GAME TASHRIF "SI ZAID" MENGGUNAKAN ALGORITMA *MODIFIED BI-DIRECTIONAL A** UNTUK MENGATUR *PATHFINDING NPC ARMY*

SKRIPSI

Oleh:
AHMAD BAIHAQI
NIM. 11650078

JURUSAN TEKNIK INFORMATIKA
FAKULTAS SAINS DAN TEKNOLOGI
UNIVERSITAS ISLAM NEGERI MAULANA MALIK IBRAHIM
MALANG
2016

GAME TASHRIF "SI ZAID" MENGGUNAKAN ALGORITMA *MODIFIED BI-DIRECTIONAL A** UNTUK MENGATUR *PATHFINDING NPC ARMY*

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 :
Ahmad Baihaqi
NIM. 11650078

JURUSAN TEKNIK INFORMATIKA
FAKULTAS SAINS DAN TEKNOLOGI
UNIVERSITAS ISLAM NEGERI MAULANA MALIK IBRAHIM MALANG
2016

GAME TASHRIF "SI ZAID" MENGGUNAKAN ALGORITMA *MODIFIED BI-DIRECTIONAL A** UNTUK MENGATUR *PATHFINDING NPC ARMY*

SKRIPSI

Oleh :

AHMAD BAIHAQI

NIM. 11650078

Telah disetujui oleh:

Dosen Pembimbing I	Dosen Pembimbing II
Dr. Muhammad Faisal, M.T.	**Hani Nurhayati, M.T.**
NIP. 19740510 200501 1 007	**NIP. 19780625 200801 1 004**

Tanggal, 16 Juni 2016

Mengetahui,
Ketua Jurusan Teknik Informatika

Dr. Cahyo Crysdian, M.CS
NIP. 19740424 200901 1 008

GAME TASHRIF "SI ZAID" MENGGUNAKAN ALGORITMA *MODIFIED BI-DIRECTIONAL A** UNTUK MENGATUR *PATHFINDING NPC ARMY*

SKRIPSI

Oleh :

AHMAD BAIHAQI
NIM. 11650078

Telah Dipertahankan di Depan Dewan Penguji Skripsi dan

Dinyatakan Diterima Sebagai Salah Satu Persyaratan Untuk

Memperoleh Gelar Sarjana Komputer (S.Kom)

Tanggal 21 Juni 2016

Susunan Dewan Penguji		Tanda Tangan
1. Penguji Utama	: Fachrul Kurniawan, M.MT NIP. 19771020 200901 1 001	( )
2. Ketua	: Hani Nurhayati, M.T NIP. 19780625 200801 2 006	( )
3. Sekretaris	: Fresy Nugroho, M.T NIP. 19710722 201101 1 001	( )
4. Anggota	: Dr. Muhammad Faisal, M.T NIP. 19740510 200501 1 007	( )

Mengetahui dan Mengesahkan
Ketua Jurusan Teknik Informatika

Dr.Cahyo Crysdian, M.CS
NIP. 19740424 200901 1 008

PERNYATAAN KEASLIAN TULISAN

Saya yang bertanda tangan di bawah ini:

Nama : AHMAD BAIHAQI

NIM : 11650078

Fakultas/ Jurusan : Sains dan Teknologi / Teknik Informatika

Judul Penelitian : ***GAME TASHRIF*** **"SI ZAID" MENGGUNAKAN ALGORITMA** ***MODIFIED BI-DIRECTIONAL A**** **UNTUK MENGATUR** ***PATHFINDING NPC ARMY***

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, 16 Juni 2016

Yang membuat pernyataan

Ahmad Baihaqi

NIM. 11650078

MOTTO

"Barangsiapa tidak mau merasakan pahitnya belajar, ia akan merasakan hinanya kebodohan sepanjang hidupnya"

(Imam Syafi'i)

HALAMAN PERSEMBAHAN

Dengan Rahmat Allah yang maha pengasih lagi maha penyayang...

