Pembuatan *Game* 3D Petualangan Labirin Menggunakan Algoritma *Dijkstra* pada *Non-Player Character* (NPC)

Aldi Ferdian[1], Mursyidah[2*], Guntur Syahputra[3]

[1,3] *Jurusan Teknologi Informasi dan Komputer Politeknik Negeri Lhokseumawe*
Jln. B.Aceh Medan Km.280 Buketrata 24301 INDONESIA

[1]aldiferdian998@gmail.com

[2*] mursyidah@pnl.ac.id

[3]guntur@pnl.ac.id

Abstrak— ***Game adalah media hiburan yang sangat digemari oleh semua kalangan dan dapat mengasah kreativitas orang yang memainkannya. Salah satu jenis game yang populer adalah game puzzle labirin yang menguji kecerdasan dan keterampilan player dalam menyelesaikan tantangan untuk melewati labirin. Penelitian ini bertujuan untuk membuat game 3D petualangan labirin menjadi lebih menantang dimana karakter NPC dapat mengikuti player di dalam labirin. Untuk membuat NPC mampu bergerak mengikuti player secara otomatis, dibutuhkan algoritma pencarian rute terpendek yang harus diterapkan pada NPC. Penelitian ini menggunakan algoritma Dijkstra sebagai pencari rute terpendek. Algoritma Dijkstra akan mencari jalan dengan biaya terkecil antara NPC terhadap player. Pengujian dilakukan terhadap NPC dengan menggunakan dua jalur yang akan dilalui untuk menuju posisi player dengan panjang path yang berbeda. Berdasarkan pengujian yang dilakukan sebanyak 20 kali, dengan metode Dijkstra maka didapatkan hasil pengujian sebesar 100%.***

Kata kunci— *Game, Labirin, Pathfinding, Dijkstra*

Abstract*— Games are entertainment media that are very popular with everybody and can hone the creativity of the people who play them. One popular type of game is the maze puzzle game that tests the player's intelligence and skill in solving challenges to get through the maze. This research aims to create a more challenging 3D maze adventure game where NPC characters can follow the player in the maze. In order to make the NPC able to move following the player automatically, a shortest route finding algorithm is required to be implemented on the NPC. Dijkstra's algorithm will find the path of least cost between the NPC and the player. Tests were carried out on the NPC using two paths to be traveled to reach the player position with different path lengths. Based on tests carried out 20 times, with the Dijkstra method, the test results are 100%.

Keywords— *Game, Maze, Pathfinding, Dijkstra*

I. Pendahuluan

Game adalah media hiburan yang digemari semua kalangan mulai anak, remaja, dewasa bahkan orang tua. *Game* juga dapat mengasah kreatifitas orang yang memainkannya, saat bermain *game* terdapat peraturan yang membatasi *player* dalam bertindak dan membuat *player* harus berpikir serta mencari solusi untuk menyelasaikan masalah yang ada dalam *game*[1]. *Game puzzle* labirin adalah jenis *game* yang menguji kecerdasan, keterampilan, dan ketelitian *player* dalam memecahkan teka-teki dan menyelesaikan tantangan dalam melewati labirin. Pada dasarnya, *game* ini membutuhkan *player* untuk menggerakkan karakter atau objek dari titik awal menuju titik akhir dalam labirin yang penuh dengan rintangan dan jebakan yang harus dihindari[2]. Namun dibalik pembuatan *game* terdapat sebuah Articifial Intelligence (AI) atau kecerdasan buatan yang ditanamkan pada *game* untuk membuat *game* tersebut menjadi lebih pintar. Salah satu kecerdasan yang digunakan di dalam *game* adalah mencari jalur terpendek yang ditanamkan untuk NPC pada *game puzzle*[3]. Salah satu algoritma yang sering digunakan dalam pembuatan *game*, Proses mencari jalan terpendek tersebut menggunakan algoritma *Dijkstra* yang merupakan algoritma yang digunakan untuk mencari jalur terpendek antara dua simpul pada sebuah graf berbobot[4].

Pada *game* labirin, terdapat berbagai NPC yang memiliki peran penting dalam menghidupkan dunia permainan. Namun, banyak dari NPC tersebut cenderung hanya berdiam diri atau bergerak dalam pola yang telah ditentukan sebelumnya. Meskipun peran mereka dapat memberikan nuansa kehidupan pada lingkungan dalam permainan, terkadang pola pergerakan yang statis ini dapat mengurangi tantangan dalam bermain. Ketika NPC hanya bergerak atau berinteraksi dalam pola yang sudah ditentukan, hal ini bisa membuat permainan mudah di tebak[1].

