena kompleksitas komputasi yang tinggi, penelitian lanjutan dapat mengimplementasikan pendekatan yang lebih efisien seperti *Ant Colony Optimization* atau *Genetic Algorithm* untuk menangani jumlah titik yang lebih besar.

Dari sisi antarmuka dan kenyamanan pengguna, perolehan skor terendah pada aspek kontrol menunjukkan perlunya perbaikan pada desain interaksi dan mekanisme input agar lebih intuitif bagi pengguna baru. Disarankan juga untuk menambah variasi skenario *game*play dan tsantangan agar pengalaman bermain lebih dinamis dan menantang. Terakhir, pengembangan fitur kompetitif seperti *leaderboard* atau *mode multiplayer* berbasis waktu dapat menjadi nilai tambah untuk meningkatkan keterlibatan dan motivasi pemain dalam menyelesaikan objektif secara optimal

DAFTAR PUSTAKA

Amelia, M., Sholeha, I. I., Revangga, Y. R., & Wamiliana. (2024). The Comparison of Brute Force, Cheapest-Insertion, and Nearest-Neighbor Heuristics for Determining the Shortest Tour for Visiting Malls in Bandar Lampung. *EXPERT: Jurnal Manajemen Sistem Informasi dan Teknologi*, *14*(1), 51. https://doi.org/10.36448/expert.v14i1.3715

Arif, Y. M., Nurhayati, H., Karami, A. F., Nugroho, F., Kurniawan, F., Rasyid, H. A., … Garcia, M. B. (2023). An Artificial Neural Network-Based Finite State Machine for Adaptive Scenario Selection in Serious Game. *International Journal of Intelligent Engineering and Systems*, *16*(5), 488–500. https://doi.org/10.22266/ijies2023.1031.42

Ayu Nugraeni, R., Mulyono, & Rochmad. (2015). Penerapan Algoritma A* Dalam Penyelesaian Rute Terpendek Pendistribusian Barang. *UNNES Journal of Mathematics (UJM)*, *4*(1), 7–12. Diambil dari http://journal.unnes.ac.id/sju/index.php/ujm

Bahtiar, A., Muhima, R. R., & Rachman, A. (2019). Penerapan Model Spiral Pada Rancang Bangun Game Platformer. *Seminar Nasional Sains dan Teknologi Terapan VII 2019*, 601–606. Institut Teknologi Adhi Tama Surabaya.

Baidoo, E., & O., S. (2016). Solving the TSP using Traditional Computing Approach. *International Journal of Computer Applications*, *152*(8), 13–19. https://doi.org/10.5120/ijca2016911906

Brown Requist, K., & Momayez, M. (2024). Minimum Cost Pathfinding Algorithm for the Determination of Optimal Paths under Airflow Constraints. *Mining*, *4*(2), 429–446. https://doi.org/10.3390/mining4020025

Burnes, A., & Siman, G. (2024, Juni 2). Introducing Project G-Assist: A Preview Of How AI Assistants Can Enhance Games & Apps. In NVIDIA. Diambil 26 April 2024, dari https://www.nvidia.com/en-us/geforce/news/g-assist-ai-assistant/

Costa, C. F., Nascimento, M. A., Macêdo, J. A. F., Theodoridis, Y., Pelekis, N., & Machado, J. (2015). Optimal time-dependent sequenced route queries in road networks. *GIS: Proceedings of the ACM International Symposium on Advances in Geographic Information Systems*, *03-06-November-2015*. Association for Computing Machinery. https://doi.org/10.1145/2820783.2820846

Danil Risaldi, Megawati, Idria Maita, & Febi Nur Salisah. (2024). Evaluasi User Experience Pada Game Playerunknown's Battlegrounds (Pubg) Mobile Menggunakan Metode Heuristic Evaluation. *The Indonesian Journal of Computer Science*, *13*(6). https://doi.org/10.33022/ijcs.v13i6.4468

Diharjo, W., Sani, D. A., & Arif, M. F. (2020). *Game Edukasi Bahasa Indonesia Menggunakan Metode Fisher Yates Shuffle Pada Genre Puzzle Game*.

Faisal, M., Nurhayati, H., Arif, Y. M., Kurniawan, F., & Nugroho, F. (2015). Immersive Bicycle Game For Health Virtual Tour Of Uin Maulana Malik Ibrahim Malang. *Jurnal Teknologi*, *78*(5), 325–328. Diambil dari www.jurnalteknologi.utm.my

Fan, G., Zhang, C., Wang, K., Li, Y., Chen, J., & Xu, Z. (2024). CUPID: Improving Battle Fairness and Position Satisfaction in Online MOBA Games with a Re-matchmaking System. *Proceedings of the ACM on Human-Computer Interaction*, *8*(CSCW2), 1–39. https://doi.org/10.1145/3686978

Hasan Syu'aibi, M., Mahmudi, A., & Auliasari, K. (2023). Perancangan Dan Implementasi Metode Fsm (Finite State Machine) Pada Game Military Defence 2D Berbasis Android. Dalam *Jurnal Mahasiswa Teknik Informatika* (Vol. 7).