Dengan ini saya persembahkan karya ini untuk:

Alm. Ayahanda Agus Subroto atas limpahan kasih sayang semasa hidupnya dan memberi rasa rindu yang sangat berarti -

Ibunda Eny Ismawati yang telah mencurahkan kasih sayang dan cinta setulus hati serta dukungan moril dan materil yang tiada bisa terbalaskan -

Abah KH. Marzuqi Mustamar dan Umi Saidatul Mustaghfiroh atas segala bimbingan khususnya tentang ilmu agama hingga membuatku menjadi pribadi yang lebih baik dari sebelumnya -

Zaki, Alfin, Nonok, Lukman, Yasir, dan teman-teman T. Informatika 2011 yang ikut berjuang dalam pengerjaan skripsi

David, Zaim, Nizam, Shobah, Angga, dan teman-teman santri Pondok Pesantren Sabilurosyad Gasek yang telah berbagi pengalaman, inspirasi, canda, tawa, dan keluh kesah bersama selama ini

KATA PENGANTAR

Assalamu'alaikum Wr. Wb.

Segala puji bagi Allah SWT tuhan semesta alam, karena atas segala rahmat dan karunia-Nya kepada penulis sehingga penulis mampu menyelesaikan skripsi dengan judul "*Game Tashrif* "Si Zaid" Menggunakan Algoritma *Modified Bi-directional A** Untuk mengatur *Pathfinding NPC Army*" dengan baik dan lancar. Shalawat dan salam selalu tercurah kepada tauladan terbaik kita Nabi Agung Muhammad SAW yang telah membimbing umatnya dari zaman kegelapan dan kebodohan menuju cahaya islam yang terang *rahmatan lil alamiin* ini.

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. Mudjia Raharjo, M.Si., selaku rektor UIN Maulana Malik Ibrahim Malang beserta seluruh staf. Dharma Bakti Bapak dan Ibu sekalian terhadap Universitas Islam Negeri Malang turut membesarkan dan mencerdaskan penulis.
2. Dr. Hj. Bayyinatul M., drh., 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. Bapak Dr. Cahyo Crysdian, selaku Ketua Jurusan Teknik Informatika Universitas Islam Negeri Maulana Malik Ibrahim Malang, yang sudah

memberi banyak memberi pengetahuan, inspirasi dan pengalaman yang berharga.

4. Bapak 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. Ibu Hani Nurhayati, M.T., selaku dosen pembimbing II yang juga senantiasa memberi masukan dan nasihat serta petunjuk dalam penyusunan skripsi ini.
6. Alm. Ayah, Ibu, dan Adik-adik serta keluarga besar saya 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.
9. Para peneliti yang telah mengembangkan Game dengan Engine *Unity3d* 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 tulis ini bisa bermanfaat dan menginspirasi bagi kita semua. Amin.

Wassalamualaikum Wr. Wb.

Malang, 16 Juni 2016

Penulis

DAFTAR ISI

DAFTAR GAMBAR

DAFTAR TABEL

ABSTRAK

Baihaqi, Ahmad. 2016. ***Game Tashrif* "Si Zaid" Menggunakan Algoritma *Modified Bi-direcional A**** **Untuk Mengatur Pathfinding *NPC Army***. Skripsi. Jurusan Teknik Informatika Fakultas Sains dan Teknologi Universitas Islam Negeri Maulana Malik Ibrahim Malang.

Pembimbing : (I) Dr. Muhammad Faisal, M.T., (II) Hani Nurhayati, M.T.

Kata Kunci: *Game, Tashrif, NPC Army, Modified Bi-directional A*.*

Game merupakan salah satu teknologi yang perkembangannya sangat pesat. Semakin banyak developer yang membuat game dengan berbagai macam genre game baik game desktop maupun mobile.

