IMPLEMENTASI ALGORITMA *MULTI AGENT SYSTEM PATHFINDING* MENGGUNAKAN *LIFELONG PLANNING* A* PADA NPC PERMAINAN LABIRIN

Danang Setyo Widodo[1],Galih Hermawan[2]

[1,2]Universitas Komputer Indonesia
Jl. Dipati Ukur No. 102-114 Bandung
E-mail : danang.wiodo65@gmail.com[1],galih.hermawan@email.unikom.ac.id[2]

ABSTRAK

Labirin adalah sebuah puzzle dalam bentuk percabangan jalan yang kompleks dan memliki banyak jalan buntu. Tujuan permainan ini adalah pemain harus menemukan jalan keluar dari sebuah pintu masuk ke satu atau lebih pintu keluar. *Pathfinfing* adalah kecerdasan buatan yang menggunakan algortitma pencarian jalur tercepat. Penerapan yang dapat dilakukan dengan *pathfinding* antara lain adalah pencarian jalur dalam suatu *game* dan pencarian jalan pada suatu peta. Salah satu algoritma yang dapat dipakai adalah *Lifelong Planning* A*, merupakan versi incremental dari A* yang dapat beradaptasi dengan perubahan dalam grafik tanpa menghitung ulang seluruh grafik, dengan memperbarui nilai- g (jarak dari awal) dari pencarian sebelumnya selama pencarian saat ini untuk memperbaikinya saat diperlukan. Algoritma *Lifelong Planning* A* bertujuan untuk mengetahui akurasi tingkat keberhasilan banyak agen dalam algoritma *pathfinding* dalam menentukan langkah *Agent* labirin. Karakteristik *game* yang akan dibangun memiliki fitur penggunaan halangan untuk menghalangi *Agent* labirin dalam mengejar pemain, dan beberapa *Agent* labirin yang bergerak secara bersamaan (*multi Agent*).. Pada proses pengujian dari 30 kali pengulangan yang telah disiapkan *Agent* npc mampu menemukan jalur sebanyak 25 pada 8 *Agent* npc dan memiliki rata rata akurasi sebesar 91,7 persen. Dan memiliki nilai kecepatan sebesar 166.999 ms dalam lamanya waktu eksekusi dengan ordo 15x15 dan 8 *Agent.* Setelah melakukan pengujian sistem dengan metode *Black Box* dan dihitung tingkat akurasi dapat ditarik kesimpulan bahwa program berfungsi dengan benar.

Kata Kunci: *Lifelong Planning a*, multi Agent*, pencarian jalur, labirin, permainan.

1. PENDAHULUAN

Permainan labirin adalah permainan mencari jalur dimana jalannya labirin ini banyak mendapat beberapa halangan yang sudah ditentukan untuk mencapai pada tujuan. Pada saat ini masih banyak player yang bingung dalam mencari jalan keluar, bagaimana cara mendapat bonus nilai dan dapat menaikkan ke langkah berikutnya berikutnya. Maka dalam mencapai tujuan diperlukan sebuah solusi,salah satu solusi yang dapat dipakai adalah algoritma *backtracking*. Algoritma *Backtracking* merupakan sebuah algoritma yang dapat digunakan untuk membuat aplikasi *game* labirin dimana cara kerja algoritma ini adalah mencari jalan keluar yang tepat untuk menentukan jalur yang tepat untuk mencapai tujuan yang telah ditetapkan[2]. Namun dalam implementasinya algoritma ini hanya menggunakan *Single Agent System* saja. Sehingga tidak diketahui hasilnya apabila terdapat beberapa *Agent* yang diuji.

Terdapat beberapa penelitian tentang penerapan *AI* pada *Pathfinding* NPC pada permainan labirin seperti yang dilakukan oleh Ilham Ramadhan tahun 2017 dikatakan Algoritma *Jump Point Search* dapat diterapkan untuk mengoptimasi pencarian jalur tercepat *NPC* secara dinamis [1]. Permasalahan dari penelitian sebelumnya adalah belum adanya pengujian terhadap 2 *NPC* atau lebih.

Yang kedua adalan penelitian yang dilakukan oleh Sri Anggraini Surianto tentang Implementasi Algoritma *Iterative Deepening* A* (IDA*) Dengan *Stochastic Node Caching* (SNC) Untuk *Pathfinding* Musuh Pada *Game* Labirin, Posisi pemain, musuh dan jumlah tembok/penghalang juga mempengaruhi jumlah ekspansi simpul. Semakin jauh jarak antara pemain dan musuh, maka akan semakin banyak simpul yang diekspansi. Semakin banyak jumlah tembok/penghalang dalam map, maka akan semakin banyak pula simpul yang diekspansi. Sehingga mengakibatkan penurunan kecepatan proses apabila dilakukan terhadap 2 *Agent* atau lebih[11].

Pathfinding adalah salah satu masalah paling dasar dari kecerdasan buatan (*Artificial Intelligence* / AI) yang ada pada *game*. *Pathfinding* yang buruk dapat membuat karakter yang ada pada *game* terlihat brainless (bodoh). Penanganan masalah *pathfinding* secara efektif dapat membuat *game* lebih menyenangkan dan memberikan pengalaman bermain yang mendalam bagi player [3].

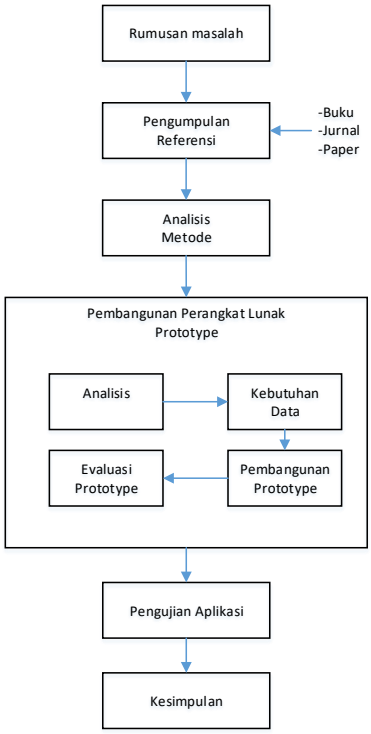
Salah satu algoritma *pathfinding* yang dapat dipakai dalam mencari dan mengenali jalur ini adalah algoritma *Lifelong Planning* A* (LPA*). Algoritma LPA* merupakan pengembangan dari algoritma A* yang dikembangkan oleh Hang Ma dan Jiaoyang Li pada tahun 2017. Algoritma ini menggunakan ratusan *Agent* dan penugasan [2].

1.1 Maksud dan Tujuan

Berdasarkan masalah yang ada, maka maksud penelitian ini adalah untuk implementasi algoritma *Lifelong Planning* A* pada NPC labirin secara bersamaan dalam pencarian jalur tercepat. Kemudian tujuan yang akan diraih dari penelitian ini adalah untuk mengetahui akurasi tingkat keberhasilan banyak agen dalam algoritma *pathfinding* dalam menentukan langkah NPC labirin dengan menggunakan *Lifelong Planning* A*.

1.2 Metode Penelitian

Metode penelitian adalah cara untuk mencapai suatu tujuan di dalam sebuah penelitian. Untuk penelitian ini penulis menggunakan metode penelitian deskriptif. Metode penelitian deskriptif adalah sebuah penelitian yang bertujuan untuk menjabarkan suatu fenomena yang terjadi saat ini dengan menggunakan prosedur ilmiah untuk menjawab masalah secara aktual.



Gambar 1. Metode Penelitian

Dalam alur metode penelitian yang dilakukan ini adalah tahap-tahap yang akan dikerjakan :

1. Pengumpulan referensi

Pada tahapan ini mengumpulkna jurnal-jurnal dan paper yang terkait dengan penelitian ini serta mencari buku untuk mendalami materi mengenai algoritma *pathfinding*, algoritma A*, dan algoritma *Lifelong Planning* A *.

2. Analisis Metode

selanjutnya melakukan analisis. Menganalisis metode Algoritma *Lifelong Planning* A *.

3. Pembangunan Perangkat Lunak

Model proses pembangunan perangkat lunak yang digunakan adalah model *Prototype*. Langkah-langkah dalam model *Prototype* adalah.

a. Analisis

Menganalisis hal-hal yang diperlukan dalam melakukan identifikasi langkah *Pathfinding*. Dalam tahap ini dilakukan analisis metode dan juga pengumpulan kebutuhan informasi dan data dengan cara mengumpulkan jurnal, paper, serta informasi yang berkaitan dengan penelitian yang akan dilakukan.

b. Kebutuhan Data

Tahap pengumpulan data berupa jumlah penghalang tembok, jumlah npc, dan titik tujuan.

c. Pembangunan *Prototype*

tahap implementasi dari proses analisis dan kebutuhan data sistem yang sudah dilakukan.

d. Evaluasi *Prototoype*

Ttahap pengujian terhadap aplikasi *Prototype* yang dibangun.

4. Pengujian Aplikasi

Pengujian Aplikasi adalah melakukan pengujian terhadap metode yang sudah diimplementasikan ke dalam program.

5. Hasil Penelitian

Hasil penelitian merupakan tahap akhir dalam penelitian ini. Pengujian yang dilakukan menghasilkan Langkah *multi Agent NPC* dan akurasi yang baik atau tidak.

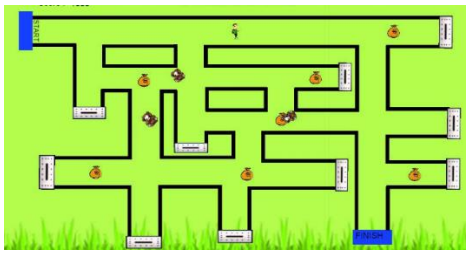
2. ISI PENELITIAN

2.1 *Game* Labirin

Pada umumnya, labirin dibuat untuk tujuan hiburan. Dalam kehidupan nyata, labirin dapat ditemukan pada susunan jalan kecil atau gang-gang di kawasan perumahan. Sangat sulit bila seseorang yang asing dengan daerah tersebut untuk mencari jalan. Labirin adalah sebuah *puzzle* dalam bentuk percabangan jalan yang kompleks dan memliki banyak jalan buntu. Tujuan permainan ini adalah pemain harus menemukan jalan keluar dari sebuah pintu masuk ke satu atau lebih pintu keluar. Bisa juga kondisi pemain menang yaitu ketika dia mencapai suatu titik atau tujuan di dalam labirin tersebut.

Dalam kehidupan sehari-hari terdapat beberapa aplikasi labirin yang dapat dijumpai terutama dalam industry *game*, karena labirin menantang pemain untuk mencari jalan keluar dari titik yang ditentukan sebelumnya. Selain itu dibeberapa Negara dibuat labirin dengan ukuran manusia dan menantang orang-orang untuk masuk kedalamnya, sebagai salah satu atraksi untuk menarik wisatawan [7].

Berikut ini adalah gambar dari salah satu *game* labirin :

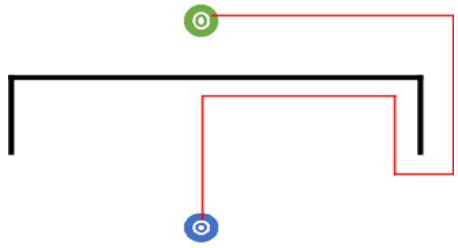


Gambar 2. *Game* labirin

2.2 Pencarian Jalur

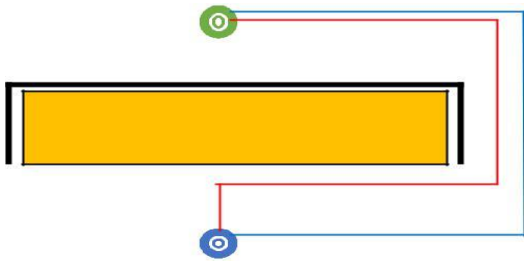
Pencarian jalur / rute (*pathfinding*) adalah salah satu penerapan yang ditangani oleh kecerdasan buatan dengan menggunakan algoritma pencarian [3].

Pengaplikasian yang dapat dilakukan dengan *pathfinding* adalah pencarian jalan pada peta dan pencarian rute atau jalur dalam suatu *game*. Algoritma *pathfinding* yang dipakai harus dapat mengenali jalan dan semua elemen peta yang tidak dapat dilewati. Sebuah algoritma *pathfinding* yang baik dapat di gunakan untuk mendeteksi beberapa halangan yang ada pada jalan dan menemukan jalur untuk menghindarinya, sehingga jalan yang ditempuh lebih pendek daripada yang seharusnya bila tidak menggunakan algoritma *pathfinding*. Lihat ilustrasi pada Gambar 3



Gambar 3. Penentuan rute tanpa *pathfinding*

Pada Gambar 3. dari awal yang berwarna hijau menuju akhir , tanpa adanya algoritma *pathfinding*, *npc* hanya akan memeriksa halangan atau rintangan dari lingkungan sekitarny saja. Unit tersebut akan maju terus untuk mencapai tujuan, baru setelah mendekati adanya halangan, kemudian jalan ke kanan dalam perjalanannya.



‘

Gambar 4. Penentuan rute dengan *pathfinding*

Selanjutnya, penentuan jalan dengan algoritma *pathfinding* pada gambar 4. akan merencanakan jalan kedepan untuk mencari jalur yang lebih pendek dan menghindari halangan atau rintangan yang dilambangkan garis biru muda untuk mencapai tujuan, tanpa berjalan ke dalam suatu halangan atau rintangan. karena itu fungsi dari algoritma *pathfinding* menjadi penting untuk menyelesaikan beberapa masalah dalam penentuan jalur.

2.3 Algoritma *Lifelong Plannning A** (LPA*)

Lifelong Planning A * adalah versi incremental dari A *, yang dapat beradaptasi dengan perubahan dalam grafik tanpa menghitung ulang seluruh grafik, dengan memperbarui nilai- g (jarak dari awal) dari pencarian sebelumnya selama pencarian saat ini untuk memperbaikinya saat diperlukan. *Lifelong Planning A* * menggunakan heuristik, yang merupakan batas bawah untuk cost jalur dari node yang diberikan ke tujuan. Pencarian pertamanya adalah sama dengan versi A * yang memutuskan hubungan antara simpul dengan nilai f yang sama dan mendukung nilai g yang lebih kecil. Menggunakan perhitungan algoritma sebagai berikut.

$$g*(s) = \begin{cases} 0 & if\ s = {}^{s}start, \\ \min_{s'} \in \mathrm{pred}(s)\big(g(s') + c(s',s)\big) & otherwise. \end{cases}$$

Dimana :

- S = simpul
- predecessors dari s = $pred\,(s) \in s$
- Cost perjalanan dari titik s ke titik s’=$0 < c(s,s') \leq \infty$
- Start titik = ${}^{s}start \in S$
- Start distance= panjang dari jarak terpendek $\mathbf{S}_{start}$ ke **S**
- g(s) = estimasi start distance g*(s)

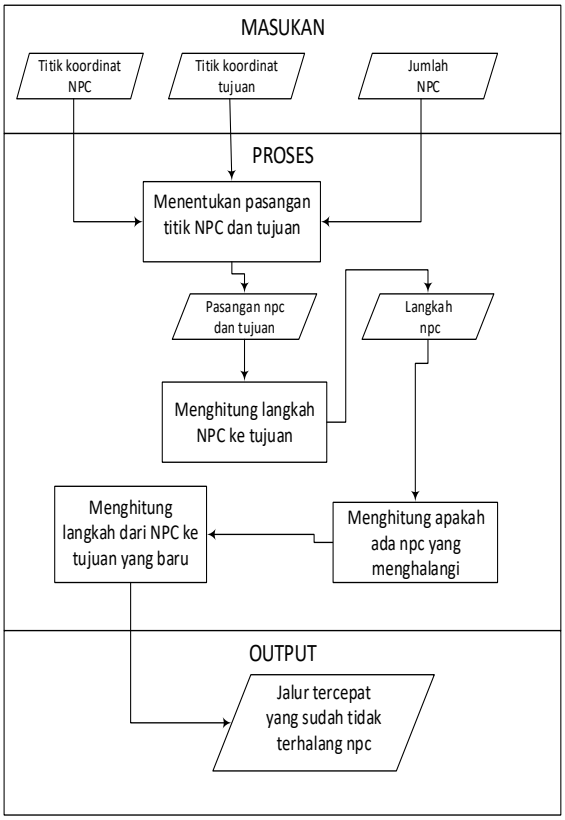
Berikut ini adalah algoritma dari *LPA** :

procedure Kalkulasikey(*s*)
return [min(*g*(*s*), *rhs*(*s*)) + *h*(*s*); min(*g*(*s*), *rhs*(*s*))];
procedure inisialisasi()
$U = \emptyset$;
for all $s \in S$ $rhs(s) = g(s) = \infty$;
$rhs(s_{start}) = 0$;
U.tambah(s_{start}, [$h(s_{start})$; 0]);
procedure UpdateSimpul(*u*)
if ($u \neq s_{start}$) $rhs(u) = \min_{s' \in pred(u)}(g(s') + c(s', u))$;
if ($u \in U$) U.hapus(*u*);
if ($g(u) \neq rhs(u)$) U.tambah(*u*, Kalkulasikey(*u*));
procedure Hitungjalurterpendek ()
while (U.TopKey() < Kalkulasikey(s_{goal}) OR $rhs(s_{goal}) \neq g(s_{goal})$)
u = U.Pop();
if ($g(u) > rhs(u)$)
$g(u) = rhs(u)$;
for all $s \in succ(u)$ UpdateSimpul(*s*);
else
$g(u) = \infty$;
for all $s \in succ(u) \cup \{u\}$ UpdateSimpul(*s*);
procedure Main()
Inisialisasi();
Hitungjalurterpendek();
Tunggu cost berubah;
for semua titik (*u*, *v*) dengan cost titik yang berubah
Update cost titik $c(u, v)$;
UpdateSimpul(*v*);

Gambar 5. Algoritma *Lifelong Planning* A*

2.4. Analisis Sistem

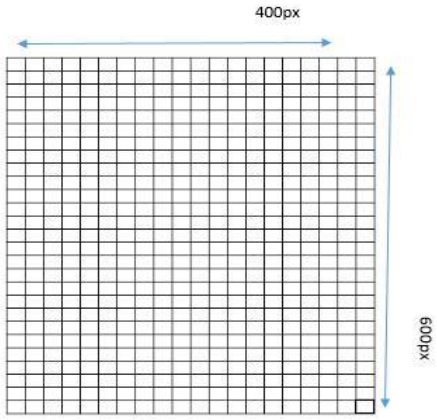
Sistem yang dibangun adalah sistem *Pathfinding* NPC pada permainan labirin. Sistem ini menghitung langkah dari beberapa NPC dari lokasi awal ke tujuan. Berikut merupakan gambaran dari sistem yang akan dibangun :



Gambar 6. Gambar umum sistem

2.5 Analisis Masukan

Analisis masukan menjelaskan data masukan yang akan diperlukan dalam penerapan algoritma LPA* .Analisis masukan yang dibutuhkan pada algoritma LPA* yaitu posisi awal *Agent*, posisi akhir *Agent*, posisi halangan dan kedalaman pencarian. Sebelum menghitung menggunakan algoritma LPA* akan dihitung dulu biaya (cost) sebenarnya yang terdapat pada jalur itu sendiri. Map yang digunakan pada simulasi ini dengan skala pixel yang berukuran lebar 400 pixel dan tinggi 600 pixel. Satu layer tersebut disimpan dalam *grid* (suatu kotak-kotak kecil) yang dinamis dimana user dapat menentukan jumlah gridnya. Setiap grid memiliki ukuran dengan lebar 20 pixel dan tinggi 20 pixel, contoh map dengan ukuran lebar 20 pixel dan tinggi 20 pixel dapat dilihat pada gambar 7. berikut:

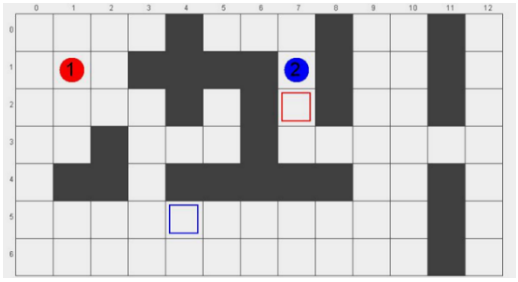


Gambar 7. Keterangan ukuran layar

Untuk mempermudah perhitungan maka ada beberapa nilai yang akan disederhanakan. Pertama nilai simpul yang tadinya 20x20 pixel diubah menjadi 1x1, artinya jarak simpul 1 ke simpul lainnya adalah 1. Jadi setiap simpul memiliki cost yang sama yaitu 1 secara horizontal maupun vertikal. Pergerakan yang dilakukan hanya bisa diagonal dan vertikal

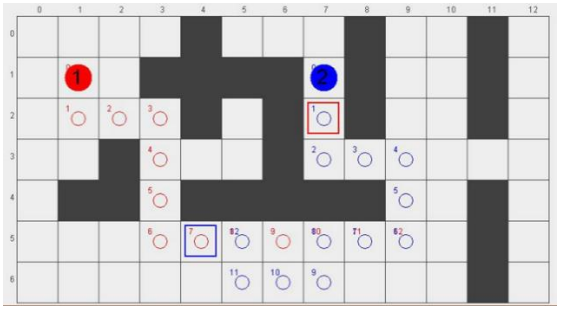
2.6 Contoh Kasus

Untuk lebih memahami algoritma *Lifelong Planning* A*, dapat dilihat dari contoh kasus seperti pada Gambar 8. Diasumsikan dalam simulasi ada 2 *Agent* (warna biru dan merah) yang memiliki goal masing – masing, selain itu juga ada tembok penghalang.



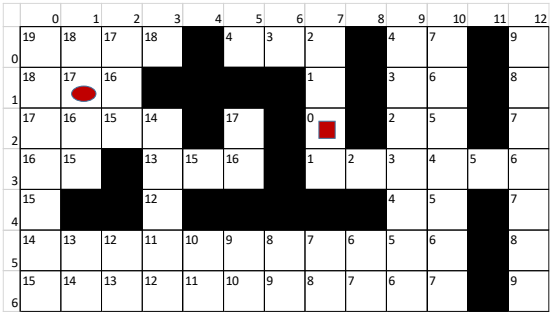
Gambar 8. Kondisi awal pencarian LPA*

Proses pertama yang dilakukan adalah menghitung heuristik dari *Agent* ke goal, mencari jalur terpendek dari node awal, penyimpanan node menggunakan tiga dimensi ruang-waktu (x,y,time). Setelah melakuan perhitungan awal, dan path awal telah ditemukan, setiap agen pindah ke tujuan node terdekat, seperti gambar 3.4 berikut ini:

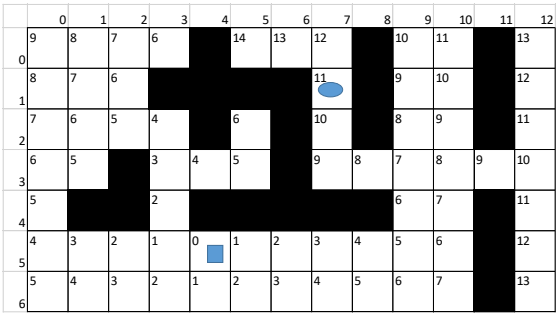


Gambar 9. Kondisi Awal dengan path yang akan dilalui

Berikut ini adalah heuristik dari *Agent* 1 ke goal 1 dan *Agent* 2 ke goal 2 dengan menggunakan manhattan distance



Gambar 10. heuristik *Agent* 1



Gambar 11. heuristik *Agent* 2

Pada iterasi ini algoritma LPA* melakukan penambahan pada variable currentWindow menjadi currentWindow+1 dan currentTime menjadi currentTime +1, setelah itu langkah selanjutnya adalah mengecek ulang, apakah ada unit yang harus di kalkulasi ulang atau tidak, unit yang akan dicek diambil secara random berdasarkan currentWindow.