Hasniati, Rahman, S., & Suyono. (2023). Penerapan Algoritma A* untuk Pencarian Rute Terpendek Pada Aplikasi Game Shoot'em Up. *Journal of Artificial Intelligence and Data Science (AGENTS)*, *3*(2), 1–6.

Hayes, B. (2017). Distribution Generation Optimization and Energy Management. Dalam G.B. Gharehpetian & S. Mohammad Mousavi Agah (Ed.), *Distributed Generation Systems: Design, Operation and Grid Integration* (hlm. 415–451). Butterworth-Heinemann: Elsevier. https://doi.org/10.1016/B978-0-12-804208-3.00009-1

Hendy. (2021). *Penerapan Line of Sight Dan Finit State Machine Pada Game Platformer "run!"* (STMIK Widya Cipta Dharma). STMIK Widya Cipta Dharma. Diambil dari http://repository.wicida.ac.id/id/eprint/3420

Hermawan, H., & Setiyani, H. (2019). Implementasi Algoritma A-star Pada Permainan Komputer Roguelike. *Jurnal Algoritma, Logika dan Komputasi*, *2*(1), 111–120. https://doi.org/10.30813/j-alu.v2i1.1571

Indriyono, B. V., & Widyatmoko. (2021). Optimization of Breadth-First Search Algorithm for Path Solutions in Mazyin Games. *International Journal of Artificial Intelligence & Robotics (IJAIR)*, *3*(2), 58–66. https://doi.org/10.25139/ijair.v3i2.4256

Irawan, C. D., Mamahit, D. J., & Sambul, A. M. (2019). Pembuatan Game Simulasi Kewirausahaan untuk Profesi Petani. *Jurnal Teknik Informatika*, *14*(1), 53–62.

Iskandar, U. A. S., Diah, N. M., & Ismail, M. (2020). Identifying Artificial Intelligence Pathfinding Algorithms for Platformer Games. *2020 IEEE International Conference on Automatic Control and Intelligent Systems, I2CACIS 2020 - Proceedings*, 74–80. Institute of Electrical and Electronics Engineers Inc. https://doi.org/10.1109/I2CACIS49202.2020.9140177

Jagdale, D. (2021). Finite State Machine in Game Development. *International Journal of Advanced Research in Science, Communication and Technology*, *10*(1), 384–390. https://doi.org/10.48175/ijarsct-2062

Kemenag. (2023, September 27). Khutbah Jumat: Mari Mudahkan Urusan Orang Lain. Diambil 15 Maret 2025, dari https://kemenag.go.id/islam/khutbah-jumat-mari-mudahkan-urusan-orang-lain-dqzq6

Kherrour, A., Robol, M., Roveri, M., & Giorgini, P. (2023). Evaluating Heuristic Search Algorithms in Pathfinding: A Comprehensive Study on Performance Metrics and Domain Parameters. *Electronic Proceedings in Theoretical Computer Science, EPTCS*, *391*, 102–112. Open Publishing Association. https://doi.org/10.4204/EPTCS.391.12

Krafft, C. (2019). *Implementation and Comparison of Pathfinding Algorithms in a Dynamic 3D Space.* University of Applied Sciences Hamburg, Hamburg.

Leuchter, I., & Kurtz, G. (2022). Effects of Instructions, Assistance, Narrative, Competition, Challenge, and Age on Performances in Digital Learning Games. *International Journal of Advanced Corporate Learning (iJAC)*, *15*(2), 16–33. https://doi.org/10.3991/ijac.v15i2.30867

Liao, S. (2023). The Solutions to Traveling Salesman Problem. *Highlights in Science, Engineering and Technology AMMMP*, *47*.

Luo, M., Hou, X., & Yang, J. (2016). Surface Optimal Path Planning Using an Extended Dijkstra Algorithm. *IEEE Access*, *4*, 147827–147838. https://doi.org/10.1109/ACCESS.2020.3015976

Nizhar Gumilang, A., Haryanto, H., & Dolphina, E. (2023). Pengembangan Game RPG dan Story Dengan Elemen Gameplay Menggunakan Metodologi Finite State Machine (FSM) Pada Game Kisah Tjepoe Development Of RPG and Story Games With Gameplay Elements Using The Finite State Machine Methodology In The Tjepoe Story Game. *Desember*, *1*(2), 129–158.