Bahasa arab merupakan bahasa yang digunakan untuk ibadah umat islam dan juga bahasa Al-qur'an. Di dalam bahasa arab terdapat ilmu tashrif. Tashrif adalah ilmu yang mempelajari perubahan kata dari bentuk satu ke bentuk lain. Pada era globalisasi menghafal kosa kata dan perubahan kata dalam bahasa arab khususnya dalam kalangan anak muda khususnya para santri sudah mulai pudar. Padahal bahasa arab merupakan bahasa umat islam, dan juga merupakan bahasa yang paling populer digunakan setelah bahasa inggris. Penulis ingin membawa semua orang khususnya anak muda untuk lebih mudah menghafal kosa kata dan perubahan kata dalam bahasa arab melalui game yang bertipe petualangan (adventure), dimana dalam game single player ini pemain akan mengumpulkan dan kata tashrif bahasa arab. Sehingga player sedikit demi sedikit akan hafal perubahan kata yang dimainkan dalam game.

Berdasarkan latar belakang tersebut, penulis mencoba membuat game sebagai media pembelajaran untuk stimulus dalam menghafal perubahan kata bahasa arab. Game ini dirancang menggunakan algoritma Modified Bi-directional A* untuk mencari jarak terpendek antara NPC Army dan kata tashrif. Dari hasil uji coba menunjukkan algoritma MBDA* dapat digunakan untuk mencari rute terdekat antara NPC Army dan kata tashrif.

ABSTRACT

Baihaqi, Ahmad. 2016. **Tashrif Game "Si Zaid" By Employing Modified Bi-directional A* Algorithm to Settle Pathfinding NPC Army**. Thesis. Informatics Engineering Department of Science and Technology Faculty Islamic State University Maulana Malik Ibrahim Malang.

Adviser : (I) Dr. Muhammad Faisal, M.T., (II) Hani Nurhayati, M.T.

Keyword: Game, *Tashrif*, NPC Army, Modified Bi-directional A*.

Game is one of technologies which has been developing rapidly. Additionally, many developers create game by employing variety of genres such as desktop or mobile game.

Arabic is a language which is employed by Moslems to pray and it is also language of Koran. Further, in Arabic there is another knowledge which is well known as Tashrif. It is a knowledge which studies the morphology in Arabic. In this globalization era, memorizing vocabulary and studying morphology specifically in Arabic have continually decreased, whereas Arabic is a main language for Moslems and become the most popular language after English. The researcher aims to encourage everyone especially young people to memorize vocabulary and learn morphology of Arabic easier through an adventure game. In this game the player will collect vocabulary and know the morphology process. As a result, the player gradually will remember the morphology of the words which are played through the game.

Accordingly, the researcher created a game as learning medium to stimulate everyone in memorizing Arabic. It was designed by employing modified Bi-directional A* algorithm to find the shortest path between NPC and Tashrif words. Furthermore, from the test which has been conducted by the researcher, MBDA* algorithm was able to find the shortest path between NPC Army and Tashrif words.

مستغلص البحث

بيهقي.احمد.٢٠١٦. تاشريف لعبة "زيد" استخدام خوارزمية ديريسيونال ثنائية معدلة أ * "الاستطلاعية لمجلس الشعب" لتنظيم الجيش. أطروحة. ومن المؤسف إدارة الكمبيوتر كلية الهندسة للعلوم والتكنولوجيا في جامعة الدولة الإسلامية مولانا مالك إبراهيم

المشرف: (I) الدكتور محمد فيصل الماجيستر(II) هاني نورحيتي الماجيستر

كلمات البحث: لعبة, تاشريف, الجيش الوطني، وتعديل ثنائية.

