Jurnal Ilmiah Sinus (JIS) Vol : 17, No. 2, Juli 2019
ISSN (Print) : 1693-1173 , ISSN (Online): 2548-4028

Penerapan *Pathfinding* Menggunakan Algoritma A* Pada *Non Player Character* (NPC) Di *Game*

Aditya Setiyawan[1)], Paulus Harsadi [2)], Sri Siswanti [3)]
[1,2,3)]Program Studi Teknik Informatika, STMIK Sinar Nusantara
[1)]adityasetyawan38@gmail.com; [2)]paulusharsadi@sinus.ac.id; [3)]syswanty@sinus.ac.id

ABSTRACT

*In this Game , we implement a character designer at the enemy by applying the algorithm A * pathfinding Algorithm. Furthermore, A * is as the algorithm for searching solutions with the use of additional information (heuristics) in which it is an optimal solution. The purpose of developing this Adventure Game is A * Algorithm implementation for Pathfinding using Unity 3D. Pathfinding is the fastest path of searching process from point of origin to point of destination by avoiding the various barriers along the path traveled without crashing the existing barrier along the way. Its design and development applies Unity 3D. The test result of algorithm in the Game shows that the total of passed node result is 50 nodes. It is similar with the previous manual count result. Furthermore, the Game route shows that its passed path is similar with manual count.*
keywords : Game , pathfinding, Algoritma A, non palyer character*

I. PENDAHULUAN

Artificial Intelligence (AI) di *game* diharapkan mampu membuat karakter baik player maupun *NPC (Non Player Character)* memiliki perilaku manusia yang sesungguhnya baik bergerak maupun berfikir (Millington & Funge, 2009). *Pathfinding* merupakan salah satu masalah di AI yang paling banyak digunakan terutama di *game advanture* seperti yang akan dibuat. *Pathfinding* merupakan salah satu teknik dalam *game* yang paling banyak diteliti karena *pathfinding* merupakan teknik utama atau dasar dalam pembuatan *game* (Adi Botea, et al., 2013).

Pathfinding sendiri dalam *game* diterapkan dalam bentuk variasi map yang berbeda dan pendekatan solusi yang berbeda-beda seperti yang pernah di paparkan oleh Ross Graham dalam makalahnya yang berjudul *"pathfinding in Computer Game s"* (Graham, McCabe, & Sheridan, 2003).

Penerapan *pathfinding* meliputi analisa sebuah peta untuk menemukan nilai terbaik dalam perjalanan dari satu titik ke titik yang lain. Lintasan terbaik disini dapat diartikan tidak hanya sekedar lintasan terpendek, tetapi bisa diartikan nilai lintasan paling sedikit, atau lintasan yang aman (Yap, 2002). Tetapi, penerapan *pathfinding* dalam pembuatan *game* ini lebih focus digunakan untuk pencarian rute terpendek.

Penelitian *pathfinding* dalam *game* sudah cukup banyak dilakukan (Adi Botea, Bruno Bouzy, Michael Buro, Christian Bauckhage, 2013). Algoritma A*, atau *A* pathfinding* dan variasi perkembangan dari *A* pathfinding* sudah umum digunakan dalam AI *Game* dan merupakan metode yang paling banyak digunakan di *game* (Cowling et al., 2014)(Cui & Shi, 2011). Algoritma ini juga dapat digunakan dalam berbagai bentuk *grid* sebagai bentuk map pada *game* dan dapat diterapkan dalam berbagai bentuk map dalam *strategy game s* (Barnouti, Al-Dabbagh, & Sahib Naser, 2016).

Algoritma A* sendiri pertama kali dikembangkan oleh P. Hart, N. Nilson dan B. Raphael pada tahun 1968 (Hart, Nilson, & Raphael, 1968).

Pada *game* yang dibuat akan terdapat tokoh *enemy* yang akan menjadi *Non Player Character (NPC)* dengan menggunakan area hutan yang luas. Dimana pada nantinya NPC

dirancang agar dapat mencari target utama / pemain dengan rute terpendek dan dapat menghindari penghalang yang ada di area *game* .

.

II. TINJAUAN PUSTAKA

2.1. Algoritma A*

Algoritma A* menyelesaikan masalah yang menggunakan graph untuk perluasan ruang statusnya. Dengan kata lain digunakan untuk menyelesaikan permasalahan yang bisa direpresentasikan dengan graph. Algoritma A* adalah sebuah algoritma yang telah dikembangkan.

Dengan menerapkan fungsi heuristik, algoritma ini membuang langkah-langkah yang tidak perlu dengan pertimbangan bahwa langkah-langkah yang dibuang sudah pasti merupakan langkah yang tidak akan mencapai solusi yang diinginkan.