Untuk mengatasi masalah ini, Peneliti ingin memberikan variasi dalam pola pergerakan NPC. Salah satunya dengan mengadopsi sistem kecerdasan buatan yaitu dengan menerapkan algoritma *Dijkstra* yang memungkinkan NPC untuk bergeraka mengikuti karakter pemain. Dengan demikian, permaina tidak akan mudah untuk dimainkan.

A. Game

Game adalah suatu permainan yang dimainkan dengan cara tertentu dan teratur, biasanya dilakukan untuk tujuan hiburan atau kompetisi. *Game* dapat dimainkan dalam berbagai bentuk dan jenis, termasuk *game* komputer, *game* konsol, *game* ponsel, *game* papan, dan *game* olahraga. Sejarah *game* dapat ditelusuri kembali ke beberapa ribu tahun yang lalu, ketika manusia mulai membuat permainan sederhana seperti catur dan dadu. Namun, *game* modern seperti yang kita kenal sekarang ini, berkembang pada abad ke-20 dengan munculnya teknologi dan industri *game* yang berkembang pesat. Industri *game* modern dimulai pada tahun 1970-an, ketika mesin arcade pertama kali muncul. Pada dekade berikutnya, *game* konsol mulai populer, seperti Atari dan Nintendo. Pada tahun 1990-an, *game* komputer mulai mendominasi pasar *game*, dan pada akhir 1990-an, *game online* dan *game* ponsel mulai berkembang[5].

B. Game 3D

Game 3D adalah jenis permainan video yang memanfaatkan teknologi tiga dimensi (3D) untuk membuat lingkungan dan karakter yang lebih realistis. Dalam *game* 3D, lingkungan, karakter, dan objek dalam permainan dibuat dengan menggunakan model 3D, sehingga memiliki tampilan yang lebih menyerupai dunia nyata. Dalam *game* 3D, *player* dapat melihat dunia permainan dari berbagai sudut pandang dan perspektif yang berbeda. Hal ini disebabkan oleh kemampuan teknologi 3D dalam membuat objek dan lingkungan yang memiliki kedalaman, lebar, dan tinggi, sehingga memungkinkan *player* untuk merasakan sensasi ruang dan gerakan dalam *game*[6].

C. *Game Online*

Game online adalah jenis permainan yang dimainkan melalui jaringan internet. Dalam *game online*, pemain dapat terhubung dengan pemain lain dari berbagai lokasi geografis yang berbeda. Hal ini memungkinkan para pemain untuk berinteraksi, bekerjasama, atau bersaing dalam lingkungan virtual yang sama. *Game online* dapat dimainkan melalui berbagai platform, termasuk komputer, konsol *game*, dan perangkat seluler. Beberapa *game online* populer termasuk permainan peran (RPG), penembak orang pertama (FPS), strategi waktu nyata (RTS), dan *game battle royale*. Dalam *game online*, pemain sering kali memiliki avatar atau karakter yang dapat dikendalikan di dalam permainan. Pemain dapat memilih karakter, meningkatkan keterampilan atau kemampuan mereka, dan berinteraksi dengan pemain lain melalui chat atau fitur lainnya[7].

D. *Multiplayer*

Game multiplayer adalah jenis permainan yang memungkinkan beberapa pemain bermain bersama dalam satu permainan yang sama. Dalam *game multiplayer*, pemain dapat berinteraksi, bekerjasama, atau bersaing satu sama lain secara langsung. *Game multiplayer* memberikan kesempatan bagi pemain untuk berinteraksi dengan pemain lain, meningkatkan aspek sosial permainan, dan memberikan pengalaman yang lebih dinamis dan menantang. Baik itu dalam bentuk kerja sama tim atau persaingan langsung, *game multiplayer* sering kali memberikan pengalaman bermain yang lebih seru dan mendalam[8].