اللعبة هي واحدة من التطور السريع جدا من التكنولوجيا. مطوري المزيد والمزيد الذي خلق مباريات مع مجموعة واسعة من أنواع الألعاب سواء المكتبية والألعاب المحمولة. اللغة العربية هي اللغة المستخدمة تصرف هو دراسة التغيرات .تصرف للعبادة للمسلمين وكذلك لغة القرآن الكريم. في اللغة العربية هناك علم كلمة من شكل إلى شكل آخر. في عصر العولمة والتغيرات حفظ المفردات في العربية وخاصة بين الشباب، وخاصة الطلاب قد بدأت تتلاشى. على الرغم من اللغة العربية هي لغة المسلمين، وأيضا اللغات الأكثر شعبية المستخدمة بعد الإنجليزية. يود الكاتب أن يحضر جميع الناس، وبخاصة الشباب على حفظ أكثر سهولة المفردات وتغيير كلمة في اللغة العربية من خلال أي نوع لعبة المغامرة (المغامرة)، حيث لعبة تصرف العربية. لذلك سيكون لاعب تدريجيا تتغير قال حفظه عبت لاعب واحد، لاعبا ولاعبة جمع وكلمة في اللعبة. وبناء على هذه الخلفية، حاولت الكتاب لجعل اللعبة كوسيلة تعليمية لتحفيز في حفظ التغييرات لإيجاد قالت أقصر مسافة بين مجلس *MBDA الكلمة العربية. تم تصميم هذه اللعبة باستخدام خوارزمية خوارزمية يمكن استخدامها للعثور قال أقصر *MBDA الشعب تصرف الجيش. من نتائج اختبار تظهر الطرق بين مجلس

BAB I

PENDAHULUAN

1.1 Latar Belakang

Bahasa Arab (اللغة العربية) adalah suatu bahasa yang termasuk dalam semit atau rumpun bahasa asal timur tengah dan berkerabat dengan bahasa Ibrani dan bahasa Neo Arami. Bahasa Arab merupakan bahasa resmi di 25 negara di dunia dan dituturkan lebih dari 280 juta orang. Bahasa Arab merupakan bahasa yang digunakan untuk beribadah umat islam karena yang dipakai Al-Qur'an.

Di dalam bahasa arab terdapat ilmu *nahwu* dan ilmu *shorof*. *Nahwu* adalah ilmu dalam bahasa arab untuk mengetahui perubahan bentuk ketika masih satu kata (*Mufrod*) dan ketika sudah tersusun (*Murokkab*) dimana biasanya perubahan ini bisa berupa harokat akhir suatu kata atau bentuk akhir dari suatu kata. Termasuk didalamnya adalah pembahasan *shorof*. Ilmu *shorof* adalah ilmu yang mempelajari kaidah-kaidah perubahan kata, dimana dengan berubahnya kata menjadi perubahan makna kata. Biasanya perubahan kata juga biasa disebut dengan *Tashrif*. Secara garis besar, *shorof* adalah ilmu yang mempelajari tentang *Tashrif*. Dalam *Tashrif*, ada juga yang namanya *wazan* (وزن) secara bahasa artinya adalah timbangan. Secara istilah *wazan* adalah *lafadz-lafadz* yang menjadi timbangan atau patokan yang harus diikuti oleh kalimat lain. *Wazan* ada 10, yaitu *fi'il madhi, fi'il mudhorik, mashdar, mashdar mim, isim fa'il, isim maf'ul, fi'il amr, fi'il nahi, isim zaman makan, isim alat.*

Perkembangan teknologi semakin hari semakin pesat dan tidak dapat dihindari. Dengan perkembangan teknologi, maka semakin terjawab kebutuhan -