2.7 Pengujian

Pengujian sistem dari penelitian ini dilakukan agar dapat mengetahui bagaimana aplikasi yang sudah dibuat dapat dijalankan sesuai atau tidak. Adapun dalam pengujian menggunakan tiga teknik yaitu pengujian fungsional, pengujian performansi serta pengujian akurasi.

2.7.1 pengujian fungsional

Pengujian fungsional dilakukan untuk mengecek dari fungsionalitas yang sudah dibuat telah berjalan atau tidak, berikut ini adalah skenario dari pengujian fungsional.

Tabel 1. Hasil dan kasus pengujian (data normal)

Aksi	Yang akan diharapkan	Kesimpulan
Memasukan titik awal	Menampilkan titik awal pada labirin	Diterima
Memasukan titik tujuan	Menampilkan titik tujuan pada labirin	Diterima
Memasukan data penghalang	Menampilkan penghalang pada labirin	Diterima
Menekan tombol *step*	Menampilkan rute pertama setiap NPC pada labirin	Diterima
Menekan tombol *Animasi*	Menampilkan setiap NPC bergerak dari titik awal	Diterima
Menekan tombol *Stop*	Menampilkan setiap NPC pada posisi terakhir	Diterima
Menekan tombol *reset*	Menampilkan map dengan tanpa halangan dan NPC	Diterima
Memasukan data nilai baris Contoh: 13	Dapat terisi *number* untuk masukan data nilai baris	Diterima

Berdasarkan pengujian terhadap setiap fungsional menggunakan metode *black box* yang sudah dibuat , maka ditarik kesimpulan bahwa sistem yang dibuat sudah bekerja sesuai dengan fungsional yang di harapkan.

2.7.2 pengujian performansi

Performasi yang akan diuji pada aplikasi yang sudah dibuat yaitu waktu tempuh dari posisi awal hingga posisi akhir setiap unit dan jumlah simpul yang diperiksa pada setiap siklus.

Performansi yang diuji pada aplikasi yang telah dibuat yaitu simpul yang diperiksa dan waktu yang ditempuh yang dihasilkan dengan Implementasi ordo minimal 5 x 5 tanpa penghalang dengan jumlah agen dari 2 hingga 8.

Tabel 2. Hasil pengujian ekspansi node

Ordo \ jumlah	2	4	6	8
5X5	96 ms	357 ms	771 ms	892ms
10X10	309 ms	870 ms	1074 ms	1798 ms
15X15	622 ms	1154	1793 ms	2486 ms

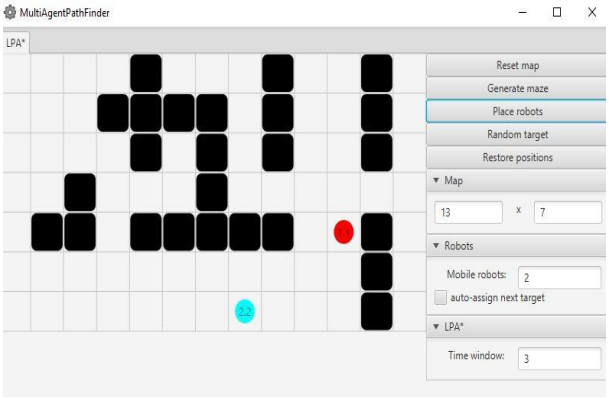
Hasil pengujian waktu eksekusi pada ordo 5x5, 10x10 dan 15x15 tanpa penghalang dengan jumlah agen dari 2 – 8 dapat dilihat pada tabel 3.

Tabel 3. Hasil waktu eksekusi

Ordo \ jumlah	2	4	6	8
5X5	4.33ms	8.81ms	15.06ms	55.46ms
10X10	38.68 ms	60.16ms	90.99ms	107.46ms
15X15	60.25 ms	93.12ms	141.77ms	166.99ms

2.7.3 Pengujian Akurasi

Setelah dilakukan pengujian performansi maka akan dilakukan dengan pengujian akurasi menggunakan algoritma LPA* dan A* sebanyak 30 kali dengan menggunakan penghalang dan target maupun *Agent* diletakkan secara acak. contoh penghalangnya sebagai berikut :



Gambar 12. contoh uji akurasi

Hasil pengujian akurasi pada *Agent* 2 - 8 dengan pengujian jumlah yang menemukan jalur, jumlah yang ditempuh dan akurasi dapat dilihat pada tabel 4 dan tabel 5.

Tabel 4 . uji akurasi LPA*

Jumlah *Agent*	*Agent* sesuai jalur	Node di tempuh	akurasi(%)
2	30	10	100,0
4	29	11	96,7
6	26	13	86,7
8	25	15	83,3
		Rata rata akurasi	91,7

Tabel 5 . uji akurasi A*

Jumlah *Agent*	*Agent* sesuai jalur	Node di tempuh	akurasi(%)
2	30	17	100,0
4	25	18	83,3
6	23	19	76,7
8	19	22	63,3
		Rata rata akurasi	80,8

3. KESIMPULAN DAN SARAN

3.1 Kesimpulan

Berdasarkan hasil dari analisis, implementasi dan pengujian, dapat diambil kesimpulan sebagai berikut:

1. Algoritma *Lifelong Planning* A* dapat digunakan untuk melakukan *pathfinding* banyak *Agent* npc dalam permainan labirin.
2. Berdasarkan pengujian akurasi yang dilakukan menggunakan 8 *Agent* dan sebanyak 30 kali pengulangan diketahui bahwa tingkat akurasi dalam keberhasilan *Agent* npc yang menemukan jalurnya adalah 25 *Agent* dengan jumlah node yang ditempuh sebanyak 15 node. Tingkat akurasi memiliki persentase sebesar 91,7 %. Dan memiliki nilai kecepatan sebesar 166.999 ms dalam lamanya waktu eksekusi dengan ordo 15x15 dan 8 *Agent* Sehingga dapat dikatakan bahwa tingkat akurasi pada penerapan algoritma *Lifelong Planning* A* sudah baik.

3.2 Saran

Adapun Saran yang dapat diberikan untuk pengembangan selanjutnya yaitu untuk penerapan algoritma lifelong plannning A* dapat dicoba dengan menambahkan jumlah tujuan, menggunakan variant terbaru dari algoritma pencarian jalur terpendek yang terbaru, dan juga tingkatkan jumlah perilaku dari NPC sehingga mengurangi kegagalan dalam menentukan pencarian jalur.

DAFTAR PUSTAKA

[1] Ramadhan, Ilham. 2017. "Implementasi Algoritma Jump Point Search Pada NPC Musuh Untuk Mengejar Pemain Dalam *Game* Labirin (Skripsi)". Bandung: Universitas Komputer Indonesia

[2] Henry, Samuel. 2010. Cerdas dengan *Game*. Jakarta: PT Gramedia Pustaka Utama.

[3] Asmiatun, Siti. 2017. Belajar Membuat *Game* 2D dan 3D Menggunakan Unit. Yogyakarta : Penerbit Budi Utama.

[4] Teresa,Dillon. 2004. Adventure *Game*s for Learning and Storytelling. A Futurelab prototype context paper: Adventure Author, FutureLab Report.

[5] Andang, Ismail. 2009. Education *Game*s (Menjadi cerdas dan ceria dengan permainan edukatif). Yogyakarta : Pilar Media

[6] Thomas. 2006. Genre and *game* studies : Toward a critical approach to video *game* genres. Simulation & Gaming, Vol. 37, University of Melbourne

[7] Imam Ahmad, W. W. 2017. Penerapan Algoritma A Star (A*) pada *Game* Petualangan Labirin Berbasis Android. Jurnal Ilmu Komputer Dan Informatika.

[8] Kusumadewi, Sri. 2002. Artificial Intelligence . Yogyakarta, Graha Ilmu.

[9] Suyanto. 2007. Artificial Intelligence. Bandung: Informatika.

[10] J.E. Kendall dan K.E. Kendall. 2006. *System*s Analysis and Design Berbasis Android. Bandung : Informatika.

[11] Surianto, Sri Anggraini. 2014. "Implementasi Algoritma Iterative Deepening A* (IDA*) Dengan Stochastic Node Caching (SNC) Untuk *Pathfinding* Musuh Pada *Game* Labirin (Skripsi)". Bandung: Universitas Komputer Indonesia