E. Normcore

Normcore adalah perusahaan yang membuat plugin Unity yang digunakan untuk membuat pengalaman *multiplayer* atau *game* dengan fokus yang lebih berat seperti *Virtual-Reality* (VR). Normcore menyediakan plugin untuk menambahkan *multiplayer* secara real-time pada *game* Unity. Normcore memperkenalkan konsep datastore. Semua keadaan, baik itu posisi *player* atau skor permainan, disimpan di datastore. Perbarui posisinya di datastore dan Normcore akan menyinkronkan perubahan ke *player* lain secara otomatis. Fast Transport: Normcore menggunakan mekanisme transportasi berpemilik untuk mendapatkan paket antara client dan *server*. Melalui penggunaan datastore, kontrol aliran, dan fragmentasi paket cerdas, Normcore secara dinamis menentukan kapan harus mengirim pembaruan dan cara memecahnya menjadi beberapa paket untuk mengurangi waktu yang diperlukan paket untuk melakukan perjalanan dari titik A ke titik B. *Server* Scaling: Normcore menghosting lebih dari satu aplikasi dan mencakup penskalaan otomatis yang cerdas. Normal adalah sistem operasi *server* untuk lebih dari sekadar aplikasi, yang berarti memiliki kumpulan besar *server* yang tersedia untuk dimasuki jika aplikasi tiba-tiba dikunjungi banyak pengguna baru. Normcore juga secara konstan mengukur beban di semua *server* dan bekerja untuk memprediksi apakah diperlukan lebih banyak atau lebih sedikit.

F. *Pathfinding*

Metode *pathfinding* adalah teknik yang digunakan untuk mencari jalur terpendek antara dua titik atau lokasi tertentu di dalam sebuah lingkungan atau peta. Metode ini sering digunakan pada aplikasi dan permainan yang memerlukan navigasi antar karakter atau objek di dalam lingkungan virtual, seperti *game* RPG, *game puzzle*, dan *game* strategi. Metode *pathfinding* bekerja dengan melakukan pencarian rute terpendek dari titik awal ke titik tujuan, sambil memperhitungkan adanya rintangan atau hambatan di sepanjang jalan. Terdapat beberapa jenis metode *pathfinding* yang umum digunakan pada *game*, seperti algoritma *Dijkstra*, A*, dan algoritma *Breadth-First Search*. Dalam pengaplikasiannya pada *game*, metode *pathfinding* memungkinkan karakter atau objek untuk bergerak secara otomatis dan menghindari rintangan atau hambatan di

sekitarnya, sehingga menciptakan pengalaman bermain yang lebih realistis dan menantang[9].

G. *Algortima Dijkstra*

Algoritma *Dijkstra* adalah algoritma yang digunakan untuk mencari jalur terpendek antara dua simpul pada sebuah graf berbobot. Graf berbobot adalah graf yang memiliki bobot atau nilai pada setiap sisi atau edge-nya. Algoritma *Dijkstra* menggunakan pendekatan greedy (serakah) untuk mencari jalur terpendek dengan mengambil keputusan terbaik pada setiap langkah.Tujuan dari algoritma *Dijkstra* adalah untuk menemukan jalur terpendek dari simpul awal ke semua simpul lain dalam graf. Algoritma ini bekerja dengan menghitung jarak terpendek dari simpul awal ke setiap simpul lainnya secara bertahap. Pada awalnya, jarak ke simpul awal dianggap nol, sedangkan jarak ke semua simpul lain dianggap tak terhingga. Selama proses berlangsung, algoritma secara iteratif memperbarui jarak-jarak ini berdasarkan bobot tepi yang terhubung[10].

H. *Quality of Service*

Quality of Service adalah konsep dan teknologi yang digunakan dalam jaringan komunikasi dan sistem komputer untuk mengelola dan memprioritaskan lalu lintas data. Tujuan utama QoS adalah untuk mengatur dan meningkatkan kualitas layanan yang disediakan oleh jaringan atau sistem komunikasi. Dengan QoS, jaringan dapat memberikan kinerja yang konsisten, andal, dan memprioritaskan pengiriman data sesuai dengan kebutuhan aplikasi dan pengguna[11].