kebutuhan seliuruh individu. Memang perkembangan teknologi tidak bisa dilepaskan dari komputer dan sekarang kita bisa melihat canggihnya teknologi yang berbasis komputer termasuk *game*. *Game* merupakan kata dalam bahasa inggris yang artinya permainan. Dalam *game* menggunakan aturan tertentu di mana dengan aturan tersebut pemainan menjadi lebih menarik. Menurut Greg Costikyan (2013), *game* adalah sebentuk karya seni di mana peserta, yang disebut Pemain, membuat keputusan untuk mengelola sumberdaya yang dimilikinya melalui benda di dalam *Game* demi mencapai tujuan. *Genre Game* merupakan penggolongan *game* berdasarkan interaksi dibidangnya. Contoh *genre game* diantaranya *(RPG) Role Playing game, FPS (First Person Shooting), TPS (Third Person Shooting), Strategy, Sport, Simulation, Tycoon, Racing, Action Adventure, Arcade, Fighting*. Dalam perkembanganya, *game* tidak hanya dirancang dan dibuat untuk sarana hiburan saja, namun juga banyak *game* yg bersifat pembelajaran. *Game* yang bersifat pembelajaran sangat membantu anak-anak untuk belajar karena tanpa disadari pada *game* yang telah ditanami nilai-nilai pembelajaran akan mampu diserap oleh mereka yang memainkannya. Menurut Dephne Bavalier (2009) bahwa bermain *game* itu menguntungkan memang fakta. Hasil penelitiannya menunjukkan bahwa proses belajar lewat main *game* ternyata sangat cepat diserap manusia. Dengan kata lain, *game* dapat membantu melatih orang yang memiliki masalah dan cara pemecahannya dan juga meningkatkan konsentrasi. *Game* berjenis media pembelajaran sangat membantu khususnya para guru untuk menyampaikan pembelajaran kepada murid yang kesulitan. Menurut Professor Angela McFarlane (2002), Direktur *Teachers Evaluating Educational Multimedia*, guru-guru mengalami kesulitan untuk memanfaatkan *game* pada saat

jam pelajaran sekolah karena penggunaan *Video game* tidak termasuk dalam kurikulum nasional. McFarlane menambahkan bahwa, seandainya *game-game* tertentu dapat dimainkan di dalam kelas secara legal dan merupakan bagian dari kurikulum, mungkin bukti dari penelitian para ahli tentang manfaat *Video Game* dapat dirasakan.

Pada era globalisasi menghafal kosa kata dan perubahan kata dalam bahasa arab khususnya dalam kalangan anak muda sudah mulai pudar, dimana mereka mulai malas dalam menghafal dan lebih mementingkan bermain gadget mereka. Padahal bahasa arab merupakan bahasa umat islam, dan juga merupakan bahasa yang paling populer digunakan setelah bahasa inggris. Penulis ingin membawa semua orang khususnya anak muda untuk lebih mudah menghafal kosa kata dan perubahan kata dalam bahasa arab melalui *game* yang bertipe petualangan (*adventure*), dimana dalam *game single player* ini pemain akan mengumpulkan dan kata bahasa arab. Sehingga *player* sedikit demi sedikit akan hafal perubahan kata yang dimainkan dalam *game*.

Dalam ajaran agama islam, kita dianjurkan untuk belajar bahasa arab karena kitab suci agama islam Al-qur'an ditulis menggunakan bahasa arab dan juga bacaan-bacaan ibadah menggunakan bahasa arab. Semua itu sudah tertulis di bahwasannya Al-qur'an diturunkan menggunakan bahasa arab pada surat *Az-zukhruf* ayat 3 di bawah ini:

إِنَّا جَعَلْنَٰهُ قُرْءَٰنًا عَرَبِيًّا لَّعَلَّكُمْ تَعْقِلُونَ ﴿٣﴾

Artinya: *"Sesungguhnya Kami menjadikan Al Quran dalam bahasa Arab supaya kamu memahami(nya)."*

Ayat diatas menjelaskan bahwa Allah menurunkan bahasa arab agar kita memahami bahwa Al-Qur'an adalah Ummul Kitab karena memang Al-Qur'an memang kitab yang bernilai tinggi, penuh dengan hikmah, dan merupakan petunjuk untuk umat manusia. Seperti pada surat *Fushilat* ayat 3 di bawah ini:

كِتَٰبٌ فُصِّلَتْ ءَايَٰتُهُۥ قُرْءَانًا عَرَبِيًّا لِّقَوْمٍ يَعْلَمُونَ ﴿٣﴾

Artinya: *"Kitab yang dijelaskan ayat-ayatnya, yakni bacaan dalam bahasa Arab, untuk kaum yang mengetahui."*

Al-Qur'an adalah kitab suci yang mengandung ilmu-ilmu yang tak terbatas jika kita memahaminya dan tak bisa digali secara tuntas. Ilmu yang dikandungnya takkan pernah habis walau terus digali dan dikuras sepanjang masa sepanjang kehidupan dunia masih ada. Al-Qur'an adalah sumber ilmu yang kaya dan abadi. Bagi umat islam, Al-Qur'an menjadi sumber hukum pertama kemudian baru hadits. Oleh karena itu, Al-Qur'an adalah sumber informasi dan ilmu bagi umat manusia.