Didalam algoritma A* menggunakan dua senarai yaitu *OPEN* dan *CLOSED*. Pada algoritma A* memiliki tiga kondisi dimana NPC berada di keadaan *OPEN* , berada dikeadaan *CLOSED*, dan tidak berada dikeduanya. Jika NPC dalam keadaan *OPEN* maka akan dilakukan pengecekan apakah perlu pengubahan parent atau tidak tergantung pada nilai g melalui parent lama maupun parent baru.

Jika NPC tidak berada dikeadaan *OPEN* maupun *CLOSED*, maka NPC tersebut akan dimasukan di dalam keadaan *OPEN* . Kemudian hitung biaya NPC tersebut dengan rumus $f = g + h$. *Algoritma A** ini akan memberikan solusi terbaik yang optimal didalam arena.

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

Keterangan :

f(n)	=	solusi biaya estimasi termurah verteks n untuk mencapai tujuan.
g(n)	=	biaya *path* / perjalanan.
h(n)	=	biaya estimasi dari verteks n ke tujuan.

Suatu fungsi *heuristik* dapat dikatakan baik, apabila dapat memberikan biaya perkiraan yang mendekati biaya sebenarnya. Semakin mendekati biaya sebenarnya, fungsi *heuristik* tersebut semakin baik. Dalam masalah pencarian rute terpendek dengan *graph*, fungsi heuristik yang dapat digunakan adalah *Manhattan Distance*.

2.2. Manhattan Distance

Manhattan Distance adalah salah satu fungsi heuristik yang paling sering digunakan dalam menghitung jarak terdekat dari dua titik. Fungsi ini hanya menghitung selisih koordinat posisi titik awal dan titik akhirnya saja. Awalnya model ini digunakan di kota Manhattan, Amerika, karena di kota ini jarak dari dua lokasi dihitung dari blok- blok jalan yang harus dilalui saja, sehingga tidak bisa dilewati secara diagonal (Niedermeier & Sanders, 1996). Fungsi heuristik ini dapat dihitung berdasarkan selisih jarak koordinat yang diambil dari dua buah titik, yang dapat diformulasikan menggunakan rumus (2).

$$h(n) = abc(n.x - tujuan.x) + abs(n.z - tujuan.z) \quad (2)$$

Keterangan :

$n.x$	: Koordinat x titik start
$n.z$	: Koordinat z titik start
$tujuan.x$	: Koordinat x titik tujuan
$tujuan.z$	: Koordinat z titik tujuan

 DOI : https://doi.org/10.30646/sinus.v17i2.423

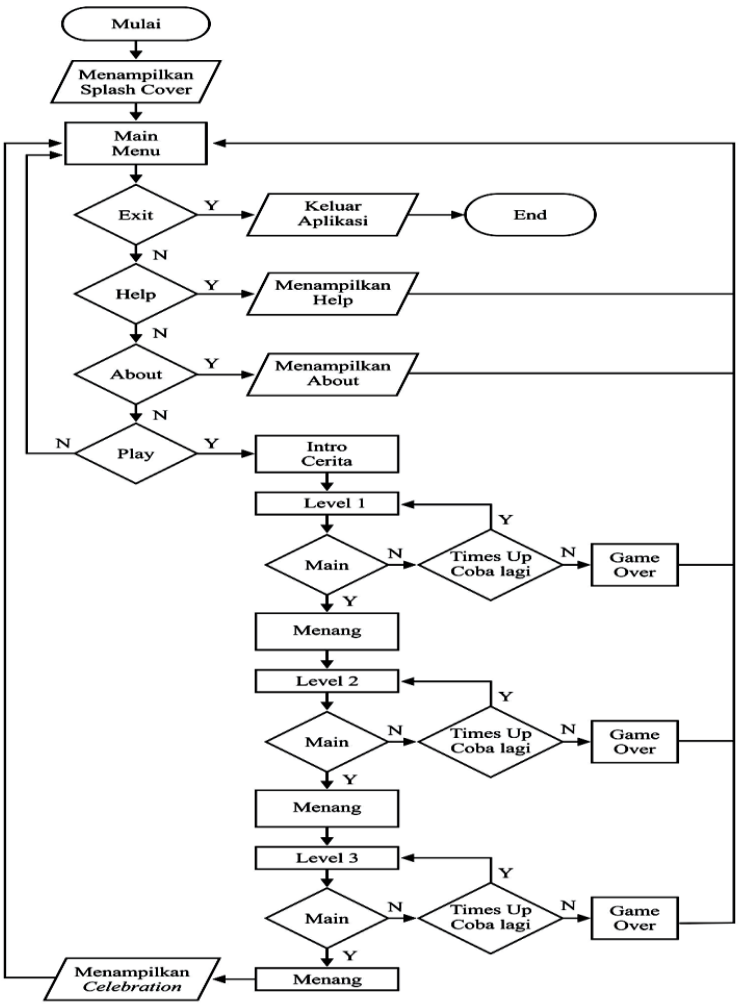
Fungsi dari *abs* diatas untuk mengubah nilai negatif menjadi nilai positif. ini terjadi apabila titik awal berada lebih jauh dari titik akhir (*reverse*), sehingga pengurangan x2-x1 mempunyai hasil yang negatif.

Game yang dibuat menggunakan *game* engine yaitu Unity 3D. *Unity 3D* adalah sebuah *game engine* yang berbasis *cross-platform*. *Unity* dapat digunakan untuk membuat sebuah *game* yang bisa digunakan pada perangkat komputer, ponsel pintar *android, iPhone, PS3, dan bahkan X-BOX*. Bahasa pemrograman yang dapat diterima *Unity* adalah *JAVA SCRIPT*, CS SCRIPT (C#) & BOO SCRIPT. *Unity 3D* adalah salah satu *software* yang bagus untuk mengembangkan *game 3D* dan selain itu juga merupakan *software* atau aplikasi yang interaktif dan atau dapat juga digunakan untuk membuat animasi 3 dimensi (Blackman, 2013).

III. METODE PENELITIAN

3.1. *Flowchart* Permainan

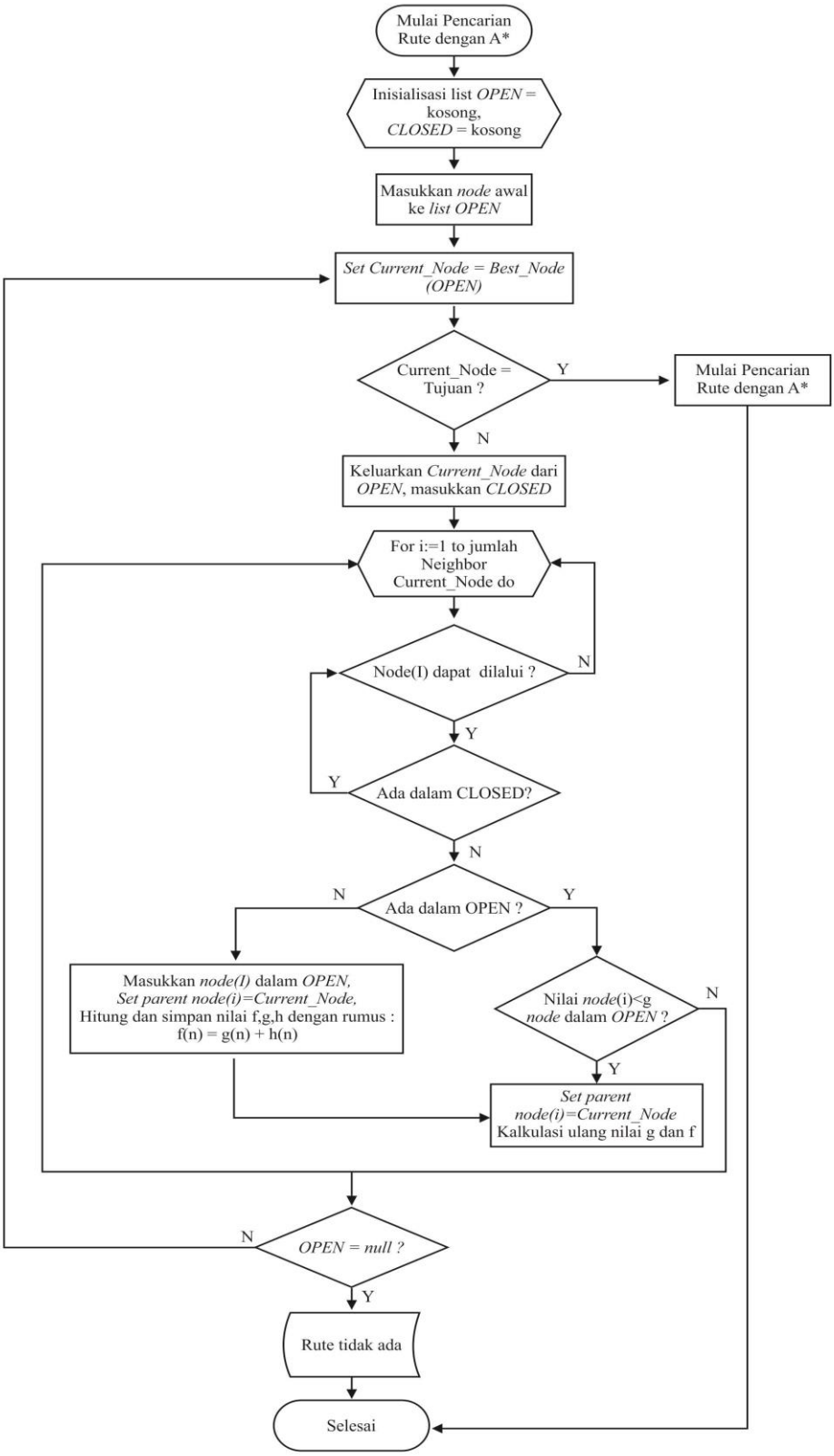
Game yang akan dibangun masuk dalam kategori *game adventure* yaitu berbasis pencarian atau *pathfinding*. *Advanture game* sendiri merupakan *game* yang membuat *player* berjalan dalam map menyelesaiakan masalah berdasarkan *story line* yang dibuat (Fairclough, et al., 2001). *Flowchart* desain level dalam *game* ini adalah seperti pada Gambar 1.



Gambar 1. *Flowchart Game*

3.2. *Flowchart* Algoritma A*

Flowchart Algoritma A* dalam *game* ini adalah seperti pada Gambar 2



Gambar 2. *Flowchart* Algoritma A*

 DOI : https://doi.org/10.30646/sinus.v17i2.423

Algoritma A* akan diterapkan dalam proses pencarian atau *pathfinding* dalam *game* yang dibuat.

IV. HASIL DAN PEMBAHASAN

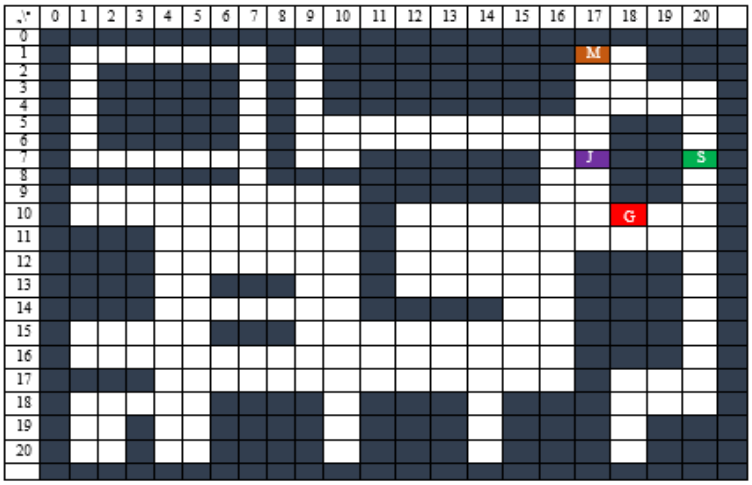
4.1. Deskripsi Karakter

Terdapat beberapa tokoh karakter yang dibangun pada *game* ini, yaitu :

- Joko Kendhil (Karakter Player)
  Karakter utama berperan sebagai Joko Kendhil, karakter ini yang memerankan sebagai first person dalam pencarian misi mendapatkan gong lokananta. Dengan menggunakan first person atau orang pertama mengacu pada perspektif grafis yang diberikan dari sudut pandang karakter pemain.
- NPC (*Non Player Character*)
  Terdapat dua peranan sebagai NPC yakni yang pertama sebagai pesaing Joko Kendhil (Divisualkan sebagai sesama manusia) yang apabila mengenai objek NPC (manusia) tersebut tidak akan *game over*, dan yang kedua sebagai *enemy* (Divisualkan sebagai wujud hewan / burung) yang dimana kalau tertangkap NPC (burung) tersebut maka akan *game over*.

4.2. Pembuatan Arena

Arena dibuat dengan menggunakan arena sesungguhnya pada *game* , seperti Gambar 3.



Gambar 3. Arena *Game*

Pada Gambar 3 adalah papan *grid* yang akan dirancang sebagai arena *game* yang telah dibuat menggunakan *grid* arena yang sebenarnya dengan menggunakan skala 1 : 10 pada penerapannya.Warna hijau pada papan *grid* merupakan starting point Pesaing Joko Kendhil / NPC sebelum bergerak. Warna merah pada papan *grid* merupakan *end point* / Gong Lokananta yang akan dituju oleh Pesaing Joko Kendhil. Warna Ungu adalah sebagai starting point Joko Kendhil, Warna Orange adalah sebagai musuh / NPC yang mengejar Joko Kendhil, Warna biru dongker pada *grid* merupakan pepohonan yang tidak dapat dilewati dan warna putih adalah *grid* yang bisa dilewati. Nilai heuristik dihitung menggunakan fungsi manhattan1 yang nantinya akan digunakan dalam perhitungan A* pada Pesaing dan Musuh Joko Kendhil.

Tabel 1 merupakan nilai heuristik dihitung menggunakan fungsi manhattan1 yang nantinya akan digunakan dalam perhitungan A* pada Pesaing dan Musuh Joko Kendhil.

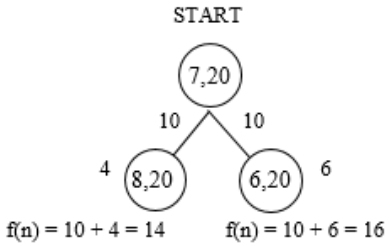
Tabel 1. Nilai Heuristik (Pesaing Joko Kendhil)

Start_X	Start_Z	Goal_X	goal_Z	h(n)=abs(Start_X-Goal_X)+abs(Start_Z-Goal_Z)	Initial *Node*
1	1	10	18	26	
1	2	10	18	25	
1	3	10	18	24	
1	4	10	18	23	
1	5	10	18	22	
1	6	10	18	21	
1	7	10	18	20	
1	9	10	18	18	
1	17	10	18	10	
1	18	10	18	9	
2	1	10	18	25	
2	7	10	18	19	
2	9	10	18	17	
2	17	10	18	9	
2	18	10	18	8	
3	1	10	18	24	
3	7	10	18	18	
3	9	10	18	16	
3	17	10	18	8	
3	18	10	18	7	
3	19	10	18	8	
3	20	10	18	9	
4	1	10	18	23	
4	7	10	18	17	
7	9	10	18	12	
7	10	10	18	11	
7	16	10	18	5	
7	17	10	18	4	
7	20	10	18	5	START
8	7	10	18	13	
8	16	10	18	4	
10	4	10	18	14	
10	5	10	18	13	
10	6	10	18	12	
10	7	10	18	11	
10	8	10	18	10	
10	9	10	18	9	
10	10	10	18	8	
10	12	10	18	6	
10	13	10	18	5	
10	14	10	18	4	
10	15	10	18	3	
10	16	10	18	2	
10	17	10	18	1	
10	18	10	18	0	GOAL
10	19	10	18	1	
10	20	10	18	2	
11	4	10	18	15	
11	5	10	18	14	
11	6	10	18	13	
11	7	10	18	12	
11	8	10	18	11	
11	9	10	18	10	
11	10	10	18	9	
11	12	10	18	7	
11	13	10	18	6	

 DOI : https://doi.org/10.30646/sinus.v17i2.423

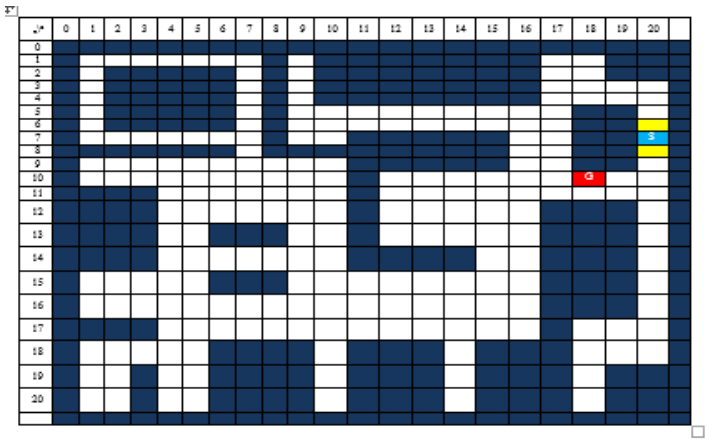
Kemudian dengan Gambar 4 pada biaya perkiraan dan juga nilai *cost* untuk bergerak atau berpindah *grid* dengan perkiraan biaya sebesar 10, maka akan dilakukan penghitungan sesuai fungsi *f(n) = g(n) + h(n)* pada *node* yang terhubung langsung dengan *node* awal saat ini. Berikut langkah pencarian pada algoritma A* *Pathfinding* :

- Langkah 1



Gambar 4. Perhitungan langkah 1

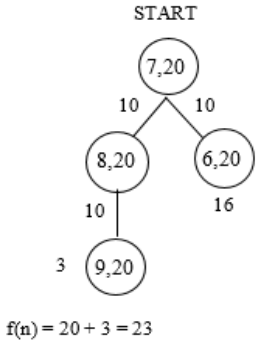
Langkah 1 seperti pada Gambar 4, didapat nilai pada simpul pertama yaitu f(n) = 10 + 4 = 14 dan f(n) = 10 + 6 = 16, maka akan di cabangkan pada simpul kedua terlebih dahulu. Simpan *node* [7,20] didalam *CLOSED list*, sedangkan *node* [8,20] dan [6,20] didalam *OPEN list.* Langkah pergerakan papan *grid* seperti pada Gambar 5.



Gambar 5. Langkah pergerakan papan *grid*

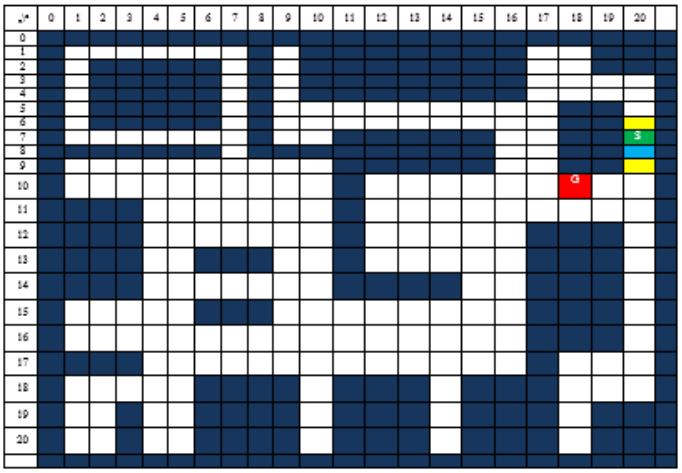
Dari perhitungan awal diatas, maka pergerakan didalam papan *grid* akan ditandai dengan kotak warna kuning sebagai *OPEN List* dan kotak warna biru nantinya merupakan *CLOSED List*.

- Langkah 2



Gambar 6. Perhitungan langkah 2

Pada Langkah 2 seperti pada Gambar 6, Simpan *node* [7,20] [8,20] kedalam *CLOSED list* serta *node* [9.20] dan [6,20] ke dalam *OPEN list*. Maka pergerakan dalam papan *grid* seperti Gambar 7.

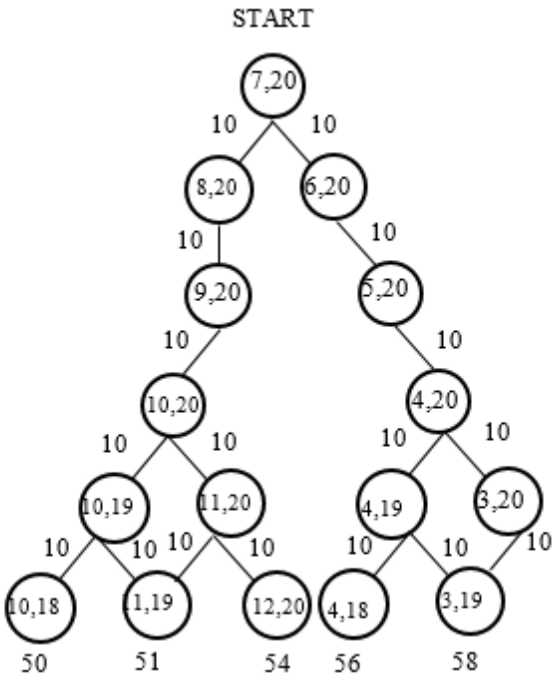


Gambar 7. Langkah 2 pergerakan papan *grid*

Ulangi langkah seperti sebelumnya dengan mencari nilai f yang paling rendah kemudian hitung menggunakan rumus A*.

Kemudian ulangi langkah tadi hingga menemukan tujuan dengan nilai simpul paling rendah (*best node).*

- Langkah 12

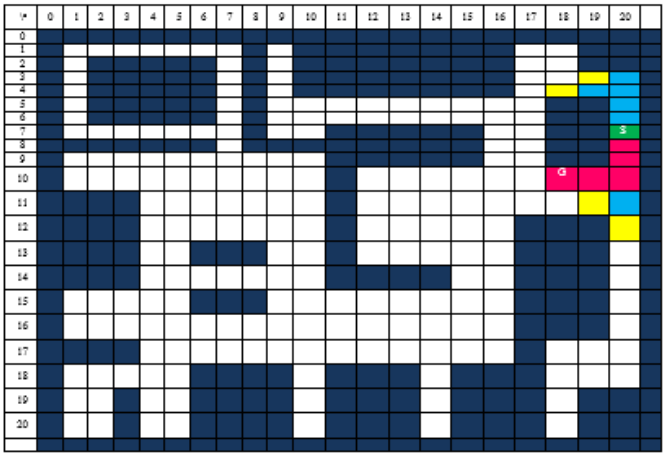


Gambar 8. Perhitungan Langkah 12

Pada langkah ke 12 seperti pada Gambar 8 didapatkan hasil pencarian Algoritma A* pathfiding dengan hasil jalur terpendeknya dimulai dari *node* [7,20] [8,20] [9,20] [10,20]

 DOI : https://doi.org/10.30646/sinus.v17i2.423

[10,19] [10,18] dengan jumlah biaya 50 yang merupakan biaya paling kecil. Pergerakan terakhir pada papan *Grid* seperti pada Gambar 9.



Gambar 9. Pergerakan terakhir pada papan *grid*

4.3. Implementasi

Pada tahap ini dilakukan pembuatan desain *game*. Berikut adalah implemetasi *game* pada bagian utama.

a. Pembangunan *Terrain* Hutan

Terrain merupakan bentuk map dari *game* secara keseluruhan disetiap levelnya. Peta *game* yang dibuat berbasis *grid* yang akan digunakan untuk *pathfinding* oleh *Player* maupun *Non Player Character*. Penegmbangan *Terrain* Hutan seperti pada Gambar 10.

Gambar 10. *Terrain* Hutan

b. Tampilan *Game*

Gambar 11 merupakan bentuk tampilan *game* ketika *Player* melakukan eksplorasi pada peta.

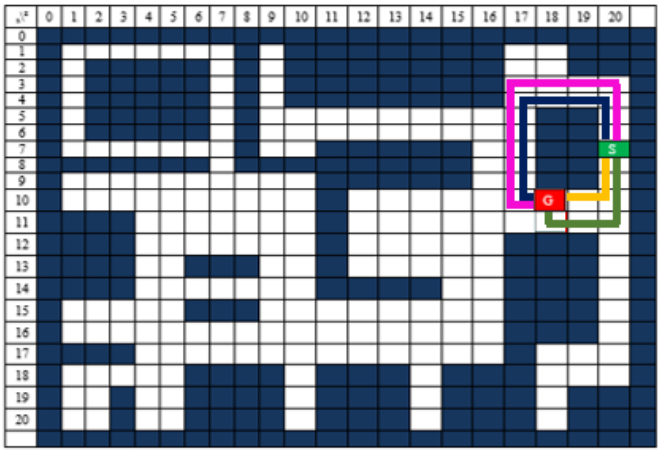
Gambar 11. Tampilan *Game*

4.4. Pengujian

Pengujian validitas lebih dilakukan uji mengenai perhitungan algoritma itu sendiri yaitu algoritma A*. Hal yang diuji adalah perbandingan hasil perhitungan manual algoritma dengan perbandingan aplikasi yang berjalan. Pada perhitungan manual perbandingan hasilnya dapat dilihat pada Tabel 2 dan Gambar 12.

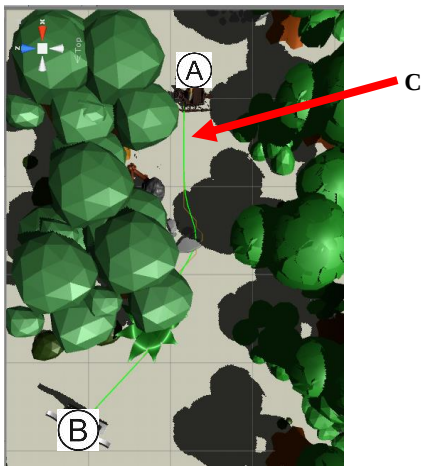
Tabel 2. Perhitungan Manual

Node Start	*Node* Tujuan	Kemungkinan Rute Manual (Terdekat)	Besar biaya (satuan kotak)
[7,20]	[10,18]	[8,20] [9,20] [10,20] [10,19] [10,18]	Panjang lintasan = 5
		[8,20] [9,20] [10,20] [11,20] [11,19] [11,18] [10,18]	Panjang lintasan = 7
		[6,20] [5,20] [4,20] [4,19] [4,18] [4,17] [5,17] [6,17] [7,17] [8,17] [9,17] [10,17] [10,18]	Panjang lintasan = 13
		[6,20] [5,20] [4,20] [3,20] [3,19] [3,18] [3,17] [4,17] [5,17] [6,17] [7,17] [8,17] [9,17] [10,17] [10,18]	Panjang lintasan = 15



Gambar 12. pergerakan papan *grid* manual

Dari hasil perhitungan secara manual pada Tabel 2 yang telah dilakukan maka akan didapat 4 kemungkinan rute terdekat A menuju B dari arena sebenarnya.



Gambar 13. Rute terbaik NPC pada *Game*

 DOI : https://doi.org/10.30646/sinus.v17i2.423

Titik A merupakan titik awal *pathfinding*, dan titik B merupakan titik tujuan sedang C yang berwarna hijau merupakan *node* optimal hasil perhitungan algoritma A* dalam *game* dan merupakan solusi tercepat jarak yang bisa ditempuh.

Solusi optimal pada *console game* pada Gambar 13 yaitu sebanyak 50 *node*, sesuai seperti perhitungan manual sebelumnya.

Tabel 3. Tabel Pengujian Program

Console	Node Dilewati	Duplikat Data	Rute A*
	50 node	Tidakada	Sesuai

Serta akan didapat solusi optimal pada *console game* seperti diatas yaitu sebanyak 50 *node*, sesuai seperti perhitungan manual sebelumnya.

V. KESIMPULAN DAN SARAN

5.1 Kesimpulan

Algoritma A* berhasil diterapkan sebagai pembangkit perilaku pencarian pada NPC pada *game* yang dibuat. Hasil pengujian algoritma yang didapat adalah hasil *node* yang dilewati yakni sebanyak 50 *node*, hal tersebut sama dengan hasil perhitungan manual algoritma A*.

Rute *player* dalam melakukan eksplorasi dalam peta *game* menunjukkan rute jalur yang sama dengan perhitungan manual menggunakan algoritma A* dan merupakan rute tercepat yang bisa dilalui.

5.2 Saran

Penggunaan algoritma A* pada *game* yang dibuat masih berbasis *grid*. Implementasi kedepan diharapkan algoritma A* bisa diimplementasikan pada area yang berbasis navigation mesh sehingga komputasi algoritma A* lebih cepat dalam menemukan titik tujuan.

DAFTAR PUSTAKA

Adi Botea, Bruno Bouzy, Michael Buro, Christian Bauckhage, D. N. (2013). *Pathfinding* in *Game* s. https://doi.org/10.4230/DFU.Vol6.12191.21

Barnouti, N. H., Al-Dabbagh, S. S. M., & Sahib Naser, M. A. (2016). *Pathfinding* in Strategy *Game* s and Maze Solving Using A* Search Algorithm. *Journal of Computer and Communications*, *04*(11), 15–25. https://doi.org/10.4236/jcc.2016.411002

Blackman, S. (2013). Beginning 3D *Game* Development with Unity 4: *Beginning 3D Game Development with Unity 4:* https://doi.org/10.1007/978-1-4302-4900-9

Cowling, P., Buro, M., Bida, M., Botea, A., Bouzy, B., Butz, M., … Sipper, M. (2014). Search in Real-Time Video *Game* s. *Artificial and Computational Intelligence in Game s*, *6*, 1–19. https://doi.org/http://dx.doi.org/10.4230/DFU.Vol6.12191.1

Cui, X., & Shi, H. (2011). A*-based *Pathfinding* in Modern Computer *Game* s. *International Journal of Computer Science and Network Security*, *11*(1), 125–130. Retrieved from http://paper.ijcsns.org/07_book/201101/20110119.pdf

Fairclough, C., Fagan, M., Mac Namee, B., & Cunningham, P. (2001). Research Directions for AI in Computer *Game* s. *Irish Conference on Artificial Intelligence and Cognitive Science*, 333–344. Retrieved from http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.17.2579&rep=rep1&type=pdf

Graham, R., McCabe, H., & Sheridan, S. (2003). *Pathfinding* in computer *Game* s. *ITB Journal*, *4*(2), 1–26. https://doi.org/10.4230/DFU.Vol6.12191.21

Hart, P., Nilson, N., & Raphael, B. (1968). A formal basis for the heuristic determination of minimum cost paths. IEEE Trans. Syst. Sci. Cybern. 4.

Millington, I., & Funge, J. (2009). *Artificial Intelligence for Game s, Second Edition*. *Representations*. https://doi.org/10.1017/S0263574700004070

Niedermeier, R., & Sanders, P. (1996). On the Manhattan-Distance Between Points on Space-Filling Mesh-Indexings. *Univ. Fak. Fur Informatik*, 1–10.

Yap, P. (2002). *Grid*-based path-finding. *Advances in Artificial Intelligence*, 44–55.