PENERAPAN A* PATHFINDING DAN FSM (FINITE STATE MACHINE) PADA GAME "LOST CIVILIZATION" BERBASIS ANDROID

Imam Satrio, Febriana Santi Wahyuni, Deddy Rudhistiar
Program Studi Teknik Informatika S1, Fakultas Teknologi Industri
Institut Teknologi Nasional Malang, Jalan Raya Karanglo km 2 Malang, Indonesia
1818011@scholar.itn.ac.id

ABSTRAK

Game adalah salah satu sarana untuk hiburan atau sekedar melepas penat, Sebagian besar masyarakat Indonesia banyak memainkan game lewat ponsel mereka, untuk saat ini game banyak diracang dalam bentuk 2D maupun 3D, tidak jarang juga dalam setiap ponsel pasti terpasamg sebuah aplikasi game, apalagi di era teknolgi sekarang game mudah didapatkan dengan melalui aplikasi penyedia game seperti *Playstore* dan *Appstore.* Dalam sebuah game juga banyak mengimplementasikan berbagai kecerdasan buatan guna untuk membuat perilaku npc *(non player character*) mirip dengan perilaku manusia. Game ini mengimplementasikan metode A* *pathfinding* untuk mencari lokasi player dengan memperhitungkan jarak total terkecil yang dilali setiap node dan menggunakan metode FSM (*Finite State Machine*) untuk membuat tingkah laku musuh dengan menggunakan tiga hal yaitu State (Keadaan), Event (kejadian) dan action (aksi). Pengujian kecerdasan buatan yag diimplementasikan akan diuji dengan melakukan pengujian model *alpha testing* untuk A* *pathfinding* dan *Finite State Machine,* pengujian blackbox juga dilakukan guna menguji apakah fungsi sistem yang sudah dubuat dapat berjalan dengan baik atau tidak. Dari pengujian yang telah dilakukan tersebut didapatkan hasil implementasi metode algorita A* dengan tingkat 100% akurasi dalam menemukan lokasi player dalam menentukan rute tercepat agar menemukan lokasi *player* dengan menggunakan nilai total terkecil didapat dari perhitungan setiap *node*. *Finite State Machine* pada *enemy* juga sudah bisa melakukan aksi sesuai dengan perilaku player, dari pengujian 10 orang responden, didapatkan hasil 7 orang menyatakan baik sekali dan 3 orang menyatakan baik.

Kata kunci : *Unity, Game, A* Pathfinding, FSM, Android.*

1. PENDAHULUAN

Video game merupakan sebuah permainan yang dimainkan dalam perangkat elektronik, sistem elektronik yang digunakan disebut sebagai platform contohnya seperti konsol game, komputer pribadi hingga telepon pintar yang digunakan sehari-hari, game saat ini sedang sangat diminati karena sebagai sarana melepas penat. Khususnya untuk perangkat mobile, Berbagai jenis game sudah dapat dengan mudah ditemukan dalam sebuah perangkat mobile, sama halnya dengan aplikasi kebutuhan lainnya, aplikasi game juga dapat didapatkan dari sebuah layanan penyedia aplikasi.

Game "Lost Civilization" adalah sebuah game 2D side scroller, action, adventure puzzle berbasis android yang mengharuskan pemain menjelajah map dan level yang ada dalam game, karena bergendre advanture, serta pemain diharuskan memecahkan beberapa teka-teki yang ada dalam game.

Metode yang digunakan dalam game tersebut adalah finite FSM (Finite State Machine) dan Pathfinding, FSM digunakan untuk membuat sebuah pengambilan keputusan, yaitu sistem kontrol yang menggambarkan tingkah laku atau prinsip kerja sistem dengan menggunakan state (Keadaan), event (kejadian) dan action (aksi). Algoitma A* juga digunakan dalam pembuatan game ini yang digunakan untuk mendukung metode pathfinding sebagai musuh untuk mencari rute terpendek berguna untuk mengejar pemain.

Unity merupakan sebuah game engine untuk mengembangkan game multi platform dalam lingkup 2D maupun 3D, yang berarti game yang dibuat mendukung banyak format file serta dapat beroperasi dalam berbagai perangkat dan sistem operasi seperti, windows, mcOS, iOS, android. Grafis pada unity juga menggunakan grafis tingkat tinggi seperti OpenGL dan dan DirectX Dalam unity juga banyak disediakan berbagai macam tool yang dapat memudahkan pengembangan game.

Sesuai dengan uraian tersebut, penulis ingin membuat sebuah game berbasis android dengan mengimplementasikan beberapa kecerdasan buatan yaitu A* pathfinding dan Finite State Machine untuk pengambilan keputusan NPC (Non Player Character) menggunakan game engine unity, game ini dibuat dalam bentuk 2D dengan tipe side scroller dan mnyungsung genre Action Adventure Puzzle yang mengharuskan pemain mengalahkan musuh dan menjelajah berbagai area demi mencapai tujuan akhir game

2. TINJAUAN PUSTAKA

2.1. Definisi Kecerdasan buatan

Game jika diartikan ke dalam Bahasa Indonesia berarti permainan. Dalam bermain game atau permainan pasti akan berakhir dengan salah satu pihak yang menang dan yang satunya kalah. Permainan adalah sistem dimana pemain terlibat dengan konflik buatan serta dapat berinteraksi dengan sistem dan

konflik buatan tersebut menurut Lestari D (Lestari D, 2012). Video game adalah sebuah permainan yang menggunakan perangkat elektronik yang melibatkan antarmuka pemain dan penggunaan perangkat masukan seperti kendali kontrol yang didalamnya tedapat sebuah peraturan bermain berguna untuk membatasi perilaku pemain untuk menentukan permainannya Game lebih sering dimainkan oleh anak-anak, akan tetapi pada zaman sekarang orang dewasa juga suka bermain game dan mengikuti perkembangan game-game yang ada sekarang.

2.2. Definisi Kecerdasan buatan

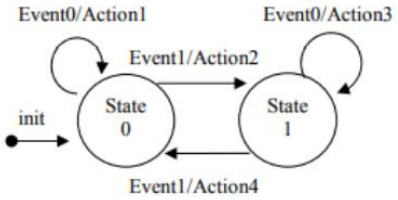
Kecerdasan Buatan (Artificial Intelligence) merupakan salah satu bagian dari ilmu komputer yang mempelajari bagaimana membuat mesin (komputer) dapat melakukan pekerjaan seperti dan sebaik yang dilakukan oleh manusia.

Menurut John McCarthy, 1956, AI: untuk mengetahui dan memodelkan proses–proses berpikir manusia dan mendesain mesin agar dapat menirukan perilaku manusia. Cerdas, berarti memiliki pengetahuan ditambah pengalaman, penalaran (bagaimana membuat keputusan dan mengambil tindakan), moral yang baik (Dahria, 2008). Manusia cerdas dalam menyelesakan masalah, dikarenakan manusia memilki pengetahuan sejak lahir, pengetahuan didapat dari proses belajar dan berpengalaman, jadi semakin banyak pengetahuan dan pengalaman, semakin mampu dalam menyelesaikan suatu masalah. Tapi bekal pengetahuan tidak cukup, manusia juga diberi akal untuk melakukan penalaran, mengambil keputusan berdasarkan pengetahuan dan pengalaman yang dimiliki. Tanpa memiliki kemampuan untuk menalar dengan baik, manusia dengan segudang pengalaman dan pengetahuan tidak dapat menyelesaikan masalah dengan baik.

2.3. Jenis-Jenis Kecerdasan Buatan

Terdapat beberapa kecerdasan buatan yang biasanya digunakan dalam pembuatan game, seperti *Decision Making* dan *Pathfinding*.

1. Decision Making
   Jenis kecerdasan buatan decision making contohnya seperti Fuzzy Logic dan FSM (Finite State Machine).
   a. Fuzzy logic
      Fuzzy logic diperkenalkan oleh Prof. Lotfi Zadeh pada tahun 1965, yaitu metode dengan kemampuan memproses variable yang tidak pasti dan bersifat samar, atau yang tidak bisa dideskripsikan secara tepat. Dalam fuzzy logic menyediakan cara dengan memahami kinerja sisitem input dan output dari hasil pengamatan. Logika fuzzy adalah peningkatan dari Boolean, yang mana nilai Boolean hanya berdasar pada tidak dan ya atau 0 dan 1, maka untuk fuzzy tingkat keanggotaan adalah 0 sampai 1 berdasarkan dengan teori kemungkinan
   b. FSM (Finite State Machine)
      FSM (*Finite State Machine*) adalah sebuah metodologi perancangan sistem kontrol yang menggambarkan tingkah laku atau prinsip kerja sistem dengan menggunakan tiga hal berikut, *State* (Keadaan), *Event* (kejadian) dan *action* (aksi), yang berarti model perhitungan berdasarkan mesin hipotetis yang dibuat dari satu atau lebih kedudukan. Hanya satu kedudukan yang bisa aktif pada saat bersamaan, sehingga mesin harus berpindah dari kedudukan ke kedudukan lain agar bisa melakukan tindakan yang berbeda.



Gambar 1. Diagram FSM sederhana (Sumber: Setiawan 2006)

2. Pathfinding
   a. Algoritma A*
      A star merupakan salah satu algoritma pathfinding terbaik dalam mencari sebuah jalur terpendek menggunakan perhitungan terkecil dari jalur awal sampai jalur akhir, hingga mendapat total biaya lintasan seminimal mungkin, algoritma ini pertama kali dikemukakan oleh Peter Hart, Nils Nilsson, dan Bertram Raphael pada tahun 1968 Perhitungan pada Algoritma A Star (A*) dapat ditentukan sebagai berikut:

$$F(x) = G(x) + H(x) \quad (1)$$

      Keterangan:
      1. G(x) adalah nilai pada pergerakan simpul awal menuju simpul berikutnya.
      2. H(x) adalah perkiraan nilai pergerakan simpul awal menuju tujuan akhir simpul. Fungsi ini seringkali disebut dengan fungsi heuristik, dinamakan heuristik karena perhitungan tersebut berdasarkan perkiraan (guess).
      3. F(x) adalah jumlah nilai dari fungsi G(x) danH(x). dengan nilai terkecil F(x) adalah jalur terpendek menuju tujuan akhir. (Maulana, Sofwan, Isnanto, 2011)

      Terdapat ketentuan pada grafik agar algoritma A Star (A*) ini bila diterapkan akan selalu mendapatkan jalan yang terpendek. ketentuan tersebut yang harus dipenuhi pada grafik yaitu.
      1. Setiap simpul (node) dalam grafik memiliki jumlah terbatas pada area pencariannya.
      2. Pada pencarian terdapat jalan yang dilalui untuk mencapai tujuan.

3. Fungsi F(x) pada grafik bernilai rendah daripada fungsi F(x) pada pencarian sebelumnya.

b. Algoritma Dijkstra
Algoritma dijsktra ditemukan oleh Edger Wybe Dijkstra pada tahun 1959, algoritma ini mencari titik dengan jarak dari titik awal paling pendek dan kemudian akan diperbarui bila menemukan titik baru yang lebih pendek. Syarat bobot nilai dalam algoritma ini adalah bilangan positif dan tidak boleh negative, missal jika ditemukan sebuah nilai negative maka penyelesaian yang diberikan adalah infiniti (tak terhingga).

c. Algoritma Min Max
Min max merupakan algoritma untuk pencarian khusus dengan fungsi menentukan Langkah terbaik selanjutnya dalam sebuah permainan dengan melibatkan dua pemain, yaitu satu pemain melawan dengan computer. Algoritma ini bekerja secara rekrusif dengan menimalkan tingkat peluanh keberhasilan pemain dengan algoritma min max menganilisis seluruh pohon pemain dengan setiap langkahnya akan dipilih computer yang membuat lawan memiliki pelaung keberhasilan minimum dan komputer mendapat keuntungan maksimum

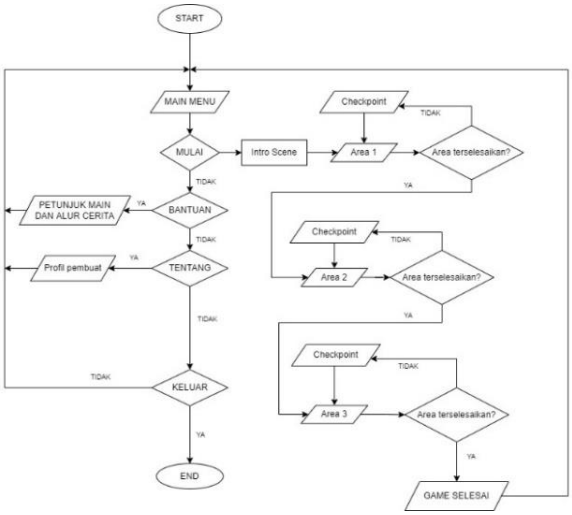
d. BFS (Breadth First Search)
Breadth First Search adalah salah satu algoritma yang digunakan dalam pencarian jalur sederhana dengan menggunakan daftar, open dan closed, dalam mengetahui Gerakan pencarian dalam ruang keadaan, keuntungan dalam algoritma ini adalah mengurangi beban komputasi karena yang diberikan hanyalah solusi dan harapan saja dan akan berhenti apabila solusi sudah mendekati yang terbaik

e. DSF (Depth Search First)
Algoritma DFS melakukan pencarian jalur pada satu pohon dalam setiap level dari paling kiri sampai menemukan sebuah solusi, missal jika pada level kiri masih belum ditemukan solusi maka pencarian dilanjutkan dalam node sebelah kanan dan node kiri akan dihapus dari memori, tahap tersebut akan berlanjut sampai menemukan sebuah solusi. Algoritma ini berawal dengan membangkitkan antrian menjadi masukan, antrian pertama akan diproses dan dicocokan dengan nilai masukan dan nilai tujuan, jika iya maka masukan pertama adalah solusi dan proses berakhir, jika tidak antrian pertama dimasukkan ke antrian kedua dan proses akan diualang sama dengan tujuan

3. METODE PENELITIAN

Pada bagian ini memuat metode yang digunakan pada pemuatan skripsi.

3.1. *Flowchart Game*



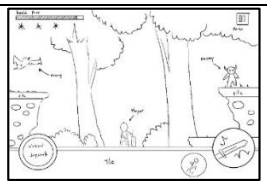
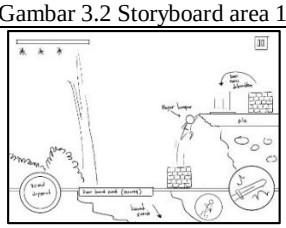
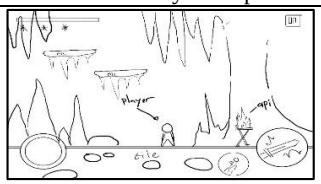
Gambar 2. Alur Flowchart Game

Pada proses pertama dari flowchart alur game tersebut adalah berada pada main menu yang berisikan mulai, bantuan, tentang dan keluar. Pemain memilih mulai kemudian akan memuat sebuah intro scene yang berisi tentang cerita dalam game, kemudian pemain akan diposisikan pada area 1 dan menjalankan game tersebut, misal apakah area tersebut sudah terselesaikan, jika gagal maka akan kembali ke checkpoint yang berada pada sekitar area 1, jika sudah berhasil maka akan lanjut ke area 2 dengan tingkat kesulitan yang ditingkatkan sampai dengan area yang sudah ditentukan, sampai game benar-benar berakhir dan kembali ke menu utama. Pada main menu jika pemain memilih bantuan akan menampilkan cara bermain dan sebuah synopsis dari cerita game, jika pemain memilih tentang maka akan menampilkan halaman pengembang game, jika pemain memilih keluar maka pemain akan dikembalikan menuju home android.

3.2. *Story Board*

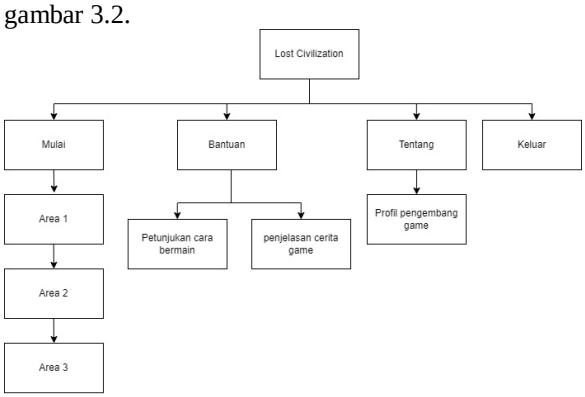
Storyboard adalah rancangan umum suatu aplikasi yang disusun secara berurutan layer-layer serta dilengkapi penjelasan dan spesifikasi dari setiap gambar, layer dan teks, storyboard juga dapat digunakan untuk merancang antar muka, karena interface juga merupakan bagian dari program yang berhubungan dengan pemain.

Tabel 1. Story Board

No	Nama	Board	Keterangan
1	Storyboard Area 1	 Gambar 3.2 Storyboard area 1	Area 1 merupakan awal permainan yang harus dilakukan, pemain akan mempelejari bagaimana berjalan, melompat menyeran dan sebagainya, area 1 beralatar di dalam hutan objektif utama adalah harus keluar dari hutan tersebut,
2	Storyboard puzzle	 Gambar 3.3 Storyboard puzzle	Pada setiap area pasti ada sebuah teka-teki yang harus diselesaikan jika ingin maju ke area selanjutnya atau jika ada item yang ingin didapat, pada storyboard dibawah jika ingin membuka pintu menuju bawah tanah maka harus meletakkan batu diatas tatakan penahan yang sudah disiapkan
3	Storyboard area 2	 Gambar 3.4 Storyboard area 2	Setelah keluar dari area 1, pemain akan melanjutkan ke area 2 yang mana tidak ada jalan lain selain melewati gua bawah tanah, pada area 2 latar tempat yang digunakan adalah gua.

3.3. Blok Diagram Sistem

Pada game "Lost Civilization" ini terdiri dari beberapa menu yaitu Mulai, Bantuan, Tentang, dan Keluar. Diagram struktur menu dapat dilihat pada gambar 3.2.



Gambar 3. Struktur Menu

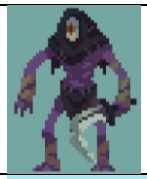
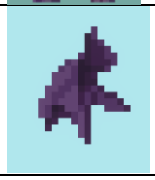
Pada diagram blok diatas, memperlihatkan terdapat 4 menu pilihan di dalam main menu, yaitu mulai, bantuan, tentang dan keluar. Pemain harus memilih menu mulai untuk memainkan game, jika pemain ingin melihat bagaimana cara bermain dan ingin membaca sedikit alur cerita game bisa memilih menu bantuan, menu tentang berisi profil pengembang game, menu keluar akan mengeluarkan game dan menuju *home screen android.*

3.4. Perancangan Karakter

Desain karakter sebagai berikut adalah karakter yang digunakan dalam game "Lost Civilization".

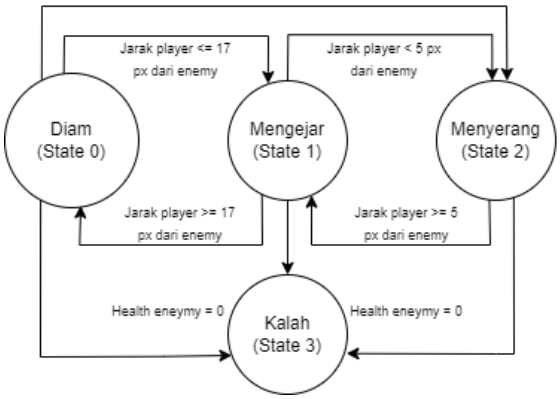
Tabel 2. Perancangan Karakter

No	Karakter	Keterangan
1		Karakter utama "Aruna" Karakter utama dan sebagai protagonis yang digunakan dalam cerita selama memainkan game, perempuan yang hidup sendirian, dia rela meninggalkan tempat tinggalnya di hutan hanya untuk mencari kelompok manusia lain yang hidup
2		Makhluk yang hanya berkeliling dalam satu tempat tetentu dan menyerang siapa saja yang mendekatinya, kapak Panjang yang digunakan memiliki skala serangan yang besar
3		Biasanya terperangkap di beberapa tempat, *enemy* ini bergerak sangat cepat tapi mudah mati, jika tidak segera dibunuh maka akan menjadi musuh yang sangat merepotkan karena dia akan mengikuti player kemana saja sambil menguras *health point player*
4		Tumbuhan berduri yang kurang diwaspadai, biasanya tumbuh pada ruang sempit, sekali menginjak akan menguras hp walaupun hanya sedikit

No	Karakter	Keterangan
5		Sering dijumpai di gua dan tempat-tempat gelap, memiliki *damage* serang yang besar dengan kecepatan serangan yang tinggi
6		Mereka memiliki skala serangan dan damage yang kecil tetapi menyerang secara bergerombol, banyak ditemui didalam gua dan tempat gelap
7		Kult mereka keras dan susah diserang, damage yang dihasilkan tidak mematikan, sering berdiam diri di dekat bebatuan

3.5. Perancangan Diagram FSM

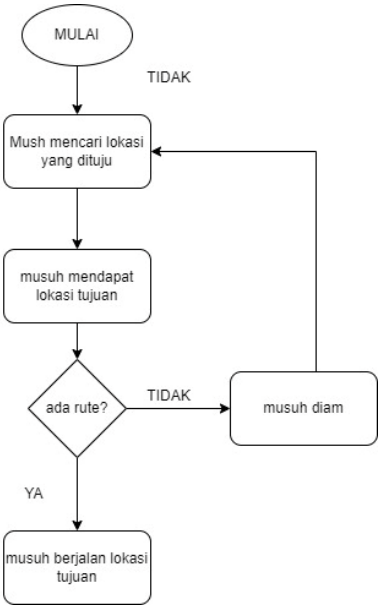
Penerapan alur FSM pada enemy di dalam game adalah misal player mendekati enemy dalam jarak < 17 px dari musuh, maka musuh akan mengejar jika >= 17 px akan diam dan tidak mengejar, pada saat mengejar dan jarak dari enemy adalah <= 5 px maka akan menyerang, kemudian jika keluar dari jarak serangan enemy akan tetap mengejar sampai jarak serangan. Misal jika enemy kehabisan *health point* maka enemy akan masuk state 3 yaitu kalah dan mati.



Gambar 4. Diagram FSM pada Enemy

3.6. Perancangan Diagram FSM

A* berguna melakukakn proses pencarian terdekat, musuh akan mencari lokasi pemain, pada game ini lokasi yang dimaksud adalah pemain, jika lokasi pemain sudah didapatkan maka enemy akan mencari rute sesuai dengan Algoritma, apakah enemy terdapat rute untuk menuju lokasi pemain, jika

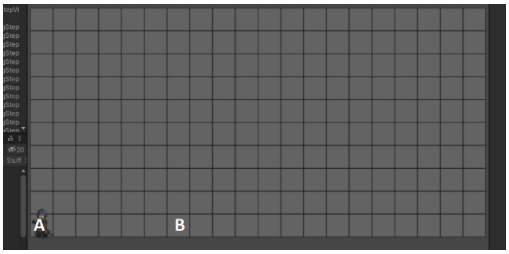


Gambar 5. Diagram Pathfinding A* pada Enemy

4. HASIL DAN PEMBAHASAN

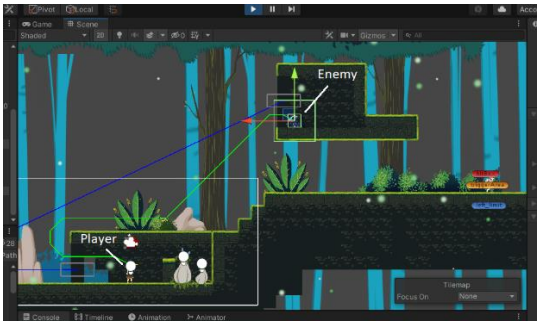
4.1. Alpha Testing A* Pathfinding

Pada pengujin ini dibuatlah sebuah *scene* baru yang dibuat khusus untuk menjalankan uji coba algoritma A*, dalam *scene* ini dilakukan uji coba misal A adalah titik awak dimana karakter mulai bergerak dan B adalah goal target yang harus dituju. Perhitungan dimulai dari *node* awal A sebagai titik awal rute, setiap *state* memiliki nilai H(x) dan nilai biaya tersendiri sebagai perkiraan nilai pergerakan simpul awal menuju tujuan akhir simpul, node awal akan mencari dan menghitung node mana yang terdekat dan memiliki nilai terkecil dihitung dari biaya jarak dan nilai setiap node, misal jika sudah menemukan node terpendek, node tersebut akan mengulangi cara yang sama dengan memperhitungkan setiap nilai *node*



Gambar 6. Scene Khusus Untuk Pengujian A*

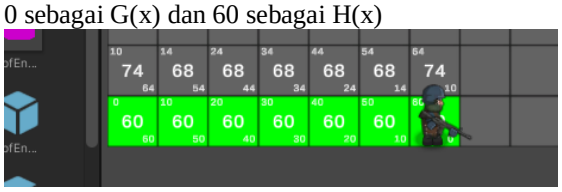
Dari hasil uji coba tersebut kemudian akan dimplementasikan ke dalam game yang sesungguhnya menggunakan metode A* yang sama, beberapa tipe musuh akan diberi kecerdasan buatan menggunakan metode algoritma A* ini untuk mencri tahu letak player saat dalam game, implementasi tersebut dapat dilihat digambar 6.



Gambar 7. Penerapan A* Pathfinding

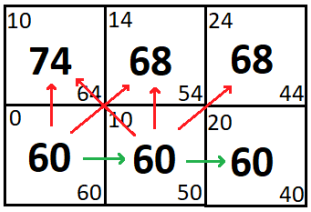
4.2. Pengujian A* Tanpa Menggunakan Tile Penghalang

Dalam pengujian ini dilakukan tanpa adanya penghalang untuk menentukan rute tercepat mana yang akan diambil dari titik A ke titik B, pada gambar dibawah ini rute tercepat yang dilalui adalah rute lurus untuk menuju node akhir, rute lurus tersebut didapatkan dari perhitungan nilai terkecil menggunakan perhitungan algoritma A* dengan rumus F(x) = G(x) + H(x), pada node awal, F(x) bernilai 60 yaitu hasil dari perhitungan rumus tersebut 0 sebagai G(x) dan 60 sebagai H(x)



Gambar 8. Uji Coba Tanpa Penghalang

Algoritma A* akan menghitung beberapa node terdekat yang memiliki nilai terkecil, node A memiliki nilai 60 kemudian ada beberapa node disekitarnya bernilai 74, 68, 60, melalui perhitungan dicari nilai terkecil yaitu 60, maka karakter akan bergerak ke nilai paling kecil, perhitungan tersebut akan terus diluang sampai tujuan.



Gambar 9. Penentuan Rute Selanjutnya Melalui Nilai Terkecil

Tabel 3. Perhitungan Nilai Rute Terkecil Selanjutnya

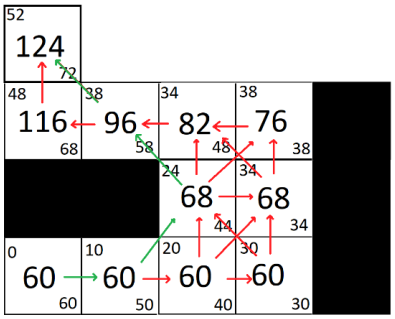
Node Awal	Node Tujuan Sekitar *F(x) = G(x) + H(x)*	Nilai terkecil	total
60 = 0 + 60	74 = 10 + 64		
	68 = 14+ 54		
	60 = 10 + 50	x	60+60=120
60 = 10 + 50	74 = 10 + 64		
	68 = 14+ 54		
	68 = 24+ 44		
	60 = 20+ 40	x	120+60=180

4.3. Pengujian A* Menggunakan Tile Penghalang

Pada pengujian ini ditambahkan adanya penghalang dengan titik tujuan yang sama, karena terdapat tile sebagai penghalang maka karakter tidak mungkin akan mengambil garis lurus tercepat seperti pengujian sebelumnya, karakter akan menemukan jalan keluar dengan menghitung setiap nilai node terkecil disekitar node awal.

Gambar 10. Uji Coba Mengunakan Penghalang

Karena pada node awal hanya ada satu node lanjutan maka rute selanjunya ada pada node tunggal tersebut, akan tetapi pada node kedua menemukan dua rute yang menunjukkan nilai 60 dan 68, secara tertulis nilai yang terkecil adalah 60, akan tetapi mengambil nilai terkecil bukan berarti akan membuat rute selanjutnya menjadi kecil, maka dari itu tetap dilakukan perhitungan A* pada nilai 60 tersebut yang menjadikan nilai yang lebih besar disekitarnya sebagai pemilihan rute terpendek karena tidak banyak melalui node.



Gambar 11. Penentuan Rute A* Jika Terdapat Penghalang

Tabel 4. Perhitungan Nilai Rute Terkecil Selanjutnya

Node Awal	Node Tujuan Sekitar *F(x) = G(x) + H(x)*	Nilai terkecil	total
60 = 0 + 60	60=10+50	x	120=60+60
60=10+50	68 = 24 + 44		
	60 = 20+ 40	x	180=120+60
60 = 20+ 40	68 = 24 + 44		
	68 = 34 + 34		
	60 = 30+ 30	x	240=180+60
60 = 30+ 30	68 = 34 + 34		
	68 = 24 + 44	x	308=240+68
68 = 24 + 44	82 = 34 + 48		
	76 = 38 + 38		
	68 = 34 + 34	x	376=308+68

Node Awal	Node Tujuan Sekitar $F(x) = G(x) + H(x)$	Nilai terkecil	total
68 = 34 + 34	82 = 34 + 48		
	76 = 38 + 38	x	452=376+76
76 = 38 + 38	82 = 34 + 48	x	534=452+82
82 = 34 + 48	96 = 38 + 58	x	630=534+96
96 = 38 + 58	116 = 48 + 68	x	746=630+116
	124 = 52 + 72		
116 = 48 + 68	124 = 52 + 72	x	870=746+124
Node yang sudah dilewati sebanyak 11 node	Total biaya rute saat ini menggunakan nilai terkecil node sekitar 870	Algoritma A* di Unity akan melakukan perhitungan dengan membadingkan nilai total biaya, maka akan ada rute dengan total biaya kecil yang sudah dilewati dan hasilnya sebagai berikut	
60 = 0 + 60	60=10+50	x	120=60+60
60=10+50	68 = 24 + 44	x	188=120+68
68 = 24 + 44	96 = 38 + 58	x	284=188+96
96 = 38 + 58	124 = 52 + 72	x	408=284+124
Node yang sudah dilewati sebanyak 5 node	Total biaya rute setelah melakukan hitungan perbandingan adalah 408	408 lebih sedikit dari 870 Jadi rute terpendek menggunakan biaya total 408 Paling sedikit biaya dan paling sedikit node daripada rute lainnya	

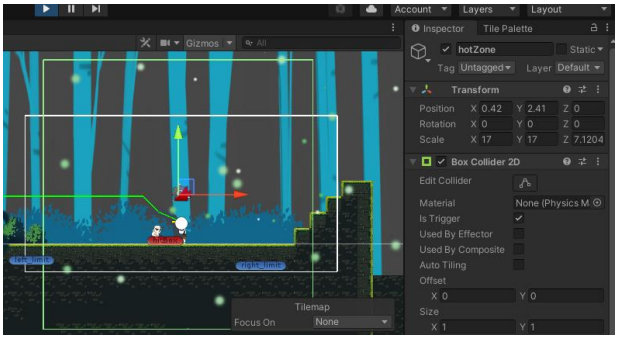
4.4. Alpha Testing Finite State Machine

Pengujian metode Finite State Machine yang diterapkan untuk pengendali animasi dan perilaku karakter musuh dilakukan pada saat permainan dijalankan menghasilkan data sebagai berikut

Tabel 5. Hasil Pengujian Metode FSM

Animasi karakter	kondisi	Keterangan	
		Sesuai	Tidak
Enemy patrol	Jika jarak player >= 17	x	-
Mengejar *Player*	Jika jarak player < 17	x	-
Menyerang *Player*	Jika jarak player < = 5	x	-
Enemy Death	Jika health musuh = 0	x	-

Diketahui jika dalam kondisi jarak musuh dari *player* adalah lebih dari jarak kejarnya yang bernilai 17, maka state yang dilakukan adalah patrol atau hanya berkeliling dalam area tertentu, jika kondisi jarak player dengan musuh <17 maka akan mengejar, semakin sedikit jarak musuh dengan player sampai pada kondisi dimana nilai <= 5 maka musuh akan menyerang *player*, sebagai pemain akan mennyerang musuh untuk mengalahkannya, jika *health* musuh = 0 maka akan dalam keadaan *enemy death.*

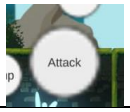
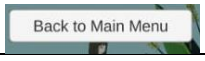


Gambar 12. Uji Coba FSM

4.5. Pengujian Blackbox

Pengujian black box dilakukan guna untuk mengamati hasil *running* aplikasi dalam segi funsionalitas dan mengamati *input output* apakah berjalan dengan baik.

Tabel 6. Hasil Pengujian Metode FSM

No	Input yang Diuji	Hasil yang Diharapkan	Hasil
1	Menekan tombol *start* 	Masuk kedalam game saat tombol ditekan	✓
2	Mengarahkan analog ke kanan kiri 	Karakter dapat berjalan ke kanan kiri	✓
3	Menekan tombol *jump* 	Karakter melompat keatas	✓
4	Menekan tombol *attack* 	Karakter melakukan animasi menyerang	✓
5	Menekan tombol *dash* 	Karakter melakukan gerakan kecil dengn cepat	✓
6	Menekan tombol *pause* 	Game akan berhenti sejenak dan menampilkan jendela *pause*	✓
7	Menekan tombol *resume* 	Game kembali berjalan	✓
8	Menekan tombol *back to main menu* 	Game kembali ke menu awal	✓
9	Menyerang lawan	Health enemy berkurang	✓

No	Input yang Diuji	Hasil yang Diharapkan	Hasil
		sampai dengan 0 dan kalah	
10	Player terkena serangan musuh	Health player berkuran sesuai damage yang diberikan enemy	✓
11	Menekan tombol exit 	Jika di Android maka akan keluar menuju *homescreen*	✓

5. KESIMPULAN DAN SARAN

Penerapan metode Finite State Machine menghasilkan perilaku *enemy* sesuai apa yang dilakukan *player*, pada pengujian algoritma A* tanpa penghalang, ditunjukkan bahwa rute dari titik A ke B menggunakan nilai total terkecil menggunakan rute lurus, Rute enemy algoritma A* yang menggunakan skenario penghalang menunjukkan bahwa, tidak selalu rute dengan nilai F(x) terkecil akan selalu dilalui, karena yang digunakan adalah nilai total *node* terkecil, jadi meskipun melalui *node* dengan nilai besar jika menghasilkan nilai total seminim mungkin besar kemungkinan akan dijadikan sebagai rute. Penerapan metode *Finite State Machine* pada jarak <17 px dari *player* terbukti jika *enemy* masuk dalam keadaan mengejar, sedangkan dalam jarak <=5 px *enemy* masuk dalam keadaan menyerang, hal ini menunjukkan perilaku cerdas npc sudah berjalan dengan baik sesuai dengan perilaku player, game dapat di download di: https://bit.ly/3JHSTYD

Penggunaan algorima A* pada game ini masih menggunakan grid, diharapkan implementasi kedepannya menggunakan area yang berbasis *navmesh* agar komputasi algoritma lebih cepat dalam menemukan titik tujuan.

DAFTAR PUSTAKA

[1] McCharty, J., 1989 Artificial Intelligence, logic and formalizing Common sense. In Philosophicallogicand artificialIntelligence (pp. 161-190) Springer Netherland

[2] Pamungkas, A., Widiyanto, E.P. and Angreni, R., 2014. Penerapan Algoritma A*(A Star) Pada Game Edukasi the Maze Island Berbasis Android.

[3] Tilawah, Hapsari., 2011. Penerapan Algoritma A-Star (A*) Untuk Menyelesaikan Masalah Maze. (Jurnal, Teknik Informatika, Institut Teknologi Bandung).

[4] Adi Wijaya, Surya., Susi Juniastuti, Supeno Mardi SN, dan Moch. Hariadi. 2009. Desain Fuzzy State Machine Untuk Menghasilkan Variasi Respon NPC (Non-Playable Character) Pada Sebuah Game. Program Studi MMT-ITS.

[5] McCarthy., 2014. Father of Artificial Intelligence, biography, LISP, arti-ficial intelligence, commonsense knowledge, India : Institute of ScienceBangalore