Pada kasus ini, penulis akan menangani *NPC (Non-Playable Character)* atau juga disebut agen dalam *game* yang tidak dikendalikan langsung oleh pemain. *NPC* secara otomatis dikendalikan oleh komputer, bisa berupa teman ataupun musuh. *NPC* dalam *game* ini berinteraksi dalam lingkungan *game* dengan

menggunakan kecerdasan buatan atau biasa disebut *Artificial Intelligent* (AI). Pada *game* ini *NPC* akan menggunakan algoritma *Modified Bi-directional A** untuk mencari jarak terdekat dengan *player*.

1.2. Identifikasi Masalah

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

1. Bagaimana mengimplementasikan metode *Modified Bi-directional A** untuk mengatur *pathfinding NPC (Non-Playable Character)* dengan menggunakan konten *tashrif*?

1.3. Tujuan Penelitian

1. Mengimplementasikan metode *Modified Bi-directional A** untuk mengatur *pathfinding NPC (Non-Playable Character)* dengan menggunakan konten *tashrif*.

1.4. Batasan Masalah

1. *Game* ini bersifat *single player.*
2. *Game* ini benggunakan *genre Arcade.*
3. *Game* ini menggunakan *Tashrif Istilahi.*
4. Algoritma diterapkan pada *NPC (Non-Playable Character)?*

1.5. Manfaat Penelitian

Manfaat yang diharapkan dari pembuatan *game* ini adalah terbentuknya sebuah *game* edukasi untuk pembelajaran bahasa arab khususnya dalam *Tashrif*,

sehingga memberikan kemudahan untuk pembelajaran *Tashrif* khususnya untuk anak-anak.

BAB II

KAJIAN PUSTAKA

2.1 Kajian Pustaka

2.2.1 *Game* (Permainan)

Game berasal dari kata dalam bahasa inggris yang memiliki arti dasar permainan. Permainan dalam hal ini merujuk pada pengertian "kelincahan intelektual" (*Intelectual Playability*). *Game* juga bisa diartikan sebagai arena keputusan dan aksi permainannya. Di dalam *game* terdapat aturan tertentu sehingga ada yang menang dan ada yang kalah (Rollings & Adams, 2003)

Teori permainan pertama kali ditemukan oleh sekelompok ahli Matematika pada tahun 1944. Teori itu dikemukakan oleh John von Neumann and Oskar Morgenstern yang berisi: "Permainan terdiri atas sekumpulan peraturan yang membangun situasi bersaing dari dua sampai beberapa orang atau kelompok dengan memilih strategi yang dibangun untuk memaksimalkan kemenangan sendiri atau pun untuk meminimalkan kemenangan lawan. Peraturan-peraturan menentukan kemungkinan tindakan untuk setiap pemain, sejumlah keterangan diterima setiap pemain sebagai kemajuan bermain, dan sejumlah kemenangan atau kekalahan dalam berbagai situasi." Pada dasarnya *game* dimainkan oleh anak-anak hingga dewasa, karena memang *game* mempunyai konteks untuk menghibur dan menyenangkan. (Neumann dkk, 1944).

Game dikategorikan dalam beberapa *genre* dalam buku yang berjudul *Andrew Rollings and Ernest Adams on Game Design*, yaitu sebagai berikut (Rolling dkk, 2003):

1. Aksi Petualangan *(Action Adventure)*

   Genre Game ini pada umumnya membuat pemain harus berjalan mengelilingi suatu tempat yang terkondisi, seperti sebuah istana, gua yang berkelok, dan planet yang jauh. Pemain melakukan navigasi suatu area, mencari pesan-pesan rahasia, