MENGENAL NABI DAN RASUL MELALUI GAME ADVENTURE 3D DENGAN METODE PARTIAL-EXPANSION A STAR (PEA*)

SKRIPSI

Oleh:
ALRIANDY PUTRA ADHA
NIM. 11650051

JURUSAN TEKNIK INFORMATIKA
FAKULTAS SAINS DAN TEKNOLOGI
UNIVERSITAS ISLAM NEGERI MAULANA MALIK IBRAHIM
MALANG
2018

MENGENAL NABI DAN RASUL MELALUI GAME ADVENTURE 3D DENGAN METODE PARTIAL-EXPANSION A STAR (PEA*)

SKRIPSI

Diajukan Kepada:
Fakultas Sains dan Teknologi
Universitas Islam Negeri (UIN)
Maulana Malik Ibrahim Malang
Untuk Memenuhi Salah Satu Persyaratan Dalam
Memperoleh Gelar Sarjana Komputer (S.Kom)

Oleh:
Alriandy Putra Adha
NIM. 11650051

JURUSAN TEKNIK INFORMATIKA
FAKULTAS SAINS DAN TEKNOLOGI
UNIVERSITAS ISLAM NEGERI MAULANA MALIK BRAHIM MALANG
2018

LEMBAR PERSETUJUAN

MENGENAL NABI DAN RASUL MELALUI GAME ADVENTURE 3D DENGAN METODE PARTIAL-EXPANSION A STAR (PEA*)

SKRIPSI

Oleh:
ALRIANDY PUTRA ADHA
NIM. 11650051

Telah disetujui oleh:

Dosen Pembimbing I,	Dosen Pembimbing II,
Dr. Muhammad Faisal, M.T	**Roro Inda Melani, MT., M.Sc**
NIP. 19740510 200501 1 007	**NIP. 19780925 200501 2 008**

Tanggal, 4 Juli 2018

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

Dr. Cahyo Crysdian
NIP. 19740424 200901 1 008

LEMBAR PENGESAHAN

MENGENAL NABI DAN RASUL MELALUI GAME ADVENTURE 3D DENGAN METODE PARTIAL-EXPANSION A STAR (PEA*)

SKRIPSI

Oleh:
ALRIANDY PUTRA ADHA
NIM. 11650051

Telah Dipertahankan di Depan Dewan Penguji Skripsi dan Dinyatakan Diterima Sebagai Salah Satu Persyaratan Untuk Memperoleh Gelar Sarjana Komputer (S.Kom)
Tanggal 30 Juni 2018

usunan Dewan Penguji			Tanda Tangan
. Penguji Utama	:	Hani Nurhayati, M.T NIP. 19780625 200801 2 006	( )
. Ketua Penguji	:	Yunifa Miftachul Arif, M.T NIP. 19830616 201101 1 004	( )
. Sekretaris Penguji	:	Dr. Muhammad Faisal, M.T NIP. 19740510 200501 1 007	( )
. Anggota Penguji	:	Roro Inda Melani, MT., M.Sc NIP. 19780925 200501 2 008	( )

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

Dr.Cahyo Crysdian
NIP. 19740424 200901 1 008

PERSEMBAHAN

Karya ilmiah skripsi ini saya persembahkan untuk kedua orang tua, Ayahanda dan Ibunda tercinta. Yang selama ini telah membesarkan dan mendidik dengan penuh kasih sayang, kesabaran dan keikhlasan. Dan semoga Allah SWT membalas semua kebaikan mereka. Amin.

PERNYATAAN KEASLIAN TULISAN

Saya yang bertanda tangan di bawah ini:

Nama : ALRIANDY PUTRA ADHA

NIM : 11650051

Fakultas / Jurusan : Sains dan Teknologi / Teknik Informatika

Judul Penelitian : **MENGENAL NABI DAN RASUL MELALUI GAME ADVENTURE 3D DENGAN METODE PARTIAL-EXPANSION A STAR (PEA*)**

Menyatakan dengan sebenar-benarnya bahwa hasil penelitian saya ini tidak terdapat unsur-unsur penjiplakan karya penelitian atau karya ilmiah yang pernah dilakukan atau dibuat oleh orang lain, kecuali yang secara tertulis dikutip dalam naskah ini dan disebutkan dalam sumber kutipan dan daftar pustaka.

Apabila ternyata hasil penelitian ini terbukti terdapat unsur-unsur penjiplakan, maka saya bersedia untuk mempertanggungjawabkan, serta diproses sesuai peraturan yang berlaku.

Malang, 4 Juli 2018

Yang membuat pernyataan

NIM. 11650051

MOTO

"Sebaik-baik manusia adalah yang paling bermanfaat bagi sesama"

(HR. Ahmad)

KATA PENGANTAR

Assalamu'alaikum Wr. Wb.

Segala puji bagi Allah SWT Tuhan semesta alam, karena atas segala rahmat dan karunia-Nya sehingga penulis mampu menyelesaikan skripsi dengan judul "Mengenal Nabi dan Rasul Melalui Game Adventure 3D Dengan Metode Partial Expansion A* (PEA*)" dengan baik dan lancar. Shalawat serta salam selalu tercurah kepada tauladan terbaik Nabi Muhammad SAW yang telah membimbing umatnya dari zaman kebodohan menuju Islam yang *rahmatan lil alamiin*.

Dalam penyelesaian skripsi ini, banyak pihak yang telah memberikan bantuan baik secara moril, nasihat dan semangat maupun materiil. Atas segala bantuan yang telah diberikan, penulis ingin menyampaikan doa dan ucapan terimakasih yang sedalam-dalamnya kepada:

1. Prof. Dr. H. Abdul Haris, M.Ag , selaku Rektor Universitas Islam Negeri Maulana Malik Ibrahim Malang beserta seluruh staf. Bakti Bapak dan Ibu sekalian terhadap UIN Maliki Malang turut membesarkan dan mencerdaskan penulis.
2. Dr. Sri Harini, M.Si, selaku Dekan Fakultas Sains dan Teknologi Universitas Islam Negeri Maulana Malik Ibrahim Malang beserta seluruh staf. Bapak dan ibu sekalian sangat berjasa memupuk dan menumbuhkan semangat untuk maju kepada penulis.
3. Dr. Cahyo Crysdian, selaku Ketua Jurusan Teknik Informatika Universitas Islam Negeri Maulana Malik Ibrahim Malang, yang sudah memberi banyak pengetahuan, inspirasi dan pengalaman yang berharga.
4. Dr. Muhammad Faisal, M.T selaku dosen pembimbing I yang telah meluangkan waktu untuk membimbing, memotivasi, mengarahkan dan memberi masukan kepada penulis dalam pengerjaan skripsi ini hingga akhir.

5. Roro Inda Melani, MT., M.Sc selaku dosen pembimbing II yang juga senantiasa memberi masukan dan nasihat serta petunjuk dalam penyusunan skripsi.
6. Bapak, Ibu, dan Adik-adik serta keluarga besar tercinta yang selalu memberi dukungan yang tak terhingga serta doa yang senantiasa mengiringi setiap langkah penulis.
7. Segenap Dosen Teknik Informatika yang telah memberikan bimbingan keilmuan kepada penulis selama masa studi.
8. Teman – teman seperjuangan Teknik Informatika 2011, dan teman-teman yang telah mendukung dan menyemangati.
9. Para peneliti yang telah mengembangkan *Game* dengan *Engine Unity* yang menjadi acuan penulis dalam pembuatan skripsi ini. Serta semua pihak yang telah membantu yang tidak bisa disebutkan satu satu. Terimakasih banyak.

Berbagai kekurangan dan kesalahan mungkin pembaca temukan dalam penulisan skripsi ini, untuk itu penulis menerima segala kritik dan saran yang membangun dari pembaca sekalian. Semoga apa yang menjadi kekurangan bisa disempurnakan oleh peneliti selanjutnya dan semoga karya ini senantiasa dapat memberi manfaat. Amin. *Wassalamualaikum Wr. Wb.*

Malang, 4 Juli 2018

Penulis

DAFTAR ISI

DAFTAR GAMBAR

DAFTAR TABEL

ABSTRAK

Adha, Alriandy Putra. 2018. **Mengenal Nabi dan Rasul Melalui Game Adventure 3D dengan Menggunakan Metode Partial Expansion A Star (PEA*).** Skripsi. Jurusan Teknik Informatika Fakultas Sains dan Teknologi Universitas Islam Negeri Maulana Malik Ibrahim Malang.

Pembimbing: (I) Dr. Muhammad Faisal, M.T, (II) Roro Inda Melani, MT., M.Sc

Kata Kunci: *Nabi dan Rasul, A*, Partial Expansion A**

Iman kepada nabi dan rasul Allah, merupakan rukun iman yang keempat, jadi keimanan ini harus dimiliki oleh setiap umat islam. Salah satu jalan untuk bisa mengimani nabi dan rasul adalah dengan mengenali nama-nama mereka dan mempelajari kisah-kisah kehidupan mereka. Selain melalui pendidikan, ada media yang lebih modern untuk mengenalkan nabi dan rasul yaitu dengan *game*. Maka dari itu peneliti tergerak untuk menciptakan sebuah aplikasi *adventure game* tiga dimensi bernama *Finding Diamond*, yang di dalamnya terdapat konten islami tentang nabi dan rasul.

Pada aplikasi *game* yang dibuat ada koin perak, koin emas dan berlian yang harus dikumpulkan *player*. Sehingga pengetahuan yang ditampilkan kelihatan lebih bervariasi. Ada dua karakter NPC di dalam *game* yaitu NPC *Enemy* dan NPC *Army*. NPC *Enemy* perilakunya adalah selalu mengejar *player*. Sedangkan NPC *Army* mengambil semua koin perak di arena permainan dengan mengimplementasikan algoritma *Partial Expansion A**, sehingga pergerakannya lebih efektif, karena NPC *Army* bergerak menuju target sesuai dengan rute terpendek yang dihasilkan oleh algoritma tersebut. Berdasarkan hasil uji coba pada penelitian ini algoritma *Partial Expansion A** membangkitkan lebih sedikit *node* daripada algoritma A* sehingga memori yang digunakan lebih kecil dan waktu untuk menemukan rute terpendek lebih cepat.

ABSTRACT

Adha, Alriandy Putra. 2018**. Know the Prophet and Messengers with 3D Adventure Game using Partial Expansion A Star Method (PEA *).** Thesis. Informatics Engineering Department of Science and Technology Faculty Islamic State University Maulana Malik Ibrahim Malang.

Supervisor: (I) Dr. Muhammad Faisal, M.T, (II) Roro Inda Melani, MT., M.Sc

Keywords: *Prophets and Messengers, A*, Partial Expansion A**

Faith in the prophets and messengers of Allah, is the fourth pillar of faith, so this faith must be owned by all Muslims. One way to be able to believe in the prophets and messengers is recognize their names and learn the stories of their lives. In addition through education, there are more modern media to introduce the prophets and messengers are with the game. Thus the researchers moved to create an application of 3D adventure game called Finding Diamond, in which there is Islamic content of the prophets and messengers.

In gaming applications was created there are silver coins, gold coins and diamonds to be collected player. So that the knowledge displayed more varied look. There are two NPC characters in this game, there is NPC Enemy and NPC Army. NPC Enemy behavior is always pursuing player. While NPC Army took all the silver coins in the game arena by implementing *Partial Expansion* A* algorithm, so that the movement is more effective, because the NPC Army move towards the target in accordance with the route that is generated by the algorithm. Based on trial results on this study *Partial Expansion* A* algorithm generate fewer nodes than A* algorithm so that the memory used is smaller and the time to find the shortest route more quickly.

الملخص

الأضحى، آلرياندي بوترا. 2018. تعرف على السلام والرسول من خلال لعبة المغامرات ثلاثية الأبعاد باستخدام التوسيع الجزئي طريقة النجمة (البازلاء *). أطروحة. قسم المعلوماتية كلية العلوم والتكنولوجيا الجامعة الإسلامية- مولانا عاطفية إبراهيم مالانج. المستشار: (الأول) د. محمد فيصل، إم تي، (الثاني) رورو اندا ميلاني، ماجستير، ماجستير كلمات البحث: السلام ورسول، A *، توسع جزئي أ *مولانا مالك إبراهيم مالانج.

بيمبيمبينج: (١) د. محمد فيصل، M.T، (٢) اندا ميلاني

الكلمات المفتاحية: *الأنبياء والمرسلين، فيشر بيتس المراوغة، *A، الترجيح دينامية *A*

الإيمان الأنبياء والمرسلين من الله، هو الركن الرابع من أركان الإيمان، لذلك يجب أن تكون مملوكة نية من قبل جميع المسلمين .طريقة واحدة لتكون قادرة على الاعتقاد في الأنبياء والمرسلين هو الاعتراف أسمائهم ومعرفة قصص حياتهم .وبالإضافة إلى ذلك من خلال التعليم، وهناك وسائل الاعلام أكثر حداثة لتقديم الأنبياء والرسل هم مع اللعبة .وهكذا انتقل الباحثون إلى إنشاء تطبيق لعبة مغامرة ثلاثية الأبعاد تسمى العثور الماس، والتي لا يوجد محتوى الإسلامي من الأنبياء والرسل.

في الألعاب خلق تطبيقات أي لاعب من الفضة، الذهب والماس التي يتعين جمعها. لكل عملة ذهبية تم اتخاذها، وسوف يعرض معرفة الأنبياء والرسل الذين تم اختيارهم بصورة عشوائية تسلسل باستخدام خوارزمية فيشر بيتس المراوغة. ذلك أن عرض المعرفة نظرة أكثر تنوعا. يمكن هذه الخوارزمية ضمان كل قطعة ذهبية يضم المواد المعرفة المختلفة. ولكن في حالة استخدام ميزة عشوائية في الوحدة لعبة المحرك لا يوجد إمكانية للمادة نفسها المعرفة في المعرض عدة مرات في مباراة واحدة.

هناك نوعان من الشخصيات في اللعبة التي مجلس الشعب مجلس الشعب ومجلس الشعب جيش العدو .السلوك مجلس الشعب العدو دائما تسعى لاعب .بينما تولى الجيش الوطني جميع القطع النقدية الفضية في الساحة لعبة من خلال تنفيذ الحيوي الترجيح خوارزمية *A، بحيث تكون الحركة أكثر فعالية، لأن الجيش الوطني المضي قدما نحو الهدف وفقا للمسار الذي تم إنشاؤه بواسطة خوارزمية .بناء على نتائج المحاكمة في هذه الدراسة الترجيح خوارزمية الحيوي *Aتوليد العقد أقل من *Aخوارزمية ذلك أن الذاكرة المستخدمة هي أصغر والوقت للعثور على أقصر الطرق بسرعة أكبر.

BAB I

PENDAHULUAN

1.1 Latar Belakang

Saat ini banyak para pengembang yang bergerak dibidang perangkat lunak permainan. Baik itu permainan sederhana sampai dengan permainan yang sangat rumit dalam pengerjaannya. Satu dekade terakhir ini perkembangan game semakin rumit dan kompleks. Dengan system yang kompleks seperti itu tentunya sebuah *game* akan menguras memori dan daya yang besar. Tentunya teknologi yang mampu mengatasi pekerjaan kompleks dan berat tanpa mengalami *over-memory* pun sangat dibutuhkan.

Beberapa penelitian terus mencari untuk menemukan dan mengembangkan suatu metode menyesuaikan dengan tuntutan *game* yang semakin kompleks. Salah satu metode yang sering digunakan pada game ialah algoritma A* untuk penyelesaian masalah *pathfinding*. Algoritma ini sangat *simple* dan efisien dan banyak digunakan dalam suatu metode pencarian rute terdekat, atau lebih dikenal sebagai *Path-Finding*. Namun dibeberapa jurnal mengatakan bahwa ada suatu kembangan dari algoritma A* yang mampu mengerjakan tugas lebih baik dan efiesien, mereka menyebutnya *Partial Expansion A**. *Partial Expansion A** (PEA*) adalah varian dari A* yang mampu mengurangi kelebihan memori yang dihasilkan oleh algoritma A*dalam kasus percabangan yang besar.

Dalam penitian ini penulis akan mengimplementasikan algoritma PEA* pada sebuah game 3D yang akan mengusung tema edukasi Nabi dan Rasul menggunakan *platform* Unity 3D. Firman Allah yang menegaskan tentang

kewajiban beriman kepada nabi dan rasul Allah ada beberapa ayat di dalam Al-Qur'an, salah satunya yaitu terdapat pada surat Al-Baqarah ayat 177 yang artinya, *"Bukanlah menghadapkan wajahmu ke arah timur dan barat itu suatu kebajikan, akan tetapi sesungguhnya kebajikan itu ialah beriman kepada Allah, hari kemudian, malaikat-malaikat, kitab-kitab, nabi-nabi dan memberikan harta yang dicintainya kepada kerabatnya, anak-anak yatim, orang-orang miskin, musafir (yang memerlukan pertolongan) dan orang-orang yang meminta-minta, dan (memerdekakan) hamba sahaya, mendirikan shalat, dan menunaikan zakat, dan orang-orang yang menepati janjinya apabila ia berjanji, dan orang-orang yang sabar dalam kesempitan, penderitaan dan dalam peperangan. Mereka itulah orang-orang yang benar (imannya), dan mereka itulah orang-orang yang bertakwa." (QS. Al-Baqarah: 177).*

Pada ayat di atas, Allah memerintahkan kaum mukminin untuk beriman kepada Allah, Rasul-Nya, Al-Qur'an dan kitab suci yang diturunkan sebelumnya. Karena pentingnya iman kepada nabi dan rasul Allah maka keimanan ini harus dimiliki oleh setiap umat islam. Salah satu jalan untuk bisa mengimani adanya nabi dan rasul Allah adalah dengan mengenali nama-nama mereka dan mempelajari kisah-kisah kehidupan mereka. Hal tersebut bisa didapat salah satunya melalui pendidikan, baik yang pendidikan formal maupun nonformal. Selain melalui pendidikan tersebut, ada media yang lebih modern untuk mengenalkan nama-nama nabi dan rasul yaitu dengan *game*. Karena pada dasarnya bermain *game* itu menyenangkan dan hampir setiap orang pernah bermain *game*, maka dari itu peneliti tergerak untuk menciptakan sebuah aplikasi *game* modern yang bertujuan untuk memperkenalkan nabi dan rasul dengan media permainan komputer.

Permainan ini bergenre *adventure game* yang bertujuan untuk semakin mendekatkan anak-anak dan para generasi muda terkait pengetahuan tentang nabi dan rasul dengan cara yang lebih menarik.

2.1 Identifikasi Masalah

Berdasarkan latar belakang diatas identifikasi masalah yang dapat dirumuskan adalah sebagai berikut:

a. Apakah PEA* mampu digunakan untuk *path-finding* pada game *adventure* 3D?
b. Apakah perbedaan antara algoritma PEA* dan algoritma A*?
c. Manakah yang lebih baik antara algoritma PEA* dan algoritma A*?

1.3 Batasan Masalah

Batasan masalah pada penelitian ini adalah sebagai berikut:

a. Konten islami di dalam *game* adalah pengetahuan tentang nabi dan rasul
b. Berbasis *desktop* dan diimplementasikan pada *platform Windows*
c. Dimainkan oleh *single player*
d. *Game* dirancang untuk usia 9-12 tahun

1.4 Tujuan Penelitian

Tujuan dari penelitian ini adalah sebagai berikut:

a. Membuat aplikasi *game* tiga dimensi yang berwawasan islam untuk pengenalan nabi dan rasul yang bisa dimainkan di *personal computer*
b. Mengimplementasikan algoritma PEA* *(Partial Expansion)* untuk pencarian rute terpendek pada NPC (*Non-Player Character*) yang ada di aplikasi *game* tersebut

c. Melakukan uji coba untuk mengetahui perbedaan antara pencarian *rute* terpendek dengan menggunakan algoritma PEA* *(Partial Expansion)* dan algortitma A*

1.5 Manfaat Penelitian

Manfaat pembuatan aplikasi *game* ini adalah dengan disertakannya konten islami di dalam *game* maka bermain game akan lebih bermanfaat dan bisa menambah pengetahuan orang yang bermain *game* terutama yang dibahas dalam penelitian ini yaitu pengetahuan tentang nabi dan rasul. Jadi dengan bermain *game* yang dibuat pada penelitian ini secara tidak langsung orang tersebut telah mempelajari pengetahuan tentang nabi dan rasul. Suasana belajar jadi lebih mudah dan menyenangkan dengan menggunakan *game*.

1.6 Tahapan Penelitian

Berikut adalah tahapan-tahapan yang dilakukan dalam penelitian:

1. Studi literatur

   Studi literatur dilakukan proses pengumpulan dan pengkajian data-data yang diperlukan dalam pembuatan *game*, diantaranya informasi tentang kisah-kisah nabi dan rasul, pembuatan aplikasi *game* dengan *Unity Game Engine*, algoritma A*, *Partial Exansion A**.

2. Perancangan *game*

   Proses ini merupakan proses perancangan, mulai dari perancangan *story board*, karakter, perancangan implementasi algoritma *Partial Expansion A**.

3. Pembuatan *game*

   Proses pembuatan *game* dibangun dengan memanfaatkan *Game Engine Unity 5.2.0f3 64-bit* dengan menggunakan bahasa C#. *Scripting* dilakukan dengan fasilitas *MonoDevelop*.

4. Uji coba dan evaluasi

   Uji coba dalam penelitian ini dilakukan pada *game* setelah karakter NPC *Army* yang sudah diimplementasikan algoritma *Partial Expansion A**.

5. Penyusunan laporan

   Penyusunan laporan akhir merupakan bagian dokumentasi dari keseluruhan pelaksanaan penelitian yang diharapkan dapat bermanfaat bagi penelitian selanjutnya.

BAB II

KAJIAN PUSTAKA

2.1 Landasan Teori

Berikut merupakan penjelasan mengenai teori-teori yang akan digunakan pada game kali ini. Penjelasan meliputi teori metode, penjabaran tentang game, dan materi konten yang akan disajikan dalam game.

2.1.1 Teori *Game*

Game merupakan kata dari bahasa inggris yang artinya adalah permainan. Permainan adalah suatu yang dapat dimainkan dengan aturan tertentu untuk menyenangkan hati (dengan menggunakan alat-alat tertentu atau tidak) sehingga ada yang menang ada yang kalah (Kamus Besar Bahasa Indonesia).

Game atau permainan adalah sesuatu yang dapat dimainkan dengan aturan tertentu sehingga ada yang menang dan ada yang kalah, biasanya dalam konteks tidak serius dengan tujuan *refreshing*. Bermain *game* sudah dapat dikatakan sebagai salah satu gaya hidup masyarakat dimasa kini. Dimulai dari usia anak-anak hingga orang dewasa pun menyukai *video game*. Itu semua dikarenakan bermain *game* adalah hal yang menyenangkan. (Anggara, 2008: 122).

Permainan terdiri atas sekumpulan peraturan yang membangun situasi bersaing dari dua atau lebih orang atau kelompok dengan memilih strategi yang dibangun untuk memaksimalkan kemenangan sendiri ataupun untuk meminimalkan kemenangan lawan. Peraturan-peraturan yang ada menentukan kemungkinan tindakan untuk setiap pemain, sejumlah keterangan diterima setiap pemain sebagai kemajuan bermain, dan sejumlah kemenangan atau kekalahan dalam berbagai situasi. (John von Neumann dan Oskar Morgenstern, 1944: 23).

2.1.2 Genre *Game*

Dalam sebuah permainan, seringkali *game* diklasifikasikan ke dalam sebuah *genre* tertentu. Pengklasifikasian didasarkan pada gaya ataupun kumpulan karakteristik seperti *gameplay*, interaksi, tujuan dan lain lain. Beberapa contoh *genre game* meliputi *adventure games*, *action games*, *FPS (First Person Shooter), RPG (Role Playing Game), MMRPGs (Massively Multiplayer Online Role Playing Games), Puzzle Games, Racing Games, Educational Games* dan lain sebagainya.

Pada *game* kali ini akan menggunakan game bergenre *Adventure*. *Adventure* menggambarkan nuansa petualangan. Berkeliling hutan, melompati bebatuan di antara lahar, bergelayutan dari pohon satu ke pohon lain, melawan naga sambil mencari kunci untuk membuka kuil tersembunyi, atau sekedar mencari petunjuk untuk mendapatkan misi berikutnya. Hal tersebut adalah beberapa hal yang akan dilakukan pemain dalam menikmati *game* bergenre *adventure*. Pengertian dari *Adventure Game* adalah permainan *video* yang mengasumsikan pemain merupakan karakter protagonis dalam sebuah cerita interaktif yang memiliki tujuan untuk mengeksplorasi cerita dan memecahkan *puzzle* (Andrew Rollings dan Ernest Adams, 2006: 37).

2.1.3 *Non-Player Character* (NPC)

NPC atau *Non-Player Character* adalah jenis *otonomous agent* yang ditujukan untuk penggunaan komputer animasi dan media interaktif seperti *games* dan *virtual reality*. Agen ini mewakili tokoh dalam cerita atau permainan dan memiliki kemampuan untuk improvisasi tindakan mereka. Ini adalah kebalikan dari seorang tokoh dalam sebuah film animasi, yang tindakannya ditulis di muka, dan untuk "*avatar*" dalam sebuah permainan atau *virtual reality*, tindakan yang

diarahkan secara *real time* oleh pemain. Dalam permainan, karakter *otonom* biasanya disebut *Non-Player Character* (NPC).

2.1.4 Finite State Machine

Finite State Machines (FSM) masuk dalam ranah *Decision Making* (pembuat keputusan) pada *Artificial Intelligence*. Dalam FSM masing-masing karakter menempati satu *state*. Biasanya, tindakan atau perilaku yang terkait dengan masing-masing *state*. Jadi selama karakter tetap dalam keadaan itu, ia akan terus melakukan tindakan yang sama. *State* terhubung bersama oleh *transition*. Setiap *transition* mengarah dari satu *state* ke *state* lain yang biasanya *state* tujuan *state* target ini disebut dengan *action* dan masing-masing memiliki seperangkat kondisi yang terkait. Jika permainan menentukan bahwa kondisi transition terpenuhi, maka karakter berubah dari *state* ke *state* target (*action*) melalui *transition* itu. (Ian Millington, 2006: 129).

FSM melacak himpunan *state* yang ada kemudian inputan masuk ke masing-masing *state*, serangkaian keadaan *transition* tetap. Setiap *transition* dapat diimplementasikan dengan kondisi yang sesuai. Pada setiap iterasi (biasanya setiap *frame*), fungsi update FSM digunakan. Ini memeriksa untuk melihat apakah ada perubahan *transition* dari kondisi saat dipicu oleh inputan. Kemudian menyusun daftar *action* dari kondisi yang sedang aktif. Jika *transition* telah menemukan *action* yang dituju, maka *transition* berhenti.

2.1.5 Nabi dan Rasul

Nabi dan rasul adalah hamba-hamba Allah pilihan yang menerima wahyu dan risalah dari Allah SWT. Nabi adalah hamba Allah pilihan yang menerima wahyu untuk dirinya sendiri dan tidak mempunyai kewajiban untuk menyampaikannya

kepada umat manusia. Rasul adalah manusia pilihan yang menerima wahyu dan risalah dari Allah dan bertanggung jawab menyampaikannya kepada umat manusia. Setiap rasul adalah nabi, sedangkan setiap nabi belum tentu rasul. (Hanafi, 2011).

Sebagian ulama menyatakan bahwa definisi ini memiliki kelemahan, karena tidaklah wahyu disampaikan Allah ke bumi kecuali untuk disampaikan, dan jika nabi tidak menyampaikan maka termasuk menyembunyikan wahyu Allah. Kelemahan lain dari definisi ini ditunjukkan dalam hadits dari nabi *shallallahu 'alaihi wa sallam*, "*Ditampakkan kepadaku umat-umat, aku melihat seorang nabi dengan sekelompok orang banyak, dan nabi bersama satu dua orang dan nabi tidak bersama seorang pun." (HR. Bukhori dan Muslim).*

Hadits ini menunjukkan bahwa nabi juga menyampaikan wahyu kepada umatnya. Ulama lain menyatakan bahwa ketika nabi tidak diperintahkan untuk menyampaikan wahyu bukan berarti nabi tidak boleh menyampaikan wahyu. *Wallahu'alam*. Perbedaan yang lebih jelas antara nabi dan rasul adalah seorang rasul mendapatkan syari'at baru sedangkan nabi diutus untuk mempertahankan syari'at yang sebelumnya.

Jumlah nabi dan rasul sangat banyak. Namun yang tersebut dalam Al-Qur'an dan wajib kita imani berjumlah 25 orang. Mereka adalah Adam, Idris, Nuh, Hud, Shaleh, Ibrahim, Luth, Ismail, Ishaq, Ya'qub, Yusuf, Syuaib, Ayub, Zulkifli, Musa, Harun, Daud, Sulaiman, Ilyas, Ilyasa', Yunus, Zakariya, Yahya, Isa dan Muhammad SAW.

Banyak nabi dan rasul lainnya yang tidak dikisahkan dalam Al-Qur'an. Allah berfirman yang artinya, *"Dan (kami telah mengutus) rasul-rasul yang sungguh telah kami kisahkan tentang mereka kepadamu dahulu, dan rasul-rasul yang tidak*

Kami kisahkan tentang mereka kepadamu." (QS. An-Nisa': 164). Walaupun dalam Al-Qur'an hanya disebut 25 nabi, maka kita tetap mengimani secara global adanya nabi dan rasul yang tidak dikisahkan dalam Al-Qur'an. Allah berfirman yang artinya, "*Dan sesungguhnya telah Kami utus beberapa orang rasul sebelum kamu, di antara mereka ada yang Kami ceritakan kepadamu dan di antara mereka ada yang tidak Kami ceritakan kepadamu." (QS. Al-Mu'min: 78).*

Setiap umat di dunia ini memiliki rasul. Allah mengutus setiap rasul-Nya untuk tiap umat sepanjang masa secara terus menerus. Tidak ada satu umat pun yang tidak punya rasul. Hal ini supaya setiap umat di muka bumi ini tetap beriman dan berbakti kepada Allah serta menghindarkan kerusakan yang dilakukan oleh umat tertentu. Karena rasul diutus dengan tujuan mengingatkan mereka yang lalai dan memberi kabar baik bagi yang ingat. Allah berfirman yang artinya, *"Demi Allah, sesungguhnya Kami telah mengutus rasul-rasul Kami kepada umat-umat sebelum kamu." (QS. An-Nahl: 63).* Dan di ayat lain Allah berfirman yang artinya, *"Dan tidak ada suatu umatpun melainkan telah ada padanya seorang pemberi peringatan." (QS. Fathir: 24).*

Konten islami yang akan dimasukkan di dalam *game* adalah pengenalan nama-nama nabi dan rasul beserta mukjizatnya yang akan ditampilkan setiap kali *player* mendapatkan satu koin emas yang ada di arena permainan. Di bawah ini rincian nama-nama nabi dan rasul beserta mukjizatnya atau beberapa cuplikan kisahnya yang akan disertakan di dalam aplikasi *game.*

2.1.6 Algoritma A*

Algoritma A* merupakan salah satu algoritma yang dimanfaatkan sebagai metode untuk menemukan *rute* (*pathfinding*). *Pathfinding* digunakan untuk

menentukan arah pergerakan suatu objek dari satu tempat ke tempat lain berdasarkan keadaan peta dan objek lainnya. Algoritma yang digunakan untuk *pathfinding* sudah banyak yang ditemukan, misalnya *Bellman–Ford, Dijkstra, Floyd–Warshall, A Star* (A*), *Dynamic Weighting A** (PEA*) dan lain-lain.

Algoritma yang akan diimplementasikan pada *game* di penelitian ini adalah algoritma PEA*. Tetapi karena algoritma tersebut merupakan pengembangan dari algoritma A* maka terlebih dahulu dibahas tentang A*. Algoritma A* merupakan algoritma *Best First Search* yang menggabungkan *Uniform Cost Seach* dan *Gredy Best-First Seach*. Biaya yang diperhitungkan didapat dari biaya sebenarnya ditambah dengan biaya perkiraan. Dalam notasi matematika dituliskan sebagai berikut:

$f(n) = g(n) + h(n)$ .....(1)

dengan:

$f(n)$ = fungsi evaluasi

$g(n)$ = biaya sebenarnya / jarak dari *start node* (implementasi di *game*)

$h(n)$ = biaya perkiraan / jarak dari *end node* (implementasi di *game*)

Dengan perhitungan biaya seperti ini, algoritma A* adalah *complete* (selalu menemukan solusi jika solusinya ada) dan *optimal* (menemukan rute terpendek). (Suyanto, 2011: 31).

A* adalah algoritma pencarian *graph*/pohon yang mencari jalur dari satu titik awal ke sebuah titik akhir yang telah ditentukan. Algoritma A* menggunakan pendekatan heuristik *h(n)* yang memberikan peringkat ke tiap-tiap titik *n* dengan cara memperkirakan rute terbaik yang dapat dilalui dari titik tersebut. Setelah itu tiap-tiap titk *n* tersebut dicek satu-persatu berdasarkan urutan yang dibuat dengan

pendekatan heuristik tersebut. Maka dari itulah algoritma A* adalah contoh dari *Best-First Search*. Algoritma ini pertama kali ditemukan pada tahun 1968 oleh *Peter Hart, Nils Nilsson* dan *Bertram Raphael*. Dalam tulisan mereka, algoritma ini dinamakan algoritma A. Penggunaan algoritma ini dengan fungsi heuristik yang tepat dapat memberikan hasil yang optimal, maka algoritma ini pun disebut A*. Beberapa terminologi dasar yang terdapat pada algoritma ini ketika diimplementasisan pada aplikasi *game* antara lain: *startnode*, *endnode*, simpul (*node*), *currentnode*, *openlist*, *closedlist*, harga/biaya (*cost*), halangan (*unwalkable*).

- *Startnode* adalah sebuah terminologi posisi awal sebuah benda.
- *Currentnode* adalah simpul yang sedang dijalankan algortima pencarian jalan terpendek.
- Simpul atau *node* adalah petak-petak kecil sebagai representasi dari area *pathfinding*. Bentuknya dapat berupa persegi, lingkaran, maupun segitiga.
- *Openlist* adalah tempat menyimpan data simpul yang mungkin diakses dari *startnode* maupun simpul yang sedang dijalankan.
- *Closedlist* adalah tempat menyimpan data simpul sebelum *currentnode* yang juga merupakan bagian dari jalur terpendek yang telah didapatkan.
- Harga (F) adalah nilai yang diperoleh dari penjumlahan nilai G, jumlah nilai tiap simpul dalam jalur terpendek dari *startnode* ke *currentnode*, dan H, jumlah nilai perkiraan dari sebuah simpul ke simpul tujuan.
- *Endnode* atau simpul tujuan yaitu simpul yang dituju.

- Rintangan (*unwalkable*) adalah sebuah atribut yang menyatakan bahwa sebuah simpul tidak dapat dilalui oleh *currentnode*.

Prinsip algoritma ini adalah mencari jalur terpendek dari sebuah simpul awal (*startnode*) menuju simpul tujuan dengan memperhatikan harga (F) terkecil. A* memperhitungkan *cost* dari *currentnode* ke tujuan dengan fungsi *heuristic*. Algoritma ini juga mempertimbangkan *cost* yang telah ditempuh selama ini dari *initial node* ke *currentnode*. Jadi jika ada jalan yang telah ditempuh sudah terlalu panjang dan ada jalan lain yang *cost*-nya lebih kecil tetapi memberikan posisi yang sama dilihat dari *goal*, jalan yang lebih pendek yang akan dipilih. Algoritma A* pada aplikasi *game* dijelaskan dengan *pseudocode* dibawah ini:

1. Masukan *startnode* ke *openlist*
2. *Loop* langkah-langkah di bawah ini :
    a. Cari *node(n)* dengan nilai *f(n)* yang paling rendah dalam *openlist*, *node* ini sekarang menjadi *currentnode*.
    b. Keluarkan *currentnode* dari *openlist* dan masukan ke *closedlist.*
    c. Untuk setiap tetangga dari *currentnode* lakukan berikut :
        • Jika tidak dapat dilalui atau sudah ada dalam *closelist*, maka abaikan.
        • Jika belum ada di *openlist* . Buat *currentnode parent* dari *node* tetangga ini. Simpan nilai *f,g* dan *h* dari *node* ini.
        • Jika sudah ada di *openlist*, cek bila *node* tetangga ini lebih baik, menggunakan nilai *g* sebagai ukuran. Jika lebih baik ganti *parent* dari *node* ini di *openlist* menjadi *currentnode*, lalu kalkulasi ulang nilai *g* dan *f* dari *node* ini.

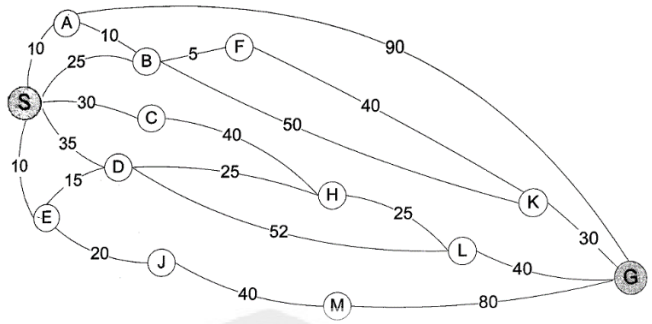
d. Hentikan *loop* jika :

• *Node* tujuan telah ditambahkan ke *openlist*, yang berarti rute telah ditemukan.

• Belum menemukan *node goal*/*endnode* sementara *openlist* kosong atau berarti tidak ada rute.

3. Simpan rute, secara *backward*, urut mulai dari *node goal* ke *parent*-nya terus sampai mencapai *node* awal sambil menyimpan *node* ke dalam array.

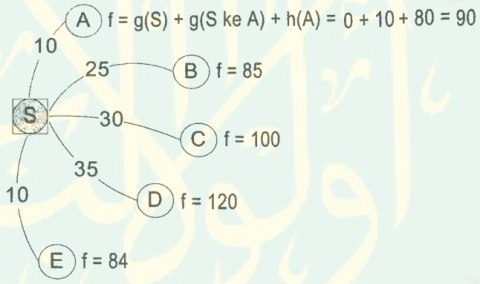
Untuk lebih jelasnya, perhatikan sebuah masalah pencarian rute terpendek dari kota S (*initial state*) ke kota G (*goal state*) pada Gambar 2.1 dibawah ini. Terdapat 13 kota yang dinyatakan oleh simpul-simpul dalam suatu *graph* dua arah. Setiap angka pada busur menyatakan biaya sebenarnya antara satu kota dengan kota lainnya. Misalnya biaya disini adalah jarak antar kota dalam satuan kilometer. Nilai *h(n)* adalah fungsi heuristik, yaitu jarak garis lurus dari simpul *n* menuju simpul G dalam satuan kilometer. Pada langkah-langkah penyelesaian masalah rute terpendek dengan algoritma A* kali ini, ada sedikit perbedaan dengan *pseudocode* diatas. Simpul-simpul yang sudah ada di *closed* tidak diabaikan, tetapi masih harus di cek lagi apakah perlu atau tidak untuk mengganti *parent*-nya, karena biaya sebenarnya atau *g(n)* pada masalah ini nilainya berbeda-beda disetiap jarak antar simpulnya. Sedangkan ketika diimplementasikan pada *game* di *Unity* (dibahas pada bab 3), *g(n)* memiliki nilai yang sama yaitu selalu +10 untuk gerakan horizontal atau vertikal dan +14 untuk gerakan diagonal.



n	S	A	B	C	D	E	F	G	H	J	K	L	M
h(n)	80	80	60	70	85	74	70	0	40	100	30	20	70

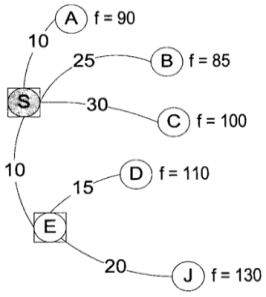
Gambar 2.1 Masalah Pencarian Rute Terpendek Pada Suatu Graf

Untuk memperjelas pemahaman tentang algoritma A*, di bawah ini akan dijelaskan penyelesaian masalah diatas menggaunakan algoritma A*. Langkah-langkah pencarian rute berdasarkan algoritma tersebut adalah sebagai berikut:



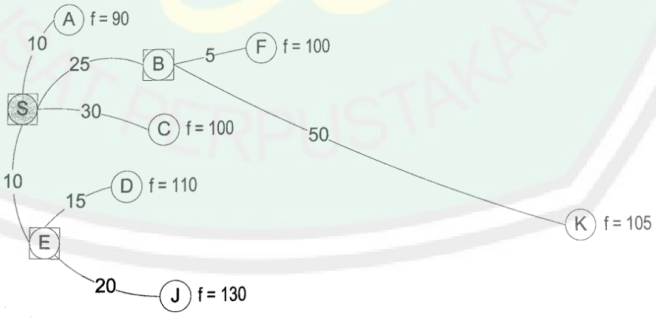
Gambar 2.2 Pencarian Rute Terpendek dengan Algoritm A* Langkah 1

Langkah pertama, karena di *open* hanya terdapat satu simpul (yaitu S), maka S terpilih sebagai *bestnode* dan dipindahkan ke *closed*. Kemudian dibangkitkan semua suksesor S, yaitu: A, B, C, D dan E. Karena kelima suksesor tidak ada di *open* maupun *closed*, maka kelimanya dimasukkan ke *open*. Langkah pertama ini menghasilkan *open* = [A,B,C,D,E] dan *closed* = [S].



Gambar 2.3 Pencarian Rute Terpendek dengan Algoritm A* Langkah 2

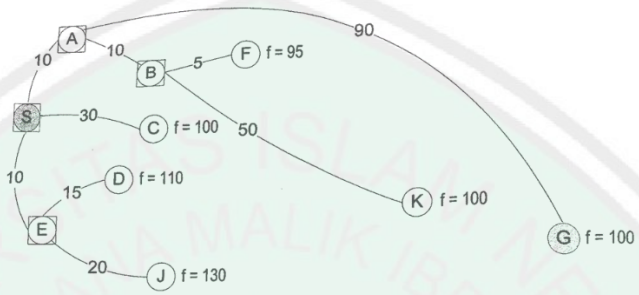
Langkah kedua, E dengan biaya terkecil (yaitu 84) terpilih sebagai *bestnode* dan dipindahkan ke *closed*. Selanjutnya, semua suksesor E dibangkitkan, yaitu: D dan J. Karena belum pernah ada di *open* maupun *closed*, maka J dimasukkan ke *open*. Sedangkan simpul D sudah ada di *open*, maka harus dicek, apakah *parent* dari D perlu diganti atau tidak. Ternyata biaya dari S ke D melalui E (yaitu 10+15 = 25) lebih kecil daripada biaya S ke D (yaitu 35). Oleh karena itu, *parent* D harus diubah, yang semula S menjadi E. Dengan perubahan *parent* ini maka nilai *g* dan *f* pada D juga diperbarui (nilai *g* yang semula 35 menjadi 25, dan nilai *f* dari 120 menjadi 110). Akhir dari langkah kedua ini menghasilkan *open* = [A,B,C,D,J] dan *closed* = [S,E].



Gambar 2.4 Pencarian Rute Terpendek dengan Algoritm A* Langkah 3

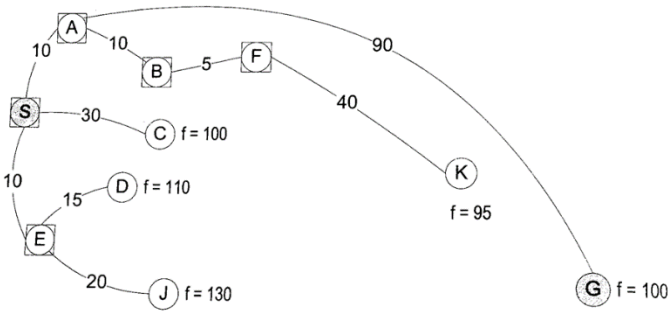
Langkah ketiga, B dengan biaya terkecil (yaitu 85) terpilih sebagai *bestnode* dan dipindahkan ke *closed*. Selanjutnya, semua suksesor B dibangkitkan, yaitu: A, F dan K. Karena belum pernah ada di *open* maupun *closed*, maka F dan K

dimasukkan ke *open*. Sedangkan simpul A sudah ada di *open*, maka harus dicek, apakah *parent* dari A perlu diganti atau tidak. Ternyata biaya dari S ke A melalui B (yaitu 25+10 = 35) lebih besar daripada biaya S ke A (yaitu 10). Oleh karena itu, *parent* dari A tidak perlu diubah (tetap S). Akhir dari langkah ketiga ini menghasilkan *open* = [A,C,D,F,J,K] dan *closed* = [S,E,B].



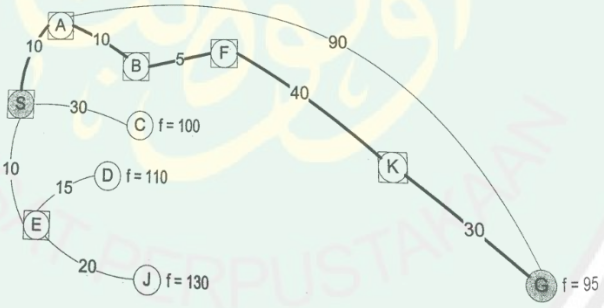
Gambar 2.5 Pencarian Rute Terpendek dengan Algoritm A* Langkah 4

Langkah keempat, A dengan biaya terkecil (yaitu 90) terpilih sebagai *bestnode* dan dipindahkan ke *closed*. Selanjutnya, semua suksesor A dibangkitkan, yaitu: B dan G. Karena belum pernah ada di *open* maupun *closed*, maka G dimasukkan ke *open*. Sedangkan simpul B sudah ada di *closed*, maka harus dicek, apakah *parent* dari B perlu diganti atau tidak. Ternyata biaya dari S ke B melalui A (yaitu 10+10 = 20) lebih kecil daripada biaya S ke B (yaitu 25). Oleh karena itu, *parent* dari B harus diubah, yang semula S menjadi A. Dengan perubahan *parent* ini maka nilai *g* dan *f* pada B juga diperbarui (nilai *g* yang semula 25 menjadi 20, dan nilai *f* dari 85 menjadi 80). Nilai *g* dan *f* pada suksesor-suksesor B juga harus diperbarui menngunakan penelusuran *Dept First Search* (DFS). Dalam kasus ini, B hanya mempunyai dua anak, yaitu F dan K. Nilai *g*(F) yang semula 30 diubah menjadi 25, dan nilai *f*(F) dari 100 menjasi 95. Nilai *g*(K) yang semula 75 diubah menjadi 70, dan nilai *f*(K) dari 105 menjadi 100. Akhir langkah keempat ini menghasilkan *open* = [C,D,F,G,J,K] dan *closed* = [S,E,B,A].



Gambar 2.6 Pencarian Rute Terpendek dengan Algoritm A* Langkah 5

Langkah kelima, F dengan biaya terkecil (yaitu 95) terpilih sebagai *bestnode* dan dipindahkan ke *closed*. Selanjutnya, semua suksesor F dibangkitkan, yaitu: K. Karena K sudah ada di *open*, maka harus dicek, apakah *parent* dari K perlu diganti atau tidak. Ternyata biaya dari S ke K melalui F lebih kecil daripada biaya S ke K melalui parent lama (B). Oleh karena itu, *parent* dari K harus diubah, yang semula B menjadi F. Selanjutnya, nilai *g*(K) yang semula 70 diubah menjadi 65, dan nilai *f*(K) dari 100 menjadi 95. Akhir dari langkah kelima ini menghasilkan *open* = [C,D,G,J,K] dan *closed* = [S,E,B,A,F].



Gambar 2.7 Pencarian Rute Terpendek dengan Algoritm A* Langkah 6

Langkah keenam, K dengan biaya terkecil (yaitu 95) terpilih sebagai *bestnode* dan dipindahkan ke *closed*. Selanjutnya, semua suksesor K dibangkitkan, yaitu: G. Karena G sudah ada di *open*, maka harus dicek, apakah *parent* dari G perlu diganti atau tidak. Ternyata biaya dari S ke G melalui K lebih kecil daripada biaya S ke G melalui *parent* lama (A). Oleh karena itu, *parent* dari G harus diubah, yang

semula A menjadi K. Selanjutnya, nilai *g*(G) yang semula 100 diubah menjadi 95, dan nilai *f*(G) dari 100 menjadi 95. Pada akhir langkah keenam ini menghasilkan *open* = [C,D,G,J] dan *closed* = [S,E,B,A,F,K].

Selanjutnya, G dengan biaya terkecil (yaitu 95) terpilih sebagai *bestnode*. Karena *bestnode* sama dengan *goal*, berarti solusi sudah ditemukan. Rute dan total biaya bisa ditelusuri balik dari G menuju S karena setiap simpul hanya memiliki satu *parent*. Penelusuran balik menghasilkan rute S-A-B-F-K-G dengan total jarak sama dengan 95 kilometer. Rute ini merupakan rute terpendek yang ada di *graph* tersebut. Jadi algoritma A* adalah *optimal* (menemukan rute terpendek) dan *complete* (selalu menemukan solusi jika solusinya ada) (Suyanto, 2011).

2.1.7 Algoritma Partial Expansion A* (PEA*)

Algoritma *Partial Expansion A*(*PEA*) merupakan kembangan dari algoritma A*. Algoritma PEA* mampu mengurangi memori yang dihasilkan oleh A* dalam kasus percabangan yang besar. Ketika menyelesaikan permasalahan dengan percabangan yang besar, A* memungkinkan untuk menghasilkan banyak node yang menghasilkan biaya yang melebihi solusi optimal. Konsep dari algoritma PEA* adalah hanya mengkspansi sebagian dari node tetangga.

Berbeda dengan A*, PEA* tidak memasukkan semua *child node* ke dalam *Open List*. *Child* yang akan dimasukkan dalam *Openlist* harus diseleksi dengan cara memangkas nilai *f-Child* yang dirasa melebihi biaya optimal. Proses seleksi yang dilakukan menggunakan nilai *C (Cutoff)*, dari hasil seleksi tersebut akan dipilih child dengan nilai f(n) terkecil, sehingga menghasilkan *Promising Child*. Dalam notasi matematika dituliskan sebagai:

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

dimana, *Promising Child* adalah

$f(n) <= C+F(f)$

dengan:

f(n) = nilai *child(neighbour)*
g(n) = biaya sebenarnya / jarak dari *start node*
h(n) = biaya perkiraan / jarak dari *end node*
F(n)= nilai *node Parent*
C = variable pemotong dengan nilai non-negatif *(Cutoff)*
Promising Child = Node child yang disimpan ke *Open List*

Prinsip algoritma PEA* sama dengan A* yaitu mencari jalur terpendek dari sebuah simpul awal (*startnode*) menuju simpul tujuan dengan memperhatikan harga (F) terkecil, hanya saja PEA* menggunakan nilai C untuk menyeleksi node-node tidak menjanjikan yang bertujuan menghemat penggunaan memori dan meringankan kinerja dari komputer. Semakin kecil nilai C maka semakin sedikit node yang tersimpan. Ketika nilai C=0 maka ini adalah kondisi jalur terbaik, namun ini hampir tidak memungkinkan karena akan terjadi kekosongan pada Openlist sebelum menemukan *End node*. Pada kondisi C=∞ yang terjadi adalah algoritma kita akan teridentifikasi sebagai A*, bukan lagi PEA*.

Algoritma PEA* pada aplikasi *game* dijelaskan dengan *pseudocode* dibawah ini:

a) Deklarasi variabel : *openList*, *startNode*, *endNode*, fungsi heuristic, *node*, *neighbors* (semua *node* yang tepat bersebelahan dengan *node* saat ini), *child* (satu *node* yang diambil dari *neighbors*)
b) *Push startNode* ke *openList* dan ubah *variable opened* menjadi *true.*

c) *Pop node* yang ada dalam *openList* untuk dijelajahi.

d) Ubah nilai *variable closed* menjadi *true* sebagai tanda bahwa *node* sudah ditelusuri.

e) Cek apakah *node* sekarang adalah *endNode*, jika benar maka *backtrace* untuk membuat *path* hasil *pathfinding* dengan menelusuri *parent* dari *node* saat ini dan hentikan perulangan dengan melakukan *return* ***path*** hasil *pathfinding*.

f) Jika *node* bukan *endNode*, lanjutkan pencarian dengan mengambil *neighbors* dengan fungsi *grid.getNeighbors*.

g) Jika *neighbor* sudah pernah ditelusuri maka tidak perlu melakukan pemeriksaan ulang.

h) Jika *neighbor* belum pernah ditelusuri, maka lanjutkan pemeriksaan.

i) Apabila *neighbor* belum pernah dimasukkan ke dalam *openList* maka hitung nilai f dari *neighbor* dengan nilai penjumlahan nilai g dan h neighbor saat ini, ubah *parent* dari *neighbor* menjadi *node* saat ini.

j) Apabila *node* belum pernah masuk ke dalam *list*, lakukan pengecekan terhadap nilai f dari *neighbor* apakah lebih kecil atau sama dengan f *parent*. Jika ya, maka masukkan *node* ke dalam *list* dan tandai bahwa sudah penah masuk ke dalam *list*. Jika tidak, abaikan atau *discard neighbors*, kemudian *update* nilai f dari *node parent* menjadi nilai f terkecil dari *node neighbor* yang tidak masuk ke dalam *openList*.

k) Jika *node* sudah pernah masuk dalam *list*, maka cukup *update* nilai dari *neighbor* yang ada didalam *list*.

2.2 Penelitian Terkait

2.2.1 A* with Partial Expansion for Large Branhing Factor Problem (Yosizumi, Miura, Ishida, 2000)

Jurnal hasil penelitian ini menjeskan bagaimana menyelesaikan permasalahan pada metode algoritma A*, yang mana memiliki faktor permasalahan pada percabangan yang besar. Pada kasus percabangan yang besar memuat sejumlah deretan *array* yang besar pula, yang mana mengonsumsi lebih banyak memori. Solusi yang ditawarkan untuk permasalahan ini adalah membuang *node-node* anak yang tidak memungkinkan untuk memberikan solusi optimal. Sebuah komponen pun ditambahkan untuk menyeleksi *childnode* yang tidak menjanjikan, mereka menggunakan variabel C *(Cutoff)*. Variable ini harus memiliki nilai non-negatif. *Child* yang menjanjikan adalah *child* yang memiliki nilai f yang ebih kecil dari nilai F *parent* yang di jumlahkan dengan nilai C. Dengan hanya menyimpan *child* yang menjanjikan akan menghemat memori dan juga tingkat kinerja komputer lebih ringan.

2.2.2 Perbandingan Penerapan Algoritma A*, IDA*, JumpPoint Search, dan PEA* pada Permainan Pacman (Rosa, dkk, 2016)

Perbandingan yang dilakukan guna untuk mengetahui algoritma yang paling efisien dalam proses *pathfinding*. Ada beberapa aspek yang menjadi penilaian pada penelitian ini, yaitu diukur dari jumlah *visited node*, panjang *path*, dan jumlah *node* dalam *openlist.* Berdasarkan hasil implementasi dan analisis system, pada permainan *costumed* pacman dengan ukuran *grid map* 19x22 pixel beberapa kesimpulan, yaitu (1) Algoritma *Jump Point Search* mampu menghasikan rata-rata jumlah *visited node* terbaik dari keempat algoritma. *Grid map* pada game pacman mendukung performa fungsi *jump* pada JPS; (2) Algoritma PEA* kurang sesuai

untuk diimplementasikan dalam permainan pacman karena perbedaan nilai *heuristic* setiap *node* yang tidak terlalu signifikan dan sedikitnya jumlah percabangan dalam permainan; (3) keempat algoritma memiliki optimalitas menemukan solusi yang sama. Hal ini diukur melalui panjang *path*.

BAB III

PERANCANGAN GAME

3.1 Perancangan *Game*

Berikut merupakan design game yang akan dibangun. Meliputi deskripsi game, *Story Board Game*, deskripsi karakter, deskripsi item, FSM, dan simulasi perhitungan PEA* dan A*.

3.1.1 Keterangan Umum *Game*

Game ini adalah *game* yang bergenre *adventure game* dengan konten islami tentang nabi dan rasul yang dimainkan secara *single player*. Pada *game* ini terdapat karakter *player* sebagai pemain utama yang akan dijalankan oleh pengguna. Dan ada dua karakter NPC yaitu NPC *Army* dan NPC *Enemy* yang merupakan karakter lawan akan dijalankan secara otomatis oleh komputer sesuai dengan yang telah di programkan. *Game* ini lebih bersifat bermain sambil belajar dan dapat memberikan pembelajaran untuk mengenal nama-nama nabi dan rasul beserta mukjizatnya.

Game ini mengisahkan perjalanan *player* dalam menyelesaikan misi untuk meningkatkan pengetahuannya dalam mengenal nama-nama nabi dan rasul beserta mukjizatnya. Misi utama *player* adalah untuk mendapatkan berlian yang ada di arena permainan. Tetapi ketika permainan dimulai berlian tersebut masih berada di dalam tembok pelindung yang tidak bisa dilewati oleh *player*. Supaya tembok tersebut runtuh dan *player* bisa mengambil berlian tersebut, terlebih dahulu *player* harus mengumpulkan koin emas dan koin perak yang ada di arena permainan. Setiap koin yang didapat akan menambah poin *player*. Jika poin *player* sudah mencukupi sesuai dengan level yang dimainkan, maka tembok pelindung berlian akan runtuh dan *player* bisa mengambil berlian tersebut.

Selama perajalan mengumpulkan koin emas, koin perak, dan berlian ada dua karakter NPC yang mengganggu perjalanan *player*. NPC *Enemy* akan bergerak mengikuti kemanapun *player* pergi. *Player* harus berlari untuk menghindari NPC *Enemy*, jika *player* terkena NPC *Enemy* maka kesehatan *player* berkurang satu poin dan NPC *Enemy* mati, tetapi masih ada NPC *Enemy* lain yang harus dihindari oleh *player*. Jumlah dari NPC *Enemy* yang mengejar *player* berbeda-beda sesuai level yang sedang dimainkan oleh *player*.

Karakter NPC yang lain yaitu NPC *Army* merupakan satu pasukan NPC yang akan mencuri atau mengambil semua koin perak yang ada di arena permainan. Sehingga *player* harus dengan segera jika ingin mengambil koin perak untuk menambah jumlah poinnya, supaya tidak terlebih dahulu dihabiskan oleh pasukan NPC *Army*. Pergerakan dari NPC *Army* ini mengimplementasikan algoritma *Partial Expansion A* (PEA*).* Ketika permainan dimulai setiap NPC *Army* akan mencari rute terdekat menuju satu koin perak yang menjadi tujuannya. Setelah rute terpendek berhasil ditemukan, NPC *Army* akan bejalan menuju koin sesuai dengan rute terpendek yang telah ditemukan. Setelah sampai tujuannya, NPC *Army* langsung mengambil koin tersebut dan tugasnya selesai. Untuk selanjutnya NPC *Army* tetap diam di tempat dia mengambil koin, yang juga menjadi tanda untuk *player* bahwa di tempat tersebut pernah ada koin tetapi telah lebih dulu diambih oleh NPC *Army*.

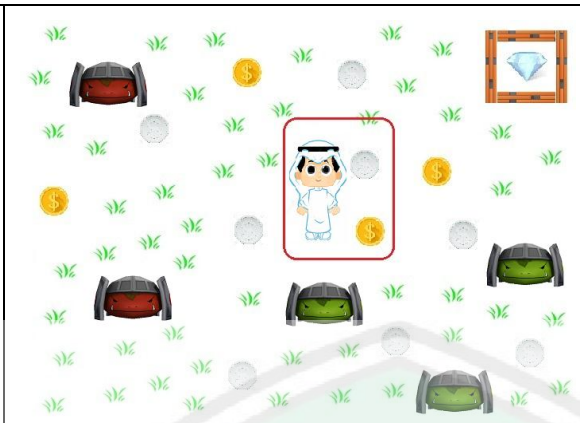
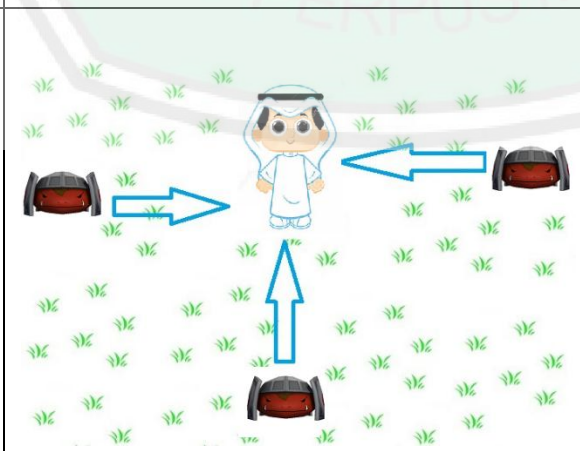
Satu koin perak yang didapat *player* akan mendapatkan lima poin, sedangkan untuk koin emas sepuluh poin. Tetapi untuk setiap satu koin emas yang didapat, *player* akan mendapatkan satu pengetahuan tentang nabi dan rasul.

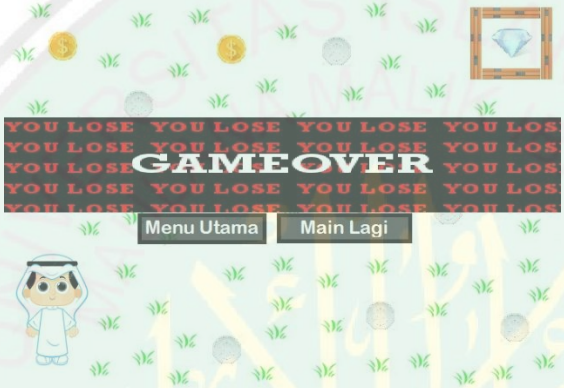
Jika poin *player* dari hasil mengumpulkan koin emas dan koin perak telah mencukupi, maka tembok pelindung berlian akan runtuh, sehingga *player* bisa mengambil berlian yang ada di dalamnya, berarti *player* telah berhasil menyelesaikan misi pada permainan ini.

3.1.2 Story Board Game

Tabel 3.1 *Story Board Game*

No	Gambar	Keterangan
1		Kondisi awal ketika *game* dimulai, *player* siap untuk bermain, selain *player* di arena permainan terdapat koin emas, koin perak, berlian, NPC *Army* dan NPC *Enemy.*
2		Misi *player* adalah untuk menemukan sebuah berlian di arena permainan, tetapi berlian tersebut terhalang oleh sebuah tembok pelindung yang tidak bisa dilewati oleh *player.*

3		*Player* harus mengum pulkan koin emas dan koin perak dengan jumlah tertentu sesuai dengan level yang dimainkan untuk membuat tembok pelindung berlian hilang dan bisa mengambil berliannya.
4		Disaat *player* berhasil mendapatkan satu koin emas maka *player* mendapat 10 poin dan mendapatkan satu penge tahuan tentang nabi dan rasul,
5		Disaat *player* berhasil mendapatkan satu koin perak maka *player* mendapatkan 5 poin.
6		Selama perajalan meng umpulkan koin emas, koin perak, dan berlian ada NPC *Enemy* yang bergerak mengikuti kemana pun *player* pergi. *Player* harus berlari menghindari NPC *Enemy,* jika *player* tertabrak oleh NPC *Enemy* maka kesehatan *player* ber kurang satu poin dan NPC *Enemy* mati, tetapi masih ada NPC *Enemy* lain yang harus dihindari oleh *player.*

7		Karakter NPC yang lain yaitu NPC *Army* meru pakan satu pasukan NPC yang akan mencuri atau mengambil semua koin perak yang ada di arena permainan. Jadi *player* harus dengan segera jika ingin mengambil koin perak untuk mena mbah poinnya sebelum dihabiskan NPC *Army*. Pergerakan NPC *Army* ini merupakan imple mentasi dari algoritma *Partial Expansion A**.
8		Jika waktu yang dimiliki *player* habis atau kesehatan *player* habis, maka *player* dinyatakan kalah atau *game over.*
9		Misalkan untuk level satu poin yang harus dikumpukan *player* adalah 100 poin, jika *player* berhasil mendapatkannya maka tembok pelindung berlian akan runtuh dan *player* bisa untuk segera mengambil berlian tersebut.
10		Ketika *player* telah ber hasil mengambil berlian, berarti *player* telah ber hasil menyelesaikan misi pada permainan ini. *Player* dinyatakan men ang dan bisa bermain di level selanjutnya.

3.1.3 Deskrispi Karakter

Pada permainan ini terdapat satu karakter utama atau *player* yang pergerakannya sesuai perintah orang yang bermian *game* dan dua karakter NPC yaitu NPC *Enemy* dan NPC *Army*.

a. Karakter Utama

Karakter utama atau *player* adalah karakter berpakaian baju muslin bernama Adam. Merupakan karakter di dalam *game* yang dimainkan oleh *user*, pergerakannya sesuai dengan perintah *user*.

Gambar 3.1 Karakter Utama

b. Karakter NPC *Enemy*

Karakter NPC *Enemy* adalah karakter pengganggu yang terus mengejar kemanapun *player* pergi. Jika berhasil mengenai *player* maka kesehatan *player* berkurang satu poin dan NPC *Enemy* akan mati.

Gambar 3.2 Karakter NPC *Enemy*

c. Karakter NPC *Arny*

Karakter NPC *Army* adalah satu pasukan NPC yang akan mencuri atau mengambil semua koin perak yang ada di arena permainan. Sehingga *player* harus dengan segera jika ingin mengambil koin perak untuk menambah jumlah poinnya, supaya tidak terlebih dahulu dihabiskan oleh pasukan NPC *Army*. Pergerakan dari NPC *Army* ini mengimplementasikan algoritma *Partial Expansion A* (PEA*)* untuk mencari rute terdekat ke koin perak yang menjadi tujuan dari setiap NPC *Army* dan kemuadian bergerak untuk mengambilnya sesuai dengan rute yang telah ditemukan.

Gambar 3.3 Karakter NPC *Army*

3.1.4 Deskripsi *Item*

Pada permainan ini terdapat tiga *item* utama yang harus didapatkan oleh *player* untuk bisa menyelaikan misi dalam permainan ini.

a. Koin Perak

Setiap kali mendapatkan satu koin perak pada permainan ini maka poin *player* bertambah lima poin.

Gambar 3.4 Koin Perak

b. Koin Emas

Setiap kali mendapatkan satu koin emas pada permainan ini maka poin *player* bertambah sepuluh poin dan satu pengetahuan tentang nabi dan rasul ditampilkan. Sehingga bisa menambah wawasan pengetahuan *player* tentang nabi dan rasul.

Gambar 3.5 Koin Emas

c. Berlian

Jika poin yang dikumpulkan *player* dari hasil mengumpulkan koin emas dan koin perak teah mencukupi sesuai dengan target poin yang harus diperoleh pada level yang dimainkan, maka *player* harus segera mencari dan mengambil item berlian ini untuk bisa menyelesaikan permainan.

Gambar 3.6 Berlian

3.2 Perancangan Finite State Machine

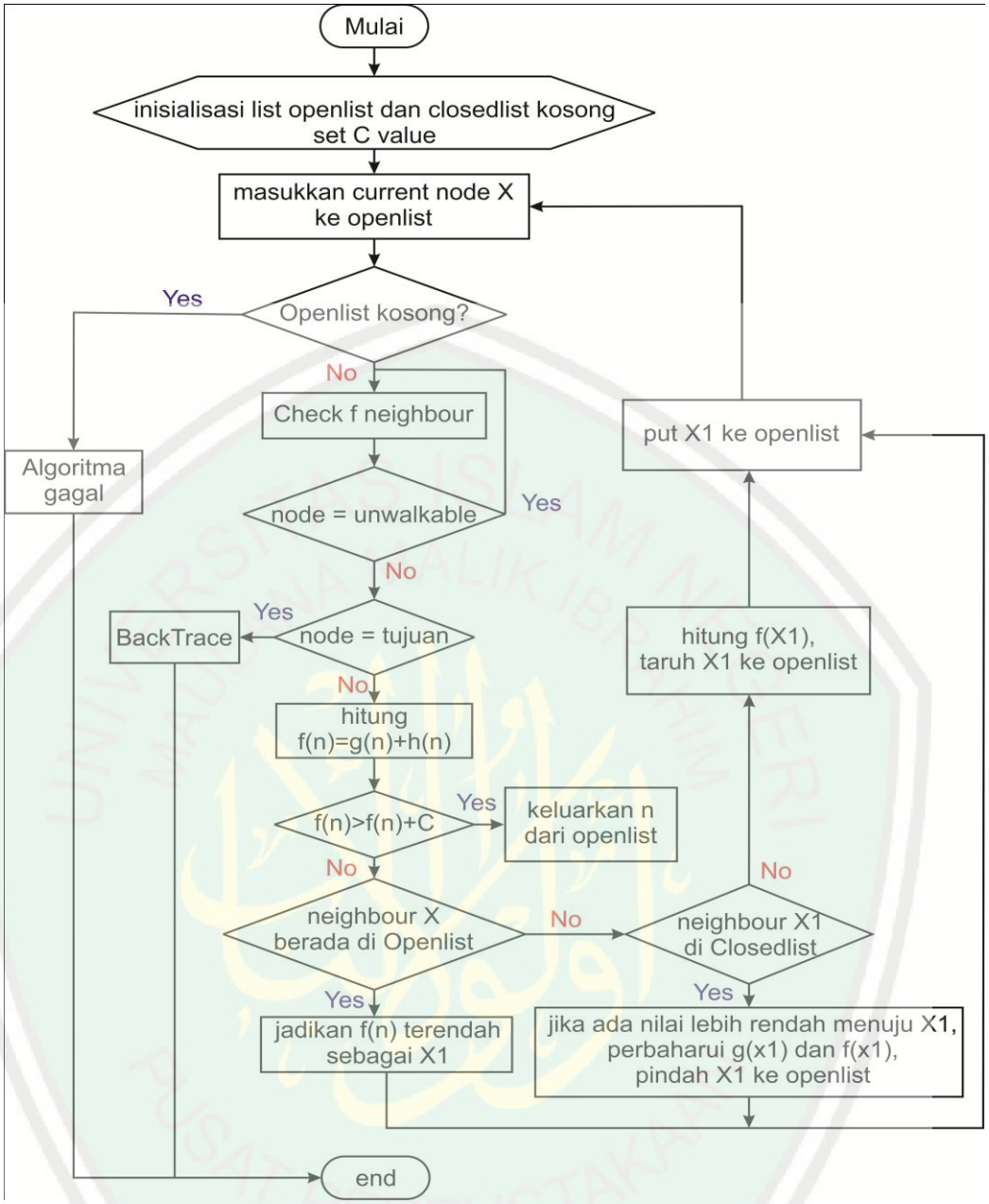
Implementasi FSM di *game* ini adalah untuk mengatur perilaku NPC. Sedangkan perilaku karakter utama mengikuti perintah orang yang bermain *game*. Pada permainan ini terdapat dua jenis NPC, yaitu NPC *Enemy* dan NPC *Army*. Jumlah NPC *Enemy* dan NPC *Army* pada *game* berbeda-beda sesuai dengan level yang dimainkan. Pergerakan NPC otomatis sesuai dengan yang diprogramkan, FSM berikut ini menggambarkan perilaku NPC *Enemy* dan NPC *Army*.

Gambar 3.7 FSM NPC *Enemy*

Gambar 3.8 FSM NPC *Army*

3.3 Perancangan Algoritma *Partial Expansion A**

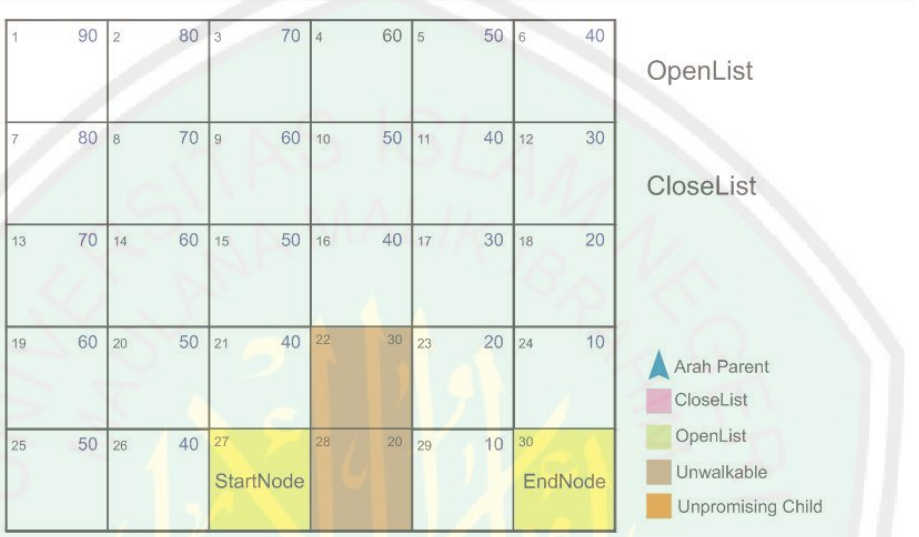
Implementasi algoritma *Partial Expansion A** terdapat pada NPC *Army*. Pada sub bab ini terdapat *flowchart* dan penghitungan manual algoritma tersebut.



Gambar 3.9 *Flowchart* Algoritma *Partial Expansion A**

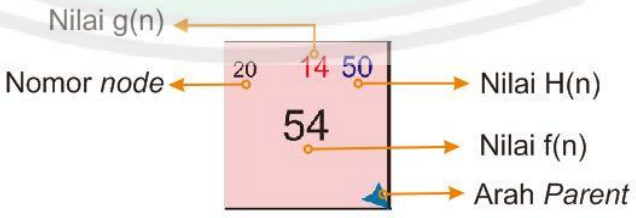
Gambar 3.9 merupakan langkah-langkah penyelesaian masalah pencarian rute terpendek dengan menggunakan algoritma PEA* dalam bentuk *flowchart* yang akan diimplementasikan pada aplikasi *game* di penelitian ini. Untuk lebih jelasnya tentang pengimplementaian algoritma tersebut ke dalam aplikasi *game*, pada sub bab ini akan dilakukan simulasi penghitungan manual algoritma PEA*. Berikut adalah contoh arena yang digunakan untuk simulasi perhitungan manual pencarian

rute terdekat dengan PEA*. Warna kuning adalah *startingpoin/ startnode* sekaligus juga *goal/endpoint/endnode*, pada posisi yang berbeda. Node berwarna coklat adalah penghalang/*unwalkable*. Tujuan dari simulasi ini adalah mencari rute terpendek dari kotak hijau ke kotak biru tanpa melewati penghalang/kotak warna merah.



Gambar 3.10 Arena Untuk Simulasi Algoritma A* dan PEA*

Ketika penghitungan dimulai, warna hijau adalah *open list* dan warna pink adalah *closed list*. Didalam satu kotak atau *node* ada beberapa parameter, diantaranya: G, H, F, koordinat *node* dan arah/*direction* yang ditunjuk merupakan *parent node* tersebut. Untuk lebih jelasnya lihat gambar dibawah ini.



Gambar 3.11 Indeks Pada Kotak (*node*) Saat Simulasi Algoritma A* dan PEA*

Skor atau biaya disetiap *node* dilambangkan dengan F. Pada algoritma PEA* nilai F dapat diperoleh dengan rumus berikut ini:

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

dimana, *Promising Child* adalah

$$f(n) <= C+F(f)$$

dengan:

f(n) = nilai *child(neighbour)*

g(n) = biaya sebenarnya / jarak dari *start node*

h(n) = biaya perkiraan / jarak dari *end node*

F(n)= nilai *node Parent*

C = variable pemotong dengan nilai non-negatif *(Cutoff)*

Promising Child = Node child yang disimpan ke *Open List*

Untuk nilai G diasumsikan setiap langkah dari *Parent* adalah legal baik vertikal, horizontal, maupun diagonal dengan catatan tidak membentur tembok. Setiap langkah yang diizinkan kita berikan nilai G dimana G adalah *cost* atau biaya dalam setiap langkah. Dalam kasus ini kita akan berikan nilai 10 untuk setiap langkah vertikal maupun horizontal, dan 14 untuk diagonal. Nilai 14 kita dapatkan dari perhitungan pitagoras dimana 14,1421 = *sqrt(sqr(10)+sqr(10))*.

Berikut adalah simulasi penghitungan manual algoritma PEA*. Pada simulasi ini kita akan menggunakan *Cutoff* dengan nilai 35. Penjelasan lengkap ada pada gambar dan keterangan langkah-langkah penghitungan manual algoritma PEA* dibawah ini.

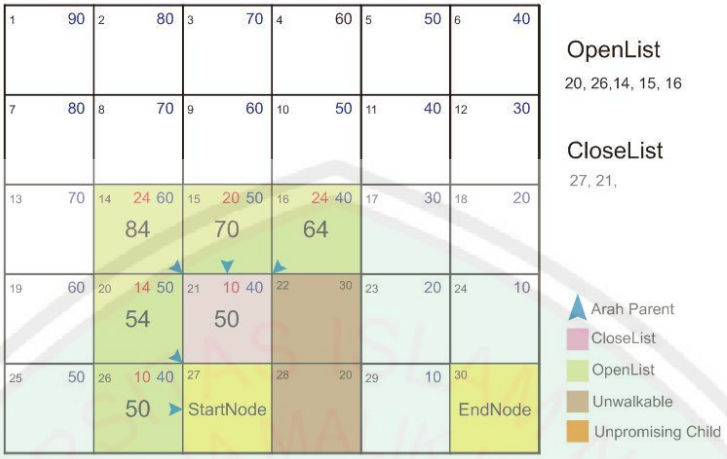
Gambar 3.12 Simulasi Penghitungan Manual Algoritma PEA* Langkah 1

Langkah pertama, pada Openlist masih kosong. Kita akan memasukkan StarNode kedalam Openlist, karena dianggap node terbaik.

Gambar 3.13 Simulasi Penghitungan Manual Algoritma PEA* Langkah 2

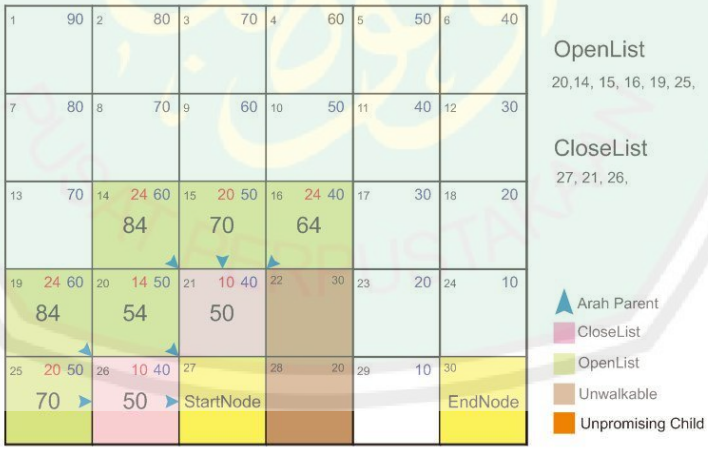
Selanjutnya langkah kedua kita memasukkan semua neighbors, namun pada neighbours terdeteksi adanya Unwalkable dan tidak dimasukkan pada Openlist. Sehingga hanya node 20, 21, dan 26 yang dimasukkan ke open. Node 27 yang telah selesai diperiksa dimasukkan kedalam closelist. Node yang telah masuk dalam openlist akan diperiksa biaya masing-masing node, dan didapati node dengan biaya

terendah yaitu node 21 dan 26. Kita akan mengambil node 21 sebagai bestnode dan kita jadikan parentnode. Keluarkan node 21, masukkan dalam closelist.



Gambar 3.14 Simulasi Penghitungan Manual Algoritma PEA* Langkah 3

Langkah berikutnya memasukkan node neighbours dari parent 21, yaitu node 14, 15, dan 16. Hitung biayanya, dan cari biayaterendah dari semua anggota di openlist. Kita dapati node 26 menjadi bestnode dan jadikan parent. Masukkan node 26 ke closelist.

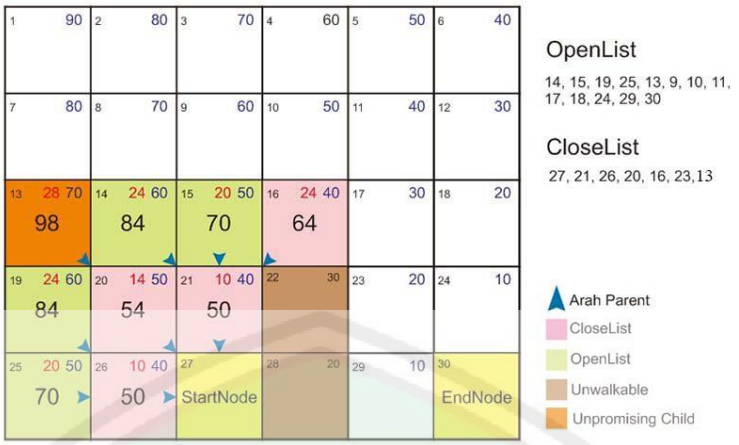


Gambar 3.15 Simulasi Penghitungan Manual Algoritma PEA * Langkah 4

Masukkan neighbour dari parent 26 ke openlist, 19, dan 25. Hitung biaya tiap node, dan jadikan biaya terendah dari semua node pada openlist sebagai Parent. Terpilih node20, masukkan dalam closelist.

Gambar 3.16 Simulasi Penghitungan Manual Algoritma PEA* Langkah 5

Masukkan neighbours dari parent20 ke openlist, yaitu node13. . Hitung biaya tiap node, dan jadikan biaya terendah dari semua node pada openlist sebagai Parent. Terpilih node16, dan masukkan ke closelist. Pada PEA* kita hanya memilih node yang memungkinkan saja (*Promising Child*) dengan syarat $f(n) \leq F(n)+C$, kita dapati node13 bukanlah *PromisingChild* karena melebihi nilai jumlah Parent nya dan C. Kita buktikan dengan memasukkan nilai *PromissingChild*, F(20)=54 dijumlah C=35, didapati nilainya adalah 89, sedangkan f(13)=98. Sehingga di keluarkan node13 dari openlist, dan masukkan dalam closelist.

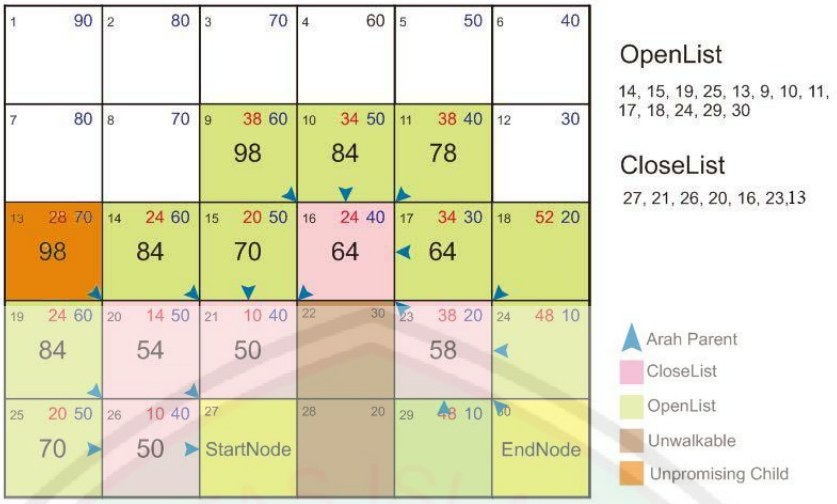


Gambar 3.17 Simulasi Penghitungan Manual Algoritma PEA * Langkah 6

Node sudah teridentifikasi sebagai *UnpromisingChild* dan node16 sebagai *BestNode* dan dijadikan *Parent* masukkan kedalam *closelist*. Kemudian masukkan neighbours dari node16 kedalam Openlist yaitu node 9, 10, 11, 17, 23. Hitung biayanya, dan cari biayaterendah dari semua anggota di openlist.

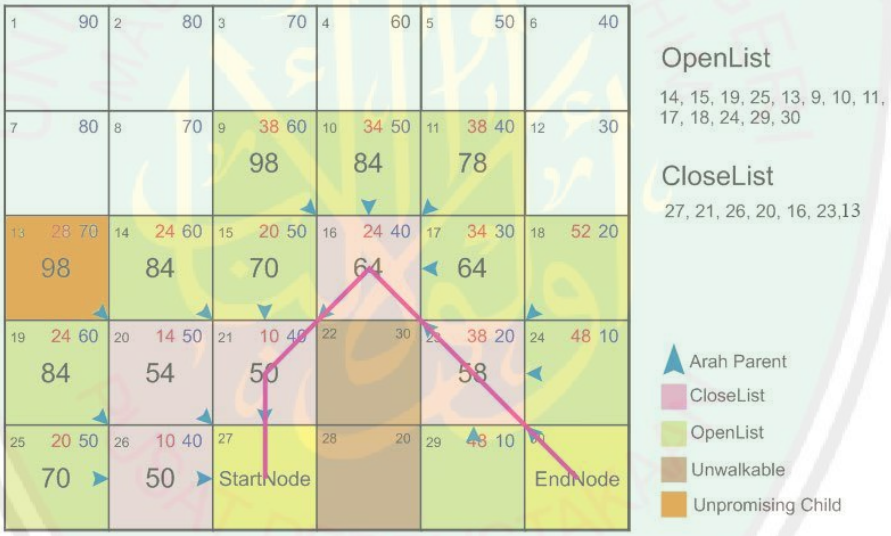
Gambar 3.18 Simulasi Penghitungan Manual Algoritma PEA * Langkah 7

Cek biaya terendah dari semua anggota *openlist*, dan didapati node23 sesbagai biaya terendah. Jadikan node23 sebagai *parent*, dan masukkan kedalam closelist.



Gambar 3.19 Simulasi Penghitungan Manual Algoritma PEA * Langkah 8

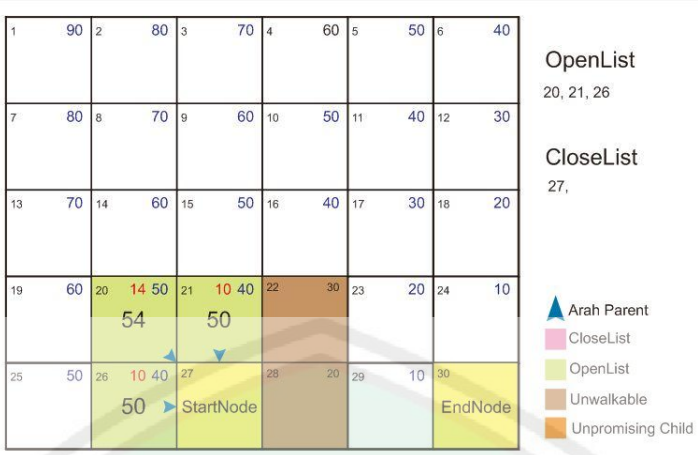
Masukkan neighbours dari parent23, yaitu node 18, 24, 29.



Gambar 3.20 Simulasi Penghitungan Manual Algoritma PEA* Langkah 9

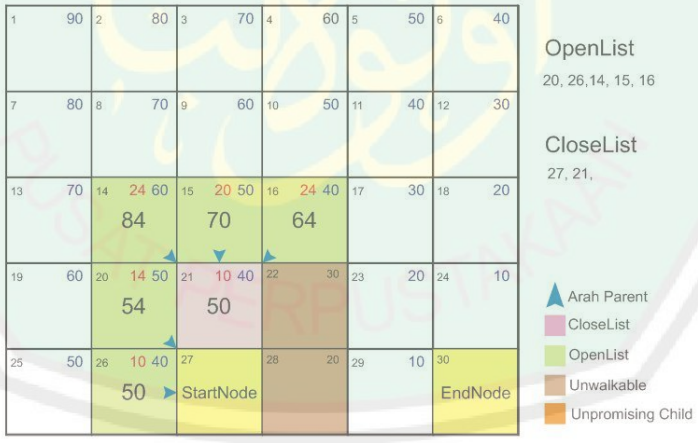
Ternyata salah satu dari neighbours parent23 merupakan *EndNote*. ***Loop*** perhitungan dihentikan dan lakukan *backtrace* ke *StartNode* sesuai arah parent yang telah dibuat sebelumnya.

Sekarang mari kita bandingkan dengan algoritma A* dalam proses pencarian. Kali ini kita tidak membutuhkan variable C, karena algoritma A* memang tidak menggunakan seleksi dalam percabangannya.



Gambar 3.21 Simulasi Penghitungan Manual Algoritma A* Langkah 1

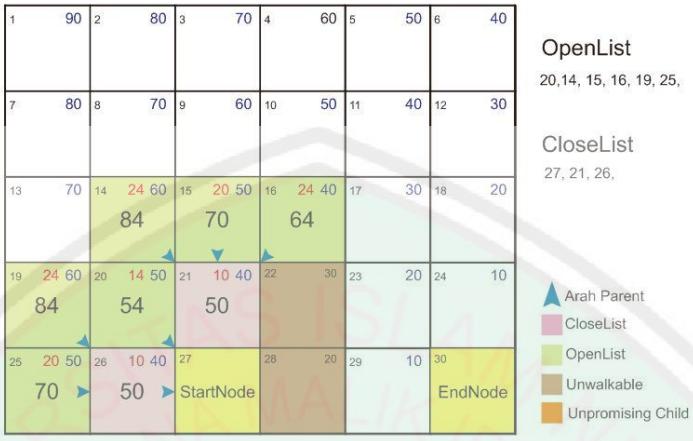
Langkah pertama kita masukkan StartNode ke openlist. Karena hanya berisi 1 node pada openlist, node27 dijadikan Parent dan pop ke Closelist. Bangkitkan Neighbours, node 20, 21, 26, masukkan ke openlist. Hitung biaya masing-masing node, dan jadikan node anggota openlist dengan biaya terendah sebagai Parent, lalu pop ke Closelist. Yaitu node21.



Gambar 3.22 Simulasi Penghitungan Manual Algoritma A* Langkah 2

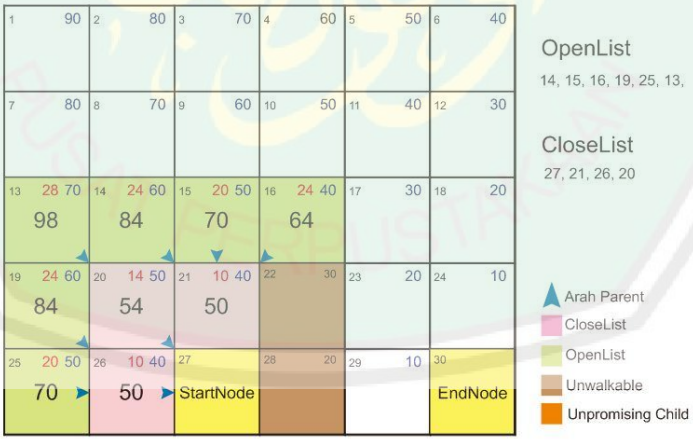
Langkah berikutnya memasukkan node neighbours dari parent 21, yaitu node 14, 15, dan 16. Hitung biayanya, dan cari biayaterendah dari semua anggota

di openlist. Kita dapati node 26 menjadi bestnode dan jadikan parent. Masukkan node 26 ke closelist.



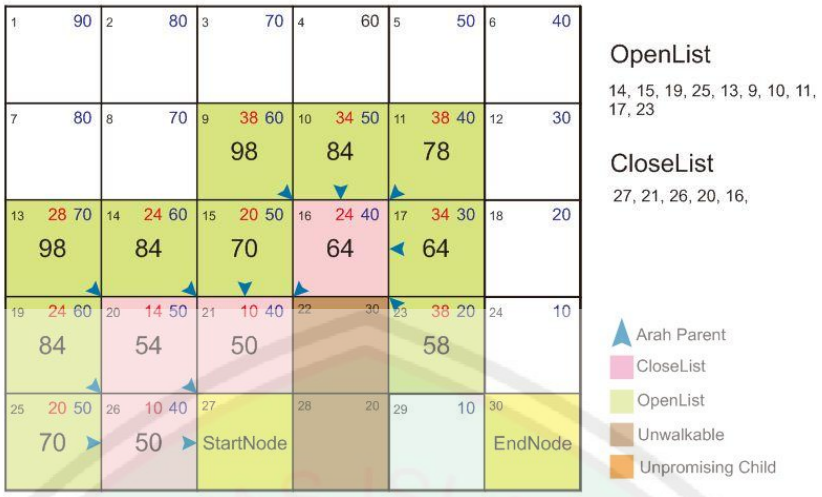
Gambar 3.23 Simulasi Penghitungan Manual Algoritma A* Langkah 6

Masukkan neighbour dari parent 26 ke openlist, 19, dan 25. Hitung biaya tiap node, dan jadikan biaya terendah dari semua node pada openlist sebagai Parent. Terpilih node20, masukkan dalam closelist.



Gambar 3.24 Simulasi Penghitungan Manual Algoritma A* Langkah 3

Jadikan biaya terendah dari semua node pada openlist sebagai Parent. Terpilih node16, masukkan dalam closelist.

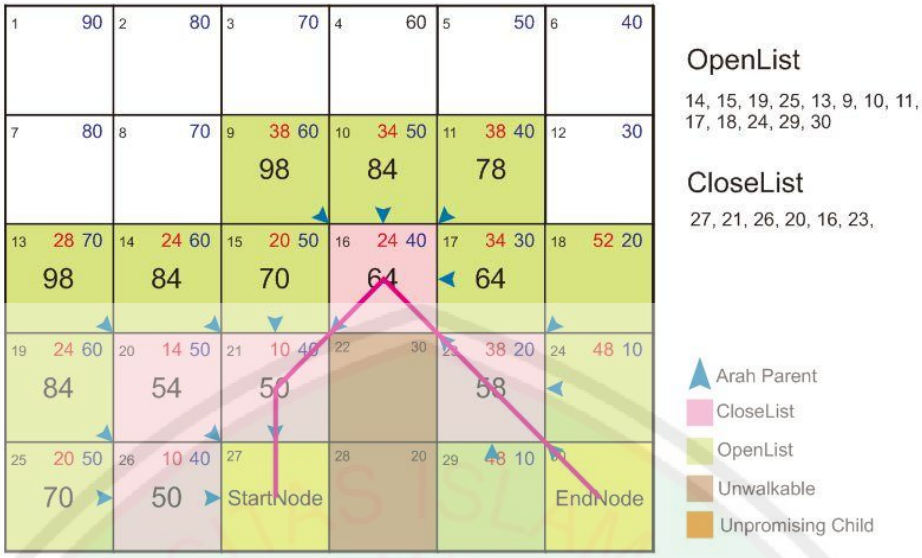


Gambar 3.25 Simulasi Penghitungan Manual Algoritma A* Langkah 4

Masukkan neighbor node 16 ke Openlist. Hitung biaya masing-masing node, dan jadikan node anggota openlist dengan biaya terendah sebagai Parent, lalu pop ke Closelist. Terpilih node23 sebagai node dengan biaya terendah.

Gambar 3.26 Simulasi Penghitungan Manual Algoritma A* Langkah 5

Bangkitkan Neighbour node23, masukkan dalam openlist. Terdeteksi adanya EndNode pada salah satu child dari parent23. Maka loop perhitungan dihentikan.



Gambar 3.26 Simulasi Penghitungan Manual Algoritma A* Langkah 6

Langkah terakhir lakukan BackTrace kearah StartNode sesuai dengan arah parent sudah ditentukan sebelumnya.

Simulasi algoritma PEA* dan A* telah kita lakukan sesuai dengan aturannya masing-masing. Kita dapati pada algoritma PEA* sebuah Node yang tidak menjanjikan dan langsung dimasukkan kedalam CloseList, yang berarti pada kondisi yang lebih kompleks akan lebih sedikit node yang tersimpan dalam openlist. Kinerja komputer pun lebih ringan, dikarenakan pengurangan percabangan yang tidak diperlukan. Dapat kita Tarik kesimpulan bahwa PEA* lebih baik dari A* dalam pemecahan masalah percabangan yang besar.

BAB IV

HASIL DAN PEMBAHASAN

4.1 Implementasi

Bab ini membahas mengenai implementasi dari perencanaan yang telah diajukan. Selain itu pada bab ini dilakukan pengujian terhadap aplikasi *game* untuk mengetahui apakah aplikasi *game* tersebut telah berjalan sesuai dengan tujuan penelitian yang ingin dicapai.

4.1.1 Kebutuhan Perangkat Keras Untuk Uji Coba

Perangkat keras yang digunakan untuk menguji perangkat lunak dari aplikasi *game* ini adalah sebagai berikut:

Tabel 4.1 Kebutuhan Perangkat Keras

No	Perangkat Keras	Spesifikasi
1	*Processor*	*Intel® Core™ i5 CPU U 470 @1,33GHz (4 CPUs)*
2	RAM	4096 MB
3	VGA	*Intel® HD Graphics*
4	HDD	320 GB
5	*Monitor*	11,6”
6	*Speaker*	*On*
7	*Mouse & Keyboard*	*On*

4.1.2 Kebutuhan Perangkat Lunak

Perangkat keras yang diperlukan untuk mengimplementasikan perangkat lunak dari aplikasi *game* ini adalah sebagai berikut:

Tabel 4.2 Kebutuhan Perangkat Lunak

No	Perangkat Lunak	Spesifikasi
1	Sistem Operasi	*Windows 7 64 Bit*
2	*Game Engine*	*Unity 5.2.0f3*
3	*Image Editor*	*Adobe Photoshop CS3*
4	Desain 3D	*Blender 2.7*
5	*Script Writer*	*Mono Develop*

4.1.3 Implementasi Algoritma *Partial Expansion A**

Implementasi kecerdasan buatan pada penelitian ini diterapkan pada NPC *Army* dengan menggunakan algoritma PEA* untuk mencari rute terpendek dari posisi NPC ke tujuan / koin perak dan kemudian bergerak untuk mengambil semua koin perak di arena permainan. Algoritma PEA* ini diimplementasikan menggunakan bahasa C#, *method*/fungsi yang digunakan adalah sebagai berikut:

Tabel 4.3 Keterangan Implementasi Algoritma *Partial Expansion A**

No	***Method* / Fungsi**	**Keterangan**
1	Grid grid; private bool cek; PathRequestManager request Manager; private, double;	Deklarasi variabel
2	void Awake() { requestManager= GetComponent <PathRequestManager>(); grid = GetComponent<Grid>();}	Memanggil komponen yang dibutuhkan dalam penghitungan algoritma PEA *

3	void Start(){
	angka = 0;
	cek = false;}
		Nilai pada variabel pada saat permainan dimulai

4	if (sNode.walkable && tNode.walkable) {	Cek apakah lokasi awal dan lokasi tujuan dapat dilalui
	List<Node> opens = new List<Node> ();	Membuat variabel untuk menampung *open node*

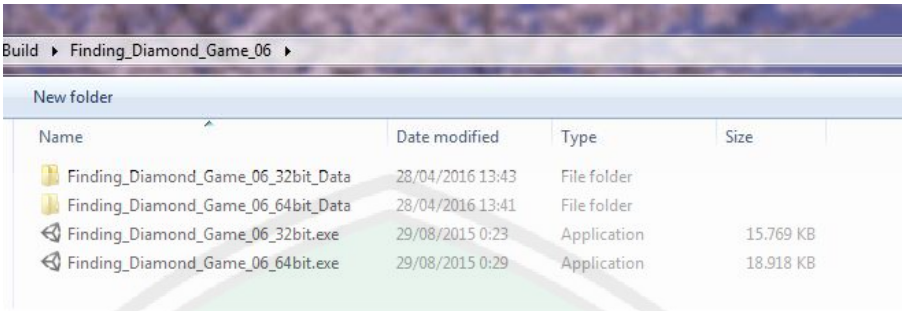
	foreach (Node tetangga in grid.GetNeighbours(sNode)) { tetangga.hCost = GetDistance (tetangga, tNode); opens.Add (tetangga);}	Menghitung nilai H setiap tetangga dari *startnode* dan memasukkannya ke *open node*
	if (opens.Count > 0) { Node cNode = opens [0];	Cek *open node* sudah terisi, kemudian menetapkan nilai *cNode* adalah *node open* indeks pertama
	for (int i = 1; i < opens.Count; i ++) { if (opens [i].hCost < cNode.hCost) { cNode = opens [i];}}	Melakukan perulangan untuk mencari nilai H terkecil dari tetangga *startnode*
5	IEnumerator FindPath(Vector3 startPos, Vector3 targetPos) {	Mulai pencarian rute terpendek dengan input lokasi awal dan lokasi tujuan
	Vector3[] waypoints = new Vector3[0]; bool pathSuccess = false;	Membuat variabel untuk menyimpan rute dan variabel untuk memberikan status apakah rute rudah ditemukan
	Node startNode = grid.NodeFromWorldPoint(startPos); Node targetNode = grid.NodeFromWorldPoint(targetPos);	Mengambil lokasi awal dan lokasi target/tujuan, kemudian memasukkan ke variabel *startNode* dan *targetNode*
	if (startNode.walkable && targetNode.walkable) {	Cek apakah lokasi awal dan lokasi tujuan dapat dilalui
	List<Node> openSet = new List<Node>(); HashSet<Node> closedSet = new HashSet<Node>();	Membuat variabel *openSet* untuk menampung *open node* dan *closedSet* untuk menampung *closed node*
	openSet.Add(startNode);	Pertama-tama memasukkan *start node* ke *open*
	while (openSet.Count > 0) {	Mulai perulangan untuk mencari rute terpendek, perulangan akan berhenti jika rute sudah ditemukan atau sampai tidak ada *node* di dalam *open*
	Node currentNode = openSet[0]; for (int i = 1; i < openSet.Count; i ++) { if (openSet[i].fCost < currentNode.fCost || openSet[i].fCost == currentNode.fCost && openSet[i].hCost < currentNode.hCost) { currentNode = openSet[i]; }}	Membuat variabel *currentNode*, merupakan *node* yang berada di *open* dengan nilai F paling kecil. Perulangan disamping adalah untuk mendapatkan *node* yang akan dijadikan sebagai *bestNode* atau yang akan dimasukkan ke variabel *currentNode* dari *node-node* yang sudah ada di dalam *open/openSet*
	openSet.Remove(currentNode); closedSet.Add(currentNode);	Memindahkan *node* yang terpilih sebagai *bestNode* dari *open* ke *closed*

	foreach (Node neighbour in grid.GetNeighbours(currentNode)) { if (!neighbour.walkable || closedSet.Contains(neighbour)) { continue; }	Lakukan penghitungan nilai F untuk setiap tetangga dari *currentnode*, jika tetangga tidak dapat dilalui atau node tetangga sudah ada di *closed* maka abaikan dan lanjutkan ke tetangga berikutnya
	double newMovementCostToNeighbour = currentNode.gCost + GetDistance(currentNode, neighbour);	Menghitung nilai G baru dari tetangga *current node*
	if (newMovementCostToNeighbour < neighbour.gCost || !openSet.Contains(neighbour){ neighbour.gCost = newMovementCostToNeighbour; neighbour.hCost = GetDistance(neighbour, targetNode); neighbour.parent = currentNode;	Jika *node* tetangga belum ada di *open* maka hitung nilai G, H, dan F kemudian *set parent node* tetangga adalah *currentnode*
	if (!openSet.Contains(neighbour)) openSet.Add(neighbour);	Jika *node* tetangga belum ada di *open* maka masukkan *node* tetangga ke dalam *open*
	}}}} yield return null; if (pathSuccess) { waypoints = RetracePath(startNode,targetNode);}	Jika *pathSuccess* bernilai *true* yang artinya rute sudah ditemukan, maka telusuri kembali rute dari lokasi awal sampai lokasi tujuan dan simpan ke dalam variabel *waypoints*
6	openSet.Contains(neighbour) && neighbour.fCost<=currentNode.fCost+ C) openSet.Add(neighbour);	Penghitungan syarat Promising childnode
7	IEnumerator FollowPath() { Vector3 direction = target.position - transform.position; transform.rotation = Quaternion.LookRotation (direction);	*Method* yang digunakan untuk menjalankan objek, di dalam *game* ini adalah NPC, pertama-tama NPC akan dibuat menghadap ke target, supaya jalannya nanti menghadap ke depan atau menghadap ke target
8	public void OnPathFound(Vector3[] newPath, bool pathSuccessful) {if (pathSuccessful) { path = newPath; StopCoroutine("FollowPath"); StartCoroutine("FollowPath"); }}	*Method* yang digunakan untuk menentukan kapan NPC akan dijalankan, yaitu jika *pathSuccessful* bernilai *true* atau rute terpendek telah ditemukan maka mulai jalankan *method* untuk menjalankan NPC *Army*

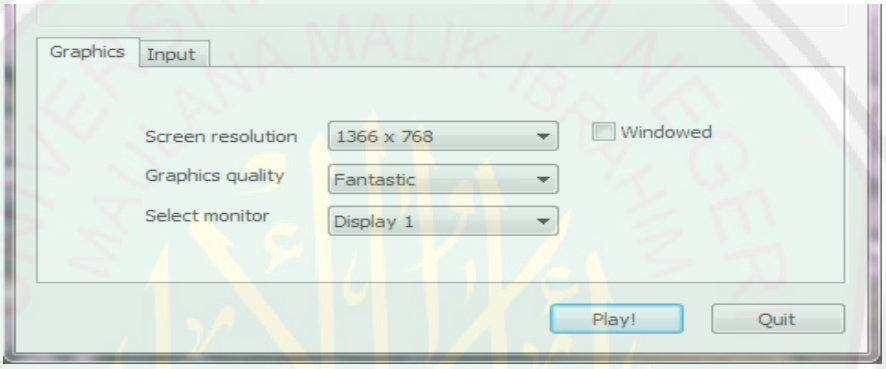
4.1.5 Implementasi Aplikasi *Game*

Aplikasi *game* yang telah selesai dibuat dengan *Unity* kemudian dibuat menjadi *stand alone*. *Stand alone* akan membuat aplikasi *game* dapat dijalankan

tanpa harus menggunakan *Unity Game Engine*. Berikut ini adalah file *game* di *Unity* yang telah berhasil di *build* menjadi *.exe* versi *32bit* dan *64bit.*



Gambar 4.1 Hasil *Build* Aplikasi *Game Finding Diamond*



Gambar 4.2 Pengaturan *Graphics* Pada Aplikasi *Game Finding Diamond*

Jika file *.exe* pada Gambar 4.1 dijalankan, baik yang *32bit* atau *64bit* sesuai dengan komputer yang digunakan, muncul halaman pengaturan grafis dan pengaturan input yang akan digunakan ketika menjalankan aplikasi *game*. Setelah selesai dengan pengaturan tersebut, klik *Play!* untuk masuk ke aplikasi *game*.

Gambar 4.4 *Unity Splash Screen*

Gambar 4.4 adalah tampilan *unity splash screen* yang menunjukkan bahwa aplikasi *game* ini dibuat dengan menggunkan *game engine unity*. *Unity* yanng digunakan untuk membuat aplikasi *game* ini adalah *Unity 5.2*.

Gambar 4.5 Menu Utama Aplikasi *Game Finding Diamond*

Gambar 4.5 adalah tampilan menu utama, terdapat dua tombol yaitu *play game* untuk memulai permainan dan *quit game* untuk menutup aplikasi *game*.

Gambar 4.6 Mulai Bermain *Game Finding Diamond*

Gambar 4.6 adalah tampilan ketika permainan dimulai. Di pojok kiri atas ada informasi yang harus diketahui *player*. Di pojok kanan atas ada tampilan *game*

top view. Di pojok kanan bawah terdapat tombol *Exit*. Di bagian bawah terdapat tulisan yang akan menampilkan konten islami tentang nabi dan rasul.

Gambar 4.7 *Player* Mendapatkan Koin Perak

Gambar 4.7 adalah tampilan ketika *player* berhasil mengambil satu koin perak. Skor *player* bertambah lima poin, koleksi koin perak yang jadi milik *player* bertambah satu, dan persediaan koin perak berkurang satu.

Gambar 4.8 *Player* Mendapatkan Koin Emas

Gambar 4.8 adalah tampilan ketika *player* berhasil mengambil satu koin emas. Skor *player* bertambah sepuluh poin, koleksi koin emas bertambah satu, dan persediaan koin emas berkurang satu. Dan setiap kali *player* mengambil koin emas, di bagian bawah *screen game* akan ditampilkan konten islami tentang nabi dan rasul berupa teks dan juga audio yang akan membacakan isi dari teks tersebut.

Gambar 4.9 Skor Telah Mencapai Target

Gambar 4.9 adalah tampilan ketika poin yang dikumpulkan *player* telah mencapai target. Pada permainan ini target poin adalah 100 poin, contoh gambar diatas poin *player* adalah 105. Maka akan muncul peringatan supaya *player* segera mencari dan mengambil berlian untuk bisa menyelesaikan permainan.

Gambar 4.10 *Player* Berhasil Menyelesaikan Misimya

Gambar 4.10 adalah tampilan ketika *player* telah mengambil berlian yang artinya *player* berhasil menyelesaikan misi pada permainan ini dan *player* dinyatakan menang, setelah itu akan ditampilakan halaman menu utama.

Gambar 4.11 NPC *Army* Mengambil Koin Perak

Gambar 4.11 adalah tampilan ketika salah satu NPC *Army* akan mengambil koin perak. Selain *player*, pada permainan ini ada NPC *Army* yang bisa mengambil semua koin perak yang ada di arena permainan. Jadi *player* harus dengan segera untuk mengambil koin perak supaya tidak kehabisan. Sedangkan unutk koin emas tidak diambil oleh NPC *Army*, supaya *player* lebih banyak belajar tentang nabi dan rasul setiap kali mendapatkan satu koin emas. NPC *Army* ini merupakan sebuah pasukan NPC, pada aplikasi *game* yang dibuat, jumlah pasukan mereka ada 20 NPC. Dibagian selanjutnya akan dilakukan uji coba menentukan kecepatan NPC ini, supaya tidak terlalu cepat atau terlalu lambat untuk menyelesaikan misinya, yaitu untuk menghabiskan koin perak.

Gambar 4.12 NPC *Enemy* Mengejar *Player*

Gambar 4.12 adalah tampilan ketika NPC *Enemy* mengejar *player*, jika *player* bertabrakan dengan NPC *Enemy* maka NPC *Enemy* akan mati dan kesehatan *player* berkurang sepuluh poin. *Player* bisa terus berlari untuk menghindari NPC *Enemy*, karena kecepatan berlari *player* lebih besar daripada kecepatan NPC *Enemy* dalam mengejar *player*. Tetapi jika *player* diam atau sering berhenti selama permainan, maka peluang NPC *Enemy* untuk bisa menabrak *player* semakin besar.

Gambar 4.13 Kesehatan *Player* Habis

Gambar 4.13 adalah tampilan ketika semua NPC *Enemy* telah menabrak *player* sehingga membuat kesehatan *player* habis. Maka dalam hal ini *player* gagal menyelesaikan misi pada permainan ini dan *player* dinyatakan kalah.

Gambar 4.14 Waktu *Player* Habis

Gambar 4.14 adalah tampilan ketika waktu yang dimiliki *player* untuk menyelesaikan permainan telah habis. Maka dalam hal ini *player* gagal menyelesaikan misi pada permainan ini dan *player* dinyatakan kalah.

Gambar 4.15 *Game Over Screen*

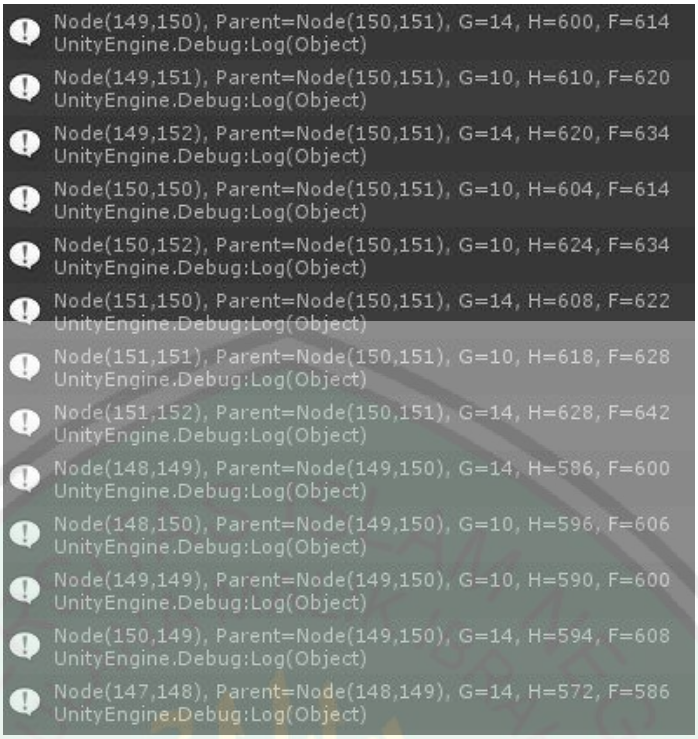
Gambar 4.15 adalah tampilan halaman *game over*, halaman ini muncul setelah *player* dinyatakan kalah karena kesehatan habis atau karena waktu habis. Halaman *game over* ini akan ditampilkan selama lima detik, setelah itu akan ditampilkan kembali halaman menu utama supaya bisa mulai permainan baru.

4.2 Uji Coba

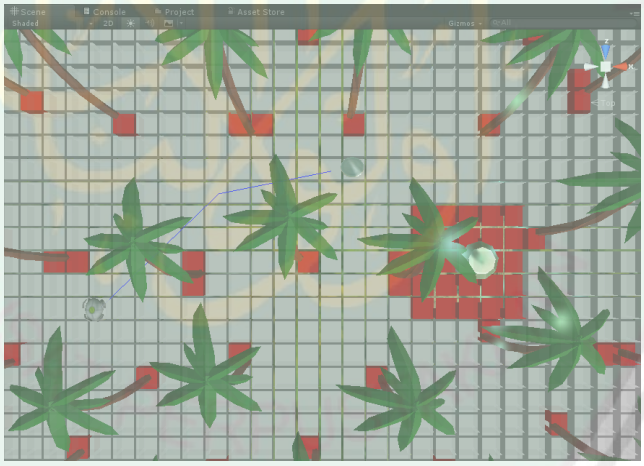
Pada subbab ini membahas tentang uji coba yang telah dilakukan terhadap aplikasi *game* yang telah dibuat. Terdapat tiga uji coba yakni uji coba imlementasi algoritma *Partial Expansion A**. Berikut ini pembahasan uji coba tersebut.

4.2.1 Uji Coba Partial Expansion A*

Uji coba algoritma *Partial Expansion A** dilakukan untuk melihat hasil dari implementasi algoritma tersebut ke dalam *game* yang telah dibuat menggunakan *Unity*. Gambar 4.16 adalah tampilan *tab console* pada *Unity* yang menampilkan hasil-hasil dari penghitungan algoritma PEA* dan pada Gambar 4.17 adalah tampilan *tab scene* pada *Unity* yang menampilkan *grid*, setiap kotaknya merupkan *node*, digunakan untuk penghitungan algortima PEA*. Ditampilkan juga rute terpendek yang berhasil ditemukan algoritma tersebut.

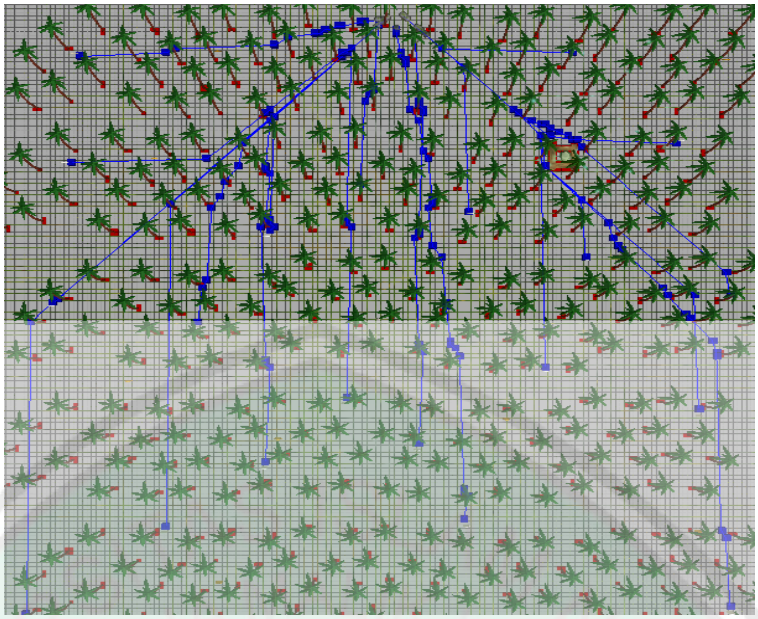


Gambar 4.16 Rute Terpendek yang Berhasil Ditemukan Pada *Console Unity*



Gambar 4.17 Rute Terpendek yang Berhasil Ditemukan Pada *Scene Unity*

Uji coba algoritma PEA* diatas merupakan salah satu contoh pergerakan NPC *Army* untuk mengambil satu koin perak. Sedangkan pada aplikasi *game* yang telah dibuat NPC *Army* akan mengambil semua koin perak yang ada di arena permainan. Gambar dibawah menampilkan semua NPC *Army* beserta rute terpendek yang akan mereka lalui untuk menghabiskan semua koin perak.



Gambar 4.18 Jalur yang Dilalui Semua NPC *Army* Pada *Scene Unity*

Dari gambar 4.18 dapat dilihat jalur dari NPC yang akan mengambil koin perak. Tiap NPC diletakkan dari kordinat-kordinat yang berbeda dan menuju satu kordinat koin perak.

Tabel 4.1 Perbandingan Jumlah Node yang Dibangkitkan Oleh PEA* dan A*

PK	*Start Node*		*Goal Node*		Jumlah *Node* yang Dibangkitkan Oleh Algoritma PEA*			Jumlah *Node* yang Dibangkitkan Oleh Algoritma A*			Selisih Total *Node*	Yang Lebih Baik
	X	Y	X	Y	*Open*	*Closed*	Total	*Open*	*Closed*	Total		
1	86	163	58	44	295	121	416	407	159	566	150	PEA*
2	86	163	92	49	243	116	359	334	25	359	0	Sama
3	86	163	161	88	286	83	369	333	89	422	53	PEA*
4	86	163	5	4	478	161	639	601	38	639	0	Sama
5	86	163	59	105	171	60	231	177	186	363	132	PEA*
6	86	163	120	69	249	96	345	327	263	590	245	PEA*
7	86	163	129	98	205	73	278	225	161	386	108	PEA*
8	86	163	161	6	451	165	616	574	486	1060	444	PEA*
9	86	163	150	129	189	64	253	170	191	361	108	PEA*
10	86	163	36	27	367	138	505	462	223	685	180	PEA*
11	86	163	43	81	250	84	334	302	262	564	230	PEA*
12	86	163	16	152	167	72	239	201	120	321	82	PEA*
13	86	163	102	29	307	136	443	439	438	877	434	PEA*

14	86	163	14	124	223	75	298	303	570	873	575	PEA*
15	86	163	76	61	226	104	330	234	100	334	4	PEA*
16	86	163	11	86	303	79	382	356	106	462	80	PEA*
17	86	163	154	58	331	113	444	420	80	500	56	PEA*
18	86	163	79	112	112	48	160	112	49	161	1	PEA*
19	86	163	103	110	140	55	195	176	251	427	232	PEA*
20	86	163	127	153	106	41	147	102	45	147	0	Sama

Tabel 4.1 dilakukan uji coba perbandingan *node* yang harus dibangkitkan untuk menemukan rute terpendek pada algoritma PEA* dan A*. Dalam penemuan rute, algoritma PEA* membangkitkan lebih sedikit *node* dari pada A*, jadi algoritma PEA* membutuh memori lebih kecil dari pada algoritma A*.

Tabel 4.2 Percobaan nilai Cutoff

Cut off	Percobaan nilai Cutoff pada NPC Army																			
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
50	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗
100	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗
150	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗
200	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗
250	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗
300	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗
350	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗
450	✗	✓	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗
500	✗	✓	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗
550	✗	✓	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✓	✗	✗	✗
600	✗	✓	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✓	✗	✗	✗
650	✗	✓	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✓	✗	✗	✗
700	✓	✓	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✓	✗	✗	✗
750	✓	✓	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✓	✓	✗	✗	✗
800	✓	✓	✓	✓	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✓	✓	✗	✓	✗
850	✓	✓	✓	✓	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✓	✓	✗	✓	✗
900	✓	✓	✓	✓	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✓	✓	✗	✓	✓
950	✓	✓	✓	✓	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✓	✓	✗	✓	✓
1000	✓	✓	✓	✓	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✓	✓	✗	✓	✓
1050	✓	✓	✓	✓	✗	✗	✗	✗	✗	✗	✗	✓	✗	✗	✗	✓	✓	✗	✓	✓

1100	✓	✓	✓	✓	✓	✓	✗	✗	✗	✗	✓	✓	✗	✗	✗	✓	✓	✓	✓	✓
1150	✓	✓	✓	✓	✓	✓	✗	✗	✗	✗	✓	✓	✗	✗	✗	✓	✓	✓	✓	✓
1200	✓	✓	✓	✓	✓	✓	✗	✗	✗	✓	✓	✓	✗	✗	✗	✓	✓	✓	✓	✓
1250	✓	✓	✓	✓	✓	✓	✗	✗	✗	✓	✓	✓	✗	✗	✗	✓	✓	✓	✓	✓
1300	✓	✓	✓	✓	✓	✓	✗	✗	✗	✓	✓	✓	✗	✗	✗	✓	✓	✓	✓	✓
1350	✓	✓	✓	✓	✓	✓	✓	✗	✗	✓	✓	✓	✓	✗	✗	✓	✓	✓	✓	✓
1400	✓	✓	✓	✓	✓	✓	✓	✗	✗	✓	✓	✓	✓	✗	✗	✓	✓	✓	✓	✓
1450	✓	✓	✓	✓	✓	✓	✓	✓	✗	✓	✓	✓	✓	✗	✗	✓	✓	✓	✓	✓
1500	✓	✓	✓	✓	✓	✓	✓	✓	✗	✓	✓	✓	✓	✗	✗	✓	✓	✓	✓	✓
1550	✓	✓	✓	✓	✓	✓	✓	✓	✗	✓	✓	✓	✓	✗	✗	✓	✓	✓	✓	✓
1600	✓	✓	✓	✓	✓	✓	✓	✓	✗	✓	✓	✓	✓	✓	✗	✓	✓	✓	✓	✓
1650	✓	✓	✓	✓	✓	✓	✓	✓	✗	✓	✓	✓	✓	✓	✗	✓	✓	✓	✓	✓
1700	✓	✓	✓	✓	✓	✓	✓	✓	✗	✓	✓	✓	✓	✓	✗	✓	✓	✓	✓	✓
1750	✓	✓	✓	✓	✓	✓	✓	✓	✗	✓	✓	✓	✓	✓	✗	✓	✓	✓	✓	✓
1800	✓	✓	✓	✓	✓	✓	✓	✓	✗	✓	✓	✓	✓	✓	✗	✓	✓	✓	✓	✓
1850	✓	✓	✓	✓	✓	✓	✓	✓	✗	✓	✓	✓	✓	✓	✗	✓	✓	✓	✓	✓
1900	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✗	✓	✓	✓	✓	✓
1950	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
2000	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓

Table 4.2 menunjukkan nilai dari Cutoff yang ideal untuk di terapkan pada keseluruhan NPC Army. Pada game ini menggunakan 20 buah NPC Army yang telah ditanam PEA*. Dari 20 buah NPC Army tersebut didapat nilai Cutoff terkecil yang bisa dipakai oleh semua NPC Army, yaitu 1950.

Tabel 4.3 Perbandingan Waktu Pencarian Rute Terpendek Pada PEA* dan A*

No	*Start Node*		*Goal Node*		Waktu (ms) yang Dibutuhkan Algoritma PEA* Untuk Menemukan Rute Terpendek					Waktu (ms) yang Dibutuhkan Algoritma A* Untuk Menemukan Rute Terpendek					Yang Lebih Baik
	X	Y	X	Y	1	2	3	4	5	1	2	3	4	5	
1	52	98	62	66	7	7	8	8	7	8	8	9	9	8	PEA*
2	52	98	76	91	7	7	7	8	7	7	7	8	7	7	Sama
3	52	98	77	59	7	7	8	7	7	8	8	8	8	8	PEA*
4	52	98	89	77	7	7	8	8	7	7	7	8	8	7	Sama

5	52	98	96	53	8	7	8	8	8	9	9	9	9	9	PEA*
6	52	98	71	41	7	8	8	8	8	10	10	10	10	10	PEA*
7	52	98	92	34	8	8	8	8	8	8	9	8	9	9	PEA*
8	52	98	61	17	8	8	8	8	8	12	12	12	12	12	PEA*
9	52	98	96	4	9	9	9	8	9	9	9	9	9	9	PEA*
10	52	98	55	29	8	8	8	8	8	9	9	10	9	9	PEA*
11	52	98	45	36	8	8	7	7	7	9	9	10	10	10	PEA*
12	52	98	26	48	7	8	8	8	7	8	8	8	8	9	PEA*
13	52	98	35	26	8	8	8	8	8	12	11	11	12	11	PEA*
14	52	98	22	16	8	8	9	9	8	13	13	13	13	13	PEA*
15	52	98	3	2	8	9	8	8	8	8	8	8	9	8	Sama
16	52	98	35	62	8	7	7	7	7	8	8	8	8	8	PEA*
17	52	98	47	70	7	7	7	7	7	9	8	7	9	8	PEA*
18	52	98	7	52	8	7	7	7	8	8	7	7	7	8	Sama
19	52	98	10	91	7	7	7	7	7	9	9	10	9	9	PEA*
20	52	98	8	74	7	7	7	7	7	7	7	7	8	7	PEA*

Selanjutnya pada Tabel 4.3 dilakukan uji coba perbandingan waktu yang dibutuhkan untuk menemukan rute terpendek pada algoritma PEA* dan A*. Waktu yang dibutuhkan oleh kedua algoritma tersebut untuk menemukan rute terpendek kurang lebih sama hanya selisih beberapa *milisecond*. Algoritma PEA* mempunyai catatan waktu yang sedikit lebih baik dari pada algoritma A*. Algoritma PEA* memang sangat memungkinkan untuk menemukan rute lebih cepat, karena jumlah *node* yang dibangkitkan lebih sedikit, sehingga algoritma PEA* bisa beberapa langkah lebih cepat dari pada algoritma A*.

4.2.3 Uji Coba Aplikasi *Game*

Uji coba ini dilakukan untuk mengetahui apakah *game* yang telah dibuat dapat diimplementasikan di *personal computer* atau laptop. Berikut hasil pengujian yang disajikan dalam tabel.

Tabel 4.3 Uji Coba Aplikasi *Game*

No	Versi OS	Spesifikasi Perangkat	Keterangan
1	*Windows 7 64bit*	*- Acer Aspire 1830T* *- Processor Intel® Core™ i5 CPU U 470 @1,33GHz* *- RAM 4096MB* *- VGA Intel® HD Graphics* *- Display 11,6”*	Tampilan menu berjalan dengan baik. Tombol berfungsi dengan baik. Tampilan *game* berjalan dengan baik.
2	*Windows 7 32bit*	*- Lenovo G40* *- Processor AMD A6-6310 APU @1,8GHz* *- RAM 2048MB* *- VGA AMD Radeon™ R4 Graphics* *- Display 14”*	Tampilan menu berjalan dengan baik. Tombol berfungsi dengan baik. Tampilan *game* berjalan dengan baik.
3	*Windows 8.1 64bit*	*- Asus A43SA* *- Processor Intel® Core™ i3-2330M @2,2GHz* *- RAM 2048MB* *- VGA AMD Radeon™ HD 6730M (2 GB)* *- Display 14”*	Tampilan menu berjalan dengan baik. Tombol berfungsi dengan baik. Tampilan *game* berjalan dengan baik.
4	*Windows 10 64bit*	*- HP Pavilion 14* *- Processor Intel® Core™ i5 @2,4GHz* *- RAM 4096MB* *- VGA NVIDIA GeForce GT840M (2GB)* *- Display 14”*	Tampilan menu berjalan dengan baik. Tombol berfungsi dengan baik. Tampilan *game* berjalan dengan baik.

Dari pengujian yang dilakukan sebanyak 4 kali pada berbagai *platform windows* yang memiliki sistem operasi dan spesifikasi berbeda. Dapat diketahui presentase hasil pengujian aplikasi *game* pada Tabel 4.12 di bawah ini.

Tabel 4.4 Persentase Hasil Uji Coba Aplikasi *Game*

No	Jenis Pengujian	Baik	
		Jumlah	%
1	Sistem	4	(4/4) x 100 = 100%
2	Tombol	12	(12/12) x 100 = 100%
3	Tampilan	4	(4/4) x 100 = 100%

Tabel 4.4 di atas merupakan tabel yang berisi hasil pengujian *game* terhadap 4 sistem operasi *windows desktop* yang telah dijelaskan pada Tabel 4.3. Hasil persentase yang didapatkan dari pengujian adalah 100% *game* dapat berjalan dengan baik pada *device* tersebut.

4.3 Integrasi Dalam Islam

Mempelajari sejarah dan peradaban islam merupakan bagian penting yang tidak mungkin dipisahkan dari kehidupan kaum muslimin dari masa ke masa. Karena dengan jalan memahami sejarah dengan baik dan benar, kaum muslimin bisa bercermin untuk mengambil banyak pelajaran dan membenahi kekurangan atau kesalahan mereka guna meraih kejayaan dan kemuliaan dunia dan akhirat.

Sebaik-baik kisah sejarah yang dapat diambil pelajaran dan hikmah berharga darinya adalah kisah-kisah yang terdapat dalam ayat-ayat Al-Qur'an dan hadits-hadits yang shahih dari Rasulullah SAW. Karena kisah-kisah tersebut disamping sudah pasti benar, bersumber dari wahyu Allah yang Maha Benar, juga karena kisah-kisah tersebut memang disampaikan oleh Allah untuk menjadi pelajaran bagi orang-orang yang berakal sehat. Allah berfirman :

لَقَدْ كَانَ فِي قَصَصِهِمْ عِبْرَةٌ لِّأُولِي الْأَلْبَابِ ۗ مَا كَانَ حَدِيثًا يُفْتَرَىٰ وَلَٰكِن تَصْدِيقَ الَّذِي بَيْنَ يَدَيْهِ وَتَفْصِيلَ كُلِّ شَيْءٍ وَهُدًى وَرَحْمَةً لِّقَوْمٍ يُؤْمِنُونَ ﴿١١١﴾

Artinya: Sesungguhnya pada kisah-kisah mereka (para Nabi dan umat mereka) itu terdapat pelajaran bagi orang-orang yang mempunyai akal (sehat). Al-Qur'an itu bukanlah cerita yang dibuat-buat, akan tetapi membenarkan (kitab-kitab) yang sebelumnya dan menjelaskan segala sesuatu, serta sebagai petunjuk dan rahmat bagi orang-orang yang beriman. (QS. Yusuf: 111).

. Kisah-kisah yang menggambarkan keadaan para nabi dan umat mereka tersebut, serta yang menjelaskan kemuliaan orang-orang yang beriman dan kebinasaan orang-orang kafir yang mendustakan seruan para nabi, berisi pelajaran bagi orang-orang yang beriman untuk memantapkan keimanan mereka dan menguatkan ketakwaan mereka kepada Allah dengan menjalankan perintah-Nya dan menjauhi larangan-Nya. Kisah-kisah sejarah para nabi adalah termasuk sebab utama untuk mengokohkan dan menyempurnakan keimanan dalam hati orang-orang yang beriman. Allah berfirman :

وَكُلًّا نَّقُصُّ عَلَيْكَ مِنْ أَنۢبَآءِ ٱلرُّسُلِ مَا نُثَبِّتُ بِهِۦ فُؤَادَكَ ۚ وَجَآءَكَ فِى هَٰذِهِ ٱلْحَقُّ وَمَوْعِظَةٌ وَذِكْرَىٰ لِلْمُؤْمِنِينَ ﴿١٢٠﴾

Artinya: Dan semua kisah dari rasul-rasul Kami ceritakan kepadamu, ialah kisah-kisah yang dengannya Kami teguhkan hatimu, dan dalam surat ini telah datang kepadamu kebenaran serta pengajaran dan peringatan bagi orang-orang yang beriman. (QS. Hud: 120).

Dalam ayat ini jelas sekali menunjukkan bahwa kisah-kisah dalam Al-Qur'an tentang ketabahan dan kesabaran para nabi dalam memperjuangkan dan

mendakwahkan agama Allah sangat berpengaruh dalam meneguhkan hati dan keimanan orang-orang yang beriman di jalan Allah.

Mempelajari kisah para nabi juga merupakan implementasi dari rukun iman yang keempat yaitu iman kepada nabi dan rasul Allah. Allah. Allah berfirman:

لَّيْسَ ٱلْبِرَّ أَن تُوَلُّوا۟ وُجُوهَكُمْ قِبَلَ ٱلْمَشْرِقِ وَٱلْمَغْرِبِ وَلَٰكِنَّ ٱلْبِرَّ مَنْ ءَامَنَ بِٱللَّهِ وَٱلْيَوْمِ ٱلْءَاخِرِ وَٱلْمَلَٰٓئِكَةِ وَٱلْكِتَٰبِ وَٱلنَّبِيِّۦنَ وَءَاتَى ٱلْمَالَ عَلَىٰ حُبِّهِۦ ذَوِى ٱلْقُرْبَىٰ وَٱلْيَتَٰمَىٰ وَٱلْمَسَٰكِينَ وَٱبْنَ ٱلسَّبِيلِ وَٱلسَّآئِلِينَ وَفِى ٱلرِّقَابِ وَأَقَامَ ٱلصَّلَوٰةَ وَءَاتَى ٱلزَّكَوٰةَ وَٱلْمُوفُونَ بِعَهْدِهِمْ إِذَا عَٰهَدُوا۟ ۖ وَٱلصَّٰبِرِينَ فِى ٱلْبَأْسَآءِ وَٱلضَّرَّآءِ وَحِينَ ٱلْبَأْسِ ۗ أُو۟لَٰٓئِكَ ٱلَّذِينَ صَدَقُوا۟ ۖ وَأُو۟لَٰٓئِكَ هُمُ ٱلْمُتَّقُونَ ﴿١٧٧﴾

Artinya: Bukanlah menghadapkan wajahmu ke arah timur dan barat itu suatu kebajikan, akan tetapi sesungguhnya kebajikan itu ialah beriman kepada Allah, hari kemudian, malaikat-malaikat, kitab-kitab, nabi-nabi dan memberikan harta yang dicintainya kepada kerabatnya, anak-anak yatim, orang-orang miskin, musafir (yang memerlukan pertolongan) dan orang-orang yang meminta-minta dan (memerdekakan) hamba sahaya, mendirikan shalat, dan menunaikan zakat dan orang-orang yang menepati janjinya apabila ia berjanji, dan orang-orang yang sabar dalam kesempitan, penderitaan dan dalam peperangan. Mereka itulah orang-orang yang benar (imannya) dan mereka itulah orang-orang yang bertakwa. (QS. Al-Baqarah: 177).

Untuk bisa beriman atau meyakini adanya nabi dan rasul Allah salah satu caranya adalah dengan mempelajari kisah-kisah kehidupan mereka. Oleh karena itu melalui aplikasi *game* yang dibuat pada peneletian ini, diharapkan pengguna mampu mengenal nama-nama nabi dan rasul beserta mukjizatnya serta dapat memetik hikmah dalam setiap ilmu yang diberikan. Dengan dimasukkannya konten islami di dalam *game*, menjadikan bermain *game* tidak hanya main-main saja tetapi ada nilai tambah, yaitu bisa menigkatkan pengetahuan keislaman khususnya

tentang nabi dan rasul bagi siapapun yang bermain *game* ini. Semoga banyaknya peminat *game* edukasi atau *game-game* dengan konten islami selaras dengan kian bertambahnya *developer game* untuk memperkaya *game* edukasi sebagai media pembelajaran yang inovatif dan mengenalkan nilai-nilai islam kepada semua orang.

BAB V

PENUTUP

5.1 Kesimpulan

Berdasarkan hasil dari implementasi dan pengujian yang dilakukan terhadap aplikasi *game* yang telah dibuat pada penelitian ini, maka dapat ditarik beberapa kesimpulan diantaranya adalah sebagai berikut:

a. Algoritma PEA* mampu diterapkan pada aplikasi game sebagai pembangkit perilaku pencarian pada NPC Army.
b. Perbedaan antara PEA* dan A* terletak pada jumlah node yang tersimpan pada Openlist dan Closelist, dimana PEA* menyimpan lebih sedikit node.
c. Dengan nilai Cutoff yang tepat, Algoritma PEA* membangkitkan lebih sedikit node daripada algoritma A* sehingga mampu menghemat lebih banyak memori daripada algoritma A*

5.2 Saran

Peneliti yakin dengan penuh kesadaran bahwa dalam pembuatan permainan ini masih banyak kekurangan yang nantinya sangat perlu untuk dilakukan pengembangan demi sumbangsih terhadap ilmu pengetahuan, diantaranya:

1. Permainan ini tidak hanya disajikan dalam *platform desktop* saja, namun juga bisa dikembangkan pada *platform smartphone* khususnya *Android*.
2. Menambah jumlah level permainan dan materi pembelajaran serta aturan untuk kenaikan level sehingga permainan menjadi lebih menarik.

3. Mengimplementasikan berbagai modifikasi algoritma A* lainnya seperti IDA*, SMA*, BDA*, MBDA* pada aplikasi *game*, kemuadian dibandingkan hasilnya dengan algoritm PEA*.

DAFTAR PUSTAKA

Agung Pamungkas, Eka Puji Widiyanto dan Renni Angreni. 2014. *Penerapan Algoritma A* (A Star) Pada Game Edukasi The Maze Island Berbasis Android.* http://eprints.mdp.ac.id/1191/ diakses pada 19 November 2015.

Alif, Ifa. 2014. *3D Wayang Adventure Game Untuk Pengenalan Budaya Wayang Nusantara Menggunakan A* Pathfinding Algorithm Sebagai Pembangkit Perilaku Pencarian Pada NPC*. Skripsi. Malang: UIN Maulana Malik Ibrahim Malang.

Anggara. 2008. *Memahami Teknik Dasar Pembuatan Game Berbasis Flash*. Yogyakarta: Gave Media.

Arif, Yunifa Miftachul. 2010. *Strategi Menyerang pada Game FPS Menggunakan Hierarchy Finite State Machine dan Logika Fuzzy*. Thesis. Surabaya: Pasca Sarjana Teknik Elektro ITS.

Bronstring, Marek. 2012. *What are adventure games?*. www.adventure gamers.com /articles/view/17547 diakses pada 15 Desember 2015.

Durstenfeld, Richard. (July 1964). "Algorithm 235: Random permutation". Communications of the ACM 7 (7): 420. doi:10.1145/364520.364540.

Fadila, Juniardi Nur. 2014. *Aplikasi Permainan Meteor Shooter Menggunakan MCRNG dan A* Sebagai Algoritma Randoming Spawn dan Pencarian User Berbasis Mobile*. Skripsi. Malang: UIN Maulana Malik Ibrahim Malang.

Ghozaly, Feisal dan Achmad Buchori Ismail, 2017. Pendidikan Agama Islam dan Budi Pekerti. Surakarta: Tiga Serangkai.

Hanafi. 2011. *Kisah 25 Nabi & Rasul*. Jakarta: Bintang Indonesia.

Millington, Ian. 2006. *Artificial Intelligence for Games.* California: Morgan Kaufmann Publishing.

Neumann, J. Von dan O. Morgenstern. 1944. *Theory of Games and Economic Behavior*. Princeton New Jersey: Princeton University Press.

Nilwan, Agustinus. 1998. *Pemrograman Animasi dan Game Profesional 4*. Jakarta: Elex Media Komputindo.

Peter Hart, Nils Nilsson dan Bertram Raphael. 1968. *A Formal Basic for the Heuristic Determination of MinimumCost Paths.* http://ai.stanford.edu/~nilsson/OnlinePubs-ils/PublishedPapers/astar.pdf diakses pada 15 Maret 2016.

Riyadi, Puanta Della Maharani. 2009. *Algoritma Pencarian A* dengan Fungsi Heuristik Jarak Manhattan*. Makalah. Bandung: Intitut Teknologi Bandung.

Riwinoto dan Alfian. 2014. *Implementasi Pathfinding dengan Algoritma A* pada Game Funny English Menggunakan Unity 3D Berbasis Graf.* http://p2m.polibatam.ac.id/wp-content/uploads/2015/01/Alfian.pdf diakses pada 19 November 2015.

Rollings, Andrew dan Ernest Adams. 2006. *Fundamental of Game Design (Game Design and Development Series)*. Unite States: Prentice Hall

Ryan Nugraha, Edo Exridores dan Hendri Sopryadi. 2014. *Penerapan Algoritma Fisher-Yates Shuffle Pada Aplikasi The Lost Insect Untuk Pengenalan Jenis Serangga Berbasis Unity 3D.* http://eprints.mdp.ac.id/1369/ diakses pada 11 November 2015.

Seno, dkk. 2014. *Mudah Membuat Game 3 Dimensi Menggunakan Unity 3D*. Yogyakarta: Andi.

Supriyanto, Berry Priangga dan Yoannita. 2014. *Penerapan Algoritma Fisher-Yates Shuffle pada Edugame Guess Caculation Berbasis Android.* http://eprints.mdp.ac.id/1254/ diakses pada 11 November 2015.

Suyanto. 2011. *Artificial Intelligence – Serching, Reasoning, Planning dan Learning*. Bandung: Informatika.

Tim Penyusun Kamus Besar Bahasa Indonesia. 1990. *Kamus Besar Bahasa Indonesia*. Jakarta: Balai Pustaka.