Nugroho, F., Basid, P. M. N. S. A., Bahtiar, F. S., Simamora, R. N. Z., Kurniawan, R. F., Janitra, G. A., & Fadilah, J. N. (2022). 2D Game "Omar's Adventure" Design Using the Finite State Machine Method. *JOURNAL OF INFORMATICS AND TELECOMMUNICATION ENGINEERING*, *6*(1), 18–26. https://doi.org/10.31289/jite.v6i1.6327

Numeroso, D., Bacciu, D., & Veličković, P. (2022). *Learning Heuristics for A**. Diambil dari http://arxiv.org/abs/2204.08938

Osaba, E., Yang, X. S., & Del Ser, J. (2020). Traveling salesman problem: a perspective review of recent research and new results with bio-inspired metaheuristics. *Nature-Inspired Computation and Swarm Intelligence: Algorithms, Theory and Applications*, 135–164. https://doi.org/10.1016/B978-0-12-819714-1.00020-8

Paillin, D. B., & Tupan, J. M. (2020). Model Integer Liniear Programming (ILP) Dalam Pemecahan Traveling Salesman Problem (TSP) (Studi Kasus : PT. Paris Jaya Mandiri - Ambon). *ALE 3*, 40–47. Fakultas Teknik Universitas Pattimura - Ambon.

Phan, M. H., Keebler, J. R., & Chaparro, B. S. (2016). The Development and Validation of the Game User Experience Satisfaction Scale (GUESS). *Human Factors*, *58*(8), 1217–1247. https://doi.org/10.1177/0018720816669646

Samah, K. A. F. A., Sabri, N., Hamzah, R., Roslan, R., Mangshor, N. A., & Asri, A. A. M. (2019). Brute Force Algorithm Implementation For Traveljoy Travelling Recommendation System. *Indonesian Journal of Electrical Engineering and Computer Science*, *16*(2), 1042–1049. https://doi.org/10.11591/ijeecs.v16.i2.pp1042-1049

Satrio, I., Santi Wahyuni, F., & Rudhistiar, D. (2022). Penerapan A* Pathfinding dan FSM (Finite State Machine) Pada Game “Lost Civilization” Berbasis Android. *Jurnal Mahasiswa Teknik Informatika (JATI)*, *6*(2), 1192–1119.

Sattu, J., & Fairul Filza, M. (2025). Analisis Kemudahan Penggunaan pada Antarmuka Pengguna Game Horor GhostyGhost menggunakan System

Usability Scale. *Journal of Informatics Management and Information Technology*, *5*(1), 65–72. https://doi.org/10.47065/jimat.v5i1.452

Sehang, J. D., Tulenan, V., & Sambul, A. M. (2019). Perancangan Game Simulasi Kewirausahaan. *Jurnal Teknik Informatika*, *14*(1), 79–89.

Shafira, R. A., Yahfizham, & Harahap, A. M. (2023). Menentukan Jarak Terpendek Dalam Pengiriman Barang Dengan Perbandingan Euclidean Distance dan Manhattan Distance. *Journal of Science and Social Research*, *6*, 678–685.

Steam Chart. (2024, Juni 12). Top 100 Most Played Games. Diambil dari https://store.steampowered.com/charts/mostplayed website: https://store.steampowered.com/charts/mostplayed

Susilawati, Rizky, R., Setiyowati, S., & Pratama, A. G. (2020). Penerapan Metode A*Star Pada Pencarian Rute Tercepat Menuju Destinasi Wisata Cagar Budaya Menes Pandeglang. *Geodika: Jurnal Kajian Ilmu dan Pendidikan Geografi*, *4*(2), 192–199. https://doi.org/10.29408/geodika.v4i2.2754

Syukriyah, Y., Falahah, & Solihin, H. (2016). Penerapan Algoritma A* (Star) Untuk Mencari Rute Tercepat Dengan Hambatan. *Seminar Nasional Telekomunikasi dan Informatika (SELISIK 2016)*, 219–224. Universitas Widyatama.

Triyanto, F., Adianto, H., & Susanty, S. (2015). Usulan Rancangan Rute Distribusi Gas LPG 3 Kg Menggunakan Metode Heuristik Dan Metode Branch and Bound di PT X*. *Jurnal Online Institut Teknologi Nasional Juli*, *3*, 194–205.

Wawan Saputra, D. (2022). Optimalisasi Rute Distribusi Kurir Menggunakan Metode Traveling Salesman Problem (Studi Kasus: JNE Balige). *G-Tech:*

Jurnal Teknologi Terapan, *6*(2), 159–165. https://doi.org/10.33379/gtech.v6i2.1577

Yan, Y., & Dublin, B.-. (2023). Research on the A Star Algorithm for Finding Shortest Path. *Highlights in Science, Engineering and Technology MCEE*, *46*, 154–161.

Zuhdi, A., Ahmad, I., & Putra, A. D. (2023). Implementation Of A* Algorithm In A Great Elephant Game With Unity 2D. *SINTECH JOURNAL*, *6*(2), 118–123. https://doi.org/https://doi.org/10.31598