Tabel I
Indeks Parameter Qos

Nilai	Persentase (%)	Indeks
3,8 – 4	95 – 100	Sangat Bagus
3 – 3,79	75 – 94,75	Bagus
2 – 2,99	50 – 74,75	Sedang
1 – 1,99	25 – 49,75	Buruk

II. Metodologi Penelitian

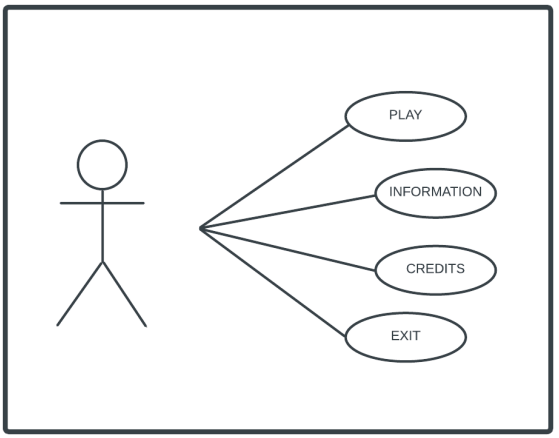
Perancangan sistem dibuat dengan tujuan untuk memberikan gambaran yang jelas mengenai sistem yang akan dirancang.

B. *Perancangan Use Case Diagram*

Diagram *Use Case* adalah salah satu jenis diagram yang digunakan dalam pemodelan perangkat lunak untuk menggambarkan cara pengguna atau aktor berinteraksi dengan sistem perangkat lunak.

1) *Use Case Diagram*

Use Case Diagram menggambarkan visualisasi pengguna saat menjalankan sistem atau aplikasi

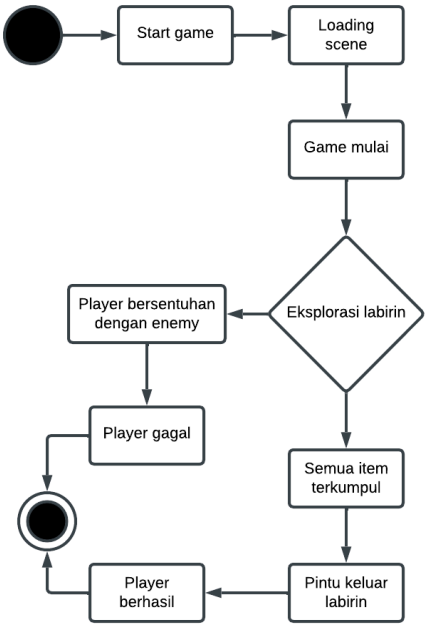


Gambar 1. Use Case Diagram

Pada gambar 2 menjelaskan tentang use case diagram yang ada dalam tampilan menu. Play untuk memulai permainan, informations untuk melihat informasi, credits untuk melihat tampilan credits dan exit untuk keluar dari aplikasi.

2) *Activity Class Diagram*

Activity class diagram menggambarkan alur dari *game* petualangan labirin.

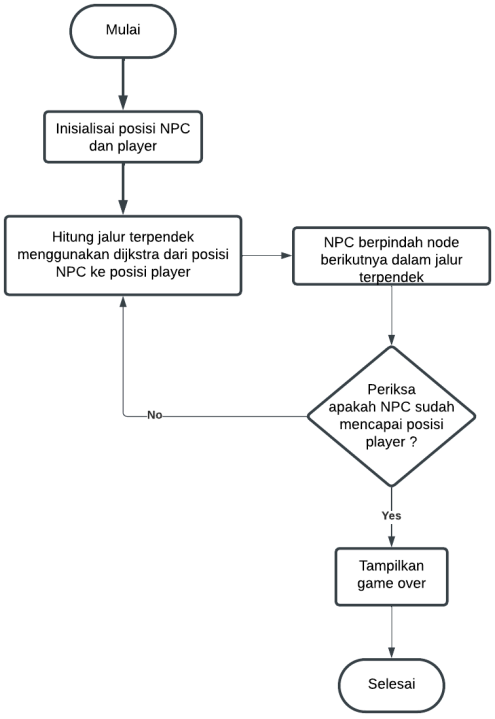


Gambar 1. Activity Class Diagram

Pada gambar 3 menjelaskan alur kerja atau aktivitas sebuah sistem *game* yang dimulai start *game* untuk memulai permainan kemudian masuk tampilan loading scene. Untuk memulai permainan *player* menjalankan karakter dan mulai mengeksplorasi labirin untuk mencari item untuk membuka pintu labirin untuk menyelesaikan permainan. Namun, apabila *player* bersentuhan dengan musuh maka *game* akan selesai dan *player* gagal.

C. *Perancangan Dijkstra*

Dalam perancangan Algoritma *Dijkstra* untuk *game* petualangan labirin algoritma ini akan digunakan untuk mengarahkan NPC dalam permainan. Tujuannya adalah agar NPC dapat mencari jalur terpendek menuju karakter pemain yang berada di dalam labirin.



Gambar 2. Flowchart *Dijkstra* pada NPC

Pada gambar 3 menjelaskan pergerakan NPC dalam permainan. Alur dimulai dengan inisialisasi posisi NPC dan *player*. NPC memeriksa apakah sudah dekat dengan *player*, jika ya, NPC diam. Jika belum, NPC menggunakan algoritma *Dijkstra* untuk mencari jalur terpendek ke *player*. Setelah jalur terhitung, NPC bergerak mengikuti jalur tersebut. Jika NPC mencapai *player*, akan menampilkan *game over* jika tidak, alur kembali ke perhitungan jalur. Alur berakhir di langkah "Selesai".

III. Hasil dan Pembahasan

A. *Desain Perangkat Lunak*

Desain yang dihasilkan untuk *game* petualangan labirin berupa menu utama, panduan, *loading*, *story*, *gameplay*, *pause game* serta tampilan *winning* dan *game over*.

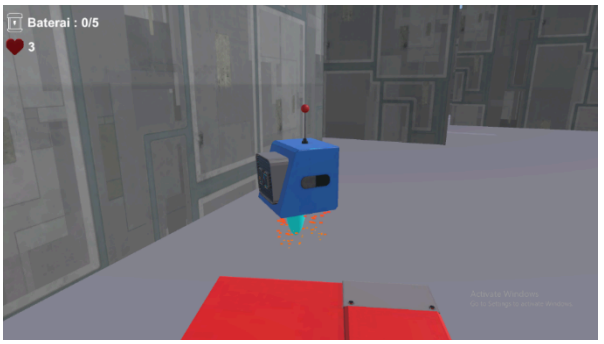
1) *Desain Tampilan Menu Utama*

Pada halaman menu utama ini menampilkan beberapa menu yang bisa diakses yaitu meliputi menu *play*, *informations*, *credits*, dan *exit*.

Gambar 5. Tampilann Halaman Utama

2) *Desain Tampilan Gameplay*

Halaman *gameplay* akan ditampilkan setelah menekan tombol *play*. Pada tampilan *gameplay* terdapat beberapa fitur meliputi, 3 nyawa, kemudian bar baterai untuk dikumpulkan.



Gambar 6. Tampilan *Game*play

B. *Implementasi Multiplayer Online*

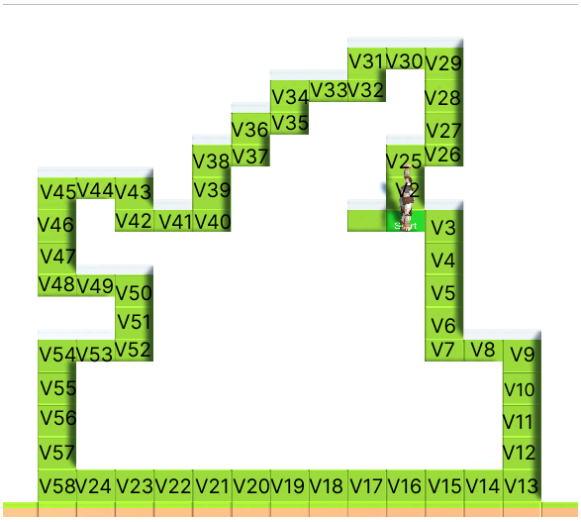
Implementasi normcore pada *server* ini akan memungkinkan pemain untuk terhubung secara *online* dan bermain bersama dengan pemain lainnya dalam permainan. Dengan adanya *server* yang kuat dan handal, pengalaman bermain *game* secara *multiplayer* menjadi lebih lancar dan menyenangkan bagi semua pemain yang terlibat.

C. *Hasil Pengujian*

Pengujian dilakukan dengan tujuan membuktikan kemampuan algoritma *Dijkstra* dalam membantu karakter NPC untuk mencari rute terpendek menuju posisi karakter pemain serta menguji kualitas jaringan pada masing-masing *player*.

1) *Pengujian Metode Dijkstra*

Pengujian dilaksanakan dengan tujuan untuk mengidentifikasi keberhasilan NPC dalam menemukan jalur terpendek dan tercepat menuju karakter pemain.



Gambar 7. Node pada Graf

Untuk membuktikan jalur yang dilalui NPC adalah jalur terpendek maka dilakukan pendekatan pengujian menggunakan algoritma *Dijkstra*.

Tabel II
Perhitungan *Dijkstra*

Vertex	Jarak
V1	0
V2	1
V3	1
V4	2
V5	3
V6	4
V7	5
V8	6
V9	7
V10	8
V11	9
V12	10
V13	11
V14	12
V15	13
V16	14
V17	15
V18	16
V19	17
V20	18
V21	19
V22	20
V23	21
V24	22
V25	Infinity
V26	Infinity
V27	Infinity
V28	Infinity
V29	Infinity
V30	Infinity
V31	Infinity
V32	Infinity
V33	Infinity
V34	Infinity
V35	Infinity
V36	Infinity
V37	Infinity
V38	Infinity
V39	Infinity
V40	Infinity
V41	Infinity
V42	Infinity
V43	Infinity
V44	Infinity
V45	Infinity
V46	Infinity
V47	Infinity
V48	Infinity
V49	Infinity
V50	Infinity
V51	Infinity
V52	Infinity
V53	Infinity
V54	Infinity
V55	Infinity
V56	Infinity
V57	Infinity
V58	23

Pada Tabel II dapat dijelaskan bahwa jarak terpendek dari V1 ke V58 adalah 23 dengan jalur V1, V3, V4, V5, V6, V7, V8, V9 ,V10, V11, V12, V13, V14, V15, V16, V17, V18, V19, V20, V21, V22, V23, V24, V58. Gambar 4. 9 menunjukkan node pada tiap graf.

Perubahan koordinat yang terjadi selama pergerakan NPC menunjukkan bahwa NPC mengikuti jalur terpendek dan tercepat menuju targetnya.

Tabel III
koordinat pergerakan npc

ID Pengujian	Koordinat Karakter pemain	Koordinat Awal NPC	Koordinat Akhir NPC	Jalur yang Dilalui	Waktu Tempuh (s)
P01	X = 35.35, Y = 0, Z = 34.73	X = -11.2, Y = 0.842, Z = -1.1	X = 35.17575, Y= 0.03132677, Z = 34.32903	Jalur pertama = 97.83865 m	53.88s
P02	X = 35.35, Y = 0, Z = 34.73	X = -11.2, Y = 0.842, Z = -1.1	X = 35.08838, Y = 0.03132677, Z = 34.22338	Jalur pertama = 97.83865 m	53.85s
P03	X = 35.35, Y = 0, Z = 34.73	X = -11.2, Y = 0.842, Z = -1.1	X = 36.22969, Y = 0.03132677, Z = 35.15655	Jalur pertama = 97.83865 m	54.32s
P04	X = 35.35, Y = 0, Z = 34.73	X = -11.2, Y = 0.842, Z = -1.1	X = 35.62729, Y = 0.03132677, Z = 34.71714	Jalur pertama = 97.83865 m	54.07s
P05	X = 35.35, Y = 0, Z = 34.73	X = -11.2, Y = 0.842, Z = -1.1	X = 35.32662, Y = 0.03132677, Z = 34.788	Jalur pertama = 97.83865 m	53.97s
P06	X = 35.35, Y = 0, Z = 34.73	X = -11.2, Y = 0.842, Z = -1.1	X = 35.93776,	Jalur pertama = 97.83865 m	54.20s

			Y = 0.03132677, Z = 34.29955		
P07	X = 35.35, Y = 0, Z = 34.73	X = -11.2, Y = 0.842, Z = -1.1	X = 35.10943, Y = 0.03132677, Z = 34.75851	Jalur pertama = 97.83865 m	53.85s
P08	X = 35.35, Y = 0, Z = 34.73	X = -11.2, Y = 0.842, Z = -1.1	X = 35.93501, Y = 0.03132677, Z = 34.62106	Jalur pertama = 97.83865 m	54.18s
P09	X = 35.35, Y = 0, Z = 34.73	X = -11.2, Y = 0.842, Z = -1.1	X = 35.95107, Y = 0.03132677, Z = 34.65586	Jalur pertama = 97.83865 m	54.21s
P10	X = 35.35, Y = 0, Z = 34.73	X = -11.2, Y = 0.842, Z = -1.1	X = 35.22352, Y = 0.03132677, Z = 34.89064	Jalur pertama = 97.83865 m	53.94s
P11	X = 35.35, Y = 0, Z = 34.73	X = -11.2, Y = 0.842, Z = -1.1	X = 35.33196, Y = 0.03132677, Z = 34.47151	Jalur pertama = 97.83865 m	53.98s
P12	X = 35.35, Y = 0, Z = 34.73	X = -11.2, Y = 0.842, Z = -1.1	X = 35.37689, Y = 0.03132677, Z = 34.67369	Jalur pertama = 97.83865 m	54.00s
P13	X = 35.35, Y = 0, Z = 34.73	X = -11.2, Y = 0.842, Z = -1.1	X = 35.36727, Y = 0.03132677, Z = 34.72451	Jalur pertama = 97.83865 m	53.99s
P14	X = 35.35, Y = 0, Z = 34.73	X = -11.2, Y = 0.842, Z = -1.1	X = 35.33585, Y = 0.03132677, Z = 34.61427	Jalur pertama = 97.83865 m	53.96s
P15	X = 35.35, Y = 0, Z = 34.73	X = -11.2, Y = 0.842, Z = -1.1	X = 35.31374, Y = 0.03132677, Z = 34.38308	Jalur pertama = 97.83865 m	53.96
P16	X = 35.35, Y = 0, Z = 34.73	X = -11.2, Y = 0.842, Z = -1.1	X = 35.3221, Y = 0.03132677, Z = 34.57553	Jalur pertama = 97.83865 m	53.96s
P17	X = 35.35, Y = 0, Z = 34.73	X = -11.2, Y = 0.842, Z = -1.1	X = 35.13293, Y = 0.03132677, Z = 34.55407	Jalur pertama = 97.83865 m	53.86s
P18	X = 35.35, Y = 0, Z = 34.73	X = -11.2, Y = 0.842, Z = -1.1	X = 35.32692, Y = 0.03132677, Z = 34.55222	Jalur pertama = 97.83865 m	53.96s
P19	X = 35.35, Y = 0, Z = 34.73	X = -11.2, Y = 0.842, Z = -1.1	X = 35.34253, Y = 0.03132677, Z = 34.7373	Jalur pertama = 97.83865 m	53.97s
P20	X = 35.35, Y = 0, Z = 34.73	X = -11.2, Y = 0.842, Z = -1.1	X = 35.33493, Y = 0.03132677, Z = 34.45841	Jalur pertama = 97.83865 m	63.59s
				Rata-rata waktu tempuh	54.48s

Pada Tabel III dapat dijelaskan bahwa algoritma *Dijkstra* secara umum berhasil dalam menemukan jalur terpendek. Terdapat variasi waktu tempuh pada tiap pemgujian, mengindikasikan adanya faktor-faktor yang memengaruhi pergerakan NPC, termasuk kondisi lingkungan, dinamika lingkungan atau mungkin kualitas jaringan.

Pada pengujian P20, NPC berhasil memilih jalur terpendek namun mengalami keterlambatan yang menyebabkan waktu tempuh lebih lama. Ini menunjukkan bahwa terdapat situasi di mana keterlambatan dapat mempengaruhi performa algoritma.

Berdasarkan hasil pengujian, algoritma *Dijkstra* berhasil dalam menemukan jalur terpendek dalam sebagian besar pengujian. Namun, variabilitas waktu tempuh menunjukkan pentingnya mempertimbangkan faktor-faktor eksternal yang dapat memengaruhi performa algoritma dalam permainan.

2) Pengujian Kualitas Jaringan

Pengujian dilaksanakan dengan tujuan untuk mengidentifikasi kualitas jaringan dalam permainan pada saat bermain *online*. Pengujian dilakukan pada masing masing pemain yang bermain pada jaringan yang berbeda. Pendekatan pengujian yang digunakan adalah dengan mengukur parameter-parameter QoS seperti *Throughput, Packet loss, Delay* dan *Jitter*.

TABEL IV
HASIL PERHITUNGAN QOS

Skenario	Parameter QoS	Rata-rata indeks	Kategori

	Throughput (bps)	***Packet Loss (%)***	***Delay* (ms)**	***Jitter* (ms)**		
Player 1	81	0,1	0,02	0,02		
Player 2	79	0,2	0,02	0,02		
Player 3	84	0,2	0,02	0,02		
Player 4	78	0,2	0,02	0,02	**3,75**	**Bagus**
Rata-rata	**80,5**	**0,175**	**0,02**	**0,02**		
Indeks	**4**	**4**	**4**	**3**		

Pada table IV dapat dilihat bahwa kualitas layanan secara keseluruhan dalam skenario permainan *multiplayer* ini cukup baik. Rata-rata throughput 80,5 bps menunjukkan bahwa *server* mampu mengirimkan data dengan baik kepada semua pemain. Tingkat packet loss yang rendah, terutama pada pemain 1, mengindikasikan bahwa hampir tidak ada paket data yang hilang dalam perjalanan, menjadikan pengalaman bermain lebih lancar. Selain itu, delay dan jitter yang rendah pada semua pemain menunjukkan bahwa koneksi internet pemain relatif stabil dan memiliki sedikit variasi dalam waktu pengiriman data. Meskipun ada sedikit peningkatan yang dapat dilakukan pada pemain 2, 3, dan 4 untuk menjaga kualitas layanan yang seragam, secara keseluruhan, skenario ini memberikan pengalaman bermain *game online* yang baik bagi semua pemain yang terlibat.

IV. SIMPULAN

Berdasarkan penelitian yang telah dilakukan oleh penulis, dapat diambil kesimpulan bahwa "Pembuatan *Game* 3D Petualangan Labirin Menggunakan Algoritma *Dijkstra* Pada Non-*Player* Character (NPC)" berhasil dilakukan dengan kesimpulan sebagai berikut:

1. Dari 20 pengujian yang dilakukan, semua berhasil mencapai jalur terpendek melalui node V1, V3, V4, V5, V6, V7, V8, V9 ,V10, V11, V12, V13, V14, V15, V16, V17, V18, V19, V20, V21, V22, V23, V24, V58, meskipun pada pengujian P20 terdapat keterlambatan. Dengan demikian, tingkat keberhasilan algoritma *Dijkstra* dalam memilih jalur terpendek adalah 100%.
2. Implementasi mode *multiplayer online* pada *game* ini telah berhasil dilakukan. Pemain berhasil untuk bermain bersama secara *online*.
3. Dari hasil pengujian QoS yang dilakukan oleh empat pemain yang bermain secara bersamaan, diketahui bahwa kualitas jaringan dari masing-masing pemain berada dalam kategori (Bagus) menurut standar Tiphon.

REFERENSI

[1] D. Sumarmo and V. Lusiana, "Implementasi Algoritma *Dijkstra* Pada *Game* Pengenalan Kebudayaan Kota Semarang," no. Mdlc, pp. 978–979, 2020.

[2] B. T. D. Irianto, S. Andryana, and A. Gunaryati, "Penerapan Algoritma A-Star Dalam Mencari Jalur Tercepat dan Pergerakan Non*Player* Character Pada *Game* Petualangan Labirin Tech-Edu," *J. Media Inform. Budidarma*, vol. 5, no. 3, p. 953, 2021, doi: 10.30865/mib.v5i3.3094.

[3] E. Junanto, A. B. Osmond, A. Siswo, and R. Ansori, "Membuat Pergerakan Non-*Player* Character (Npc) Menggunakan Metode a Star Making Non-*Player* Character (Npc) Movement Using the a Star Method," vol. 7, no. 1, p. 1491, 2020.

[4] W. Wibawanto, "Untuk Gerakan Kendaraan NPC Dalam *Game* Pendahuluan Permainan elektronik atau *game* telah menjadi bagian integral kehidupan," vol. 3, no. 1, pp. 15–32, 2017.

[5] I. R. C. Harits Ar Rosyid, Syaad Patmanthara, *GAME DEVELOPMENT*. Ahlimedia Book, 2021.

[6] F. D. Luna, *Introduction to 3D Game Programming with DirectX 10*. Wordware Pub., 2008.

[7] Chairunisa, "Mengenal *Game Online*: Pengertian, Industri, Sejarah hingga Jenisnya," *dailysocial.id*, 2022. https://dailysocial.id/post/mengenal-*game*-*online*-pengertian-industri-sejarah-hingga-jenisnya (accessed May 17, 2023).

[8] S. K. Thorsten Quandt, Ed., *Multiplayer: The Social Aspects of Digital Gaming*, Berilustra. Routledge, 2013.

[9] S. Rabin, Ed., *AI Game Programming Wisdom 3*. Charles River Media, 2006.

[10] C. S. Thomas H Cormen, Charles E Leiserson, Ronald L Rivest, *Introduction To Algorithms*. MIT Press, 2001.

[11] A. A. Sukmandhani, "QoS (Quality of Services)," *onlinelearning.binus.ac.id*, 2020. https://*online*learning.binus.ac.id/computer-science/post/qos-quality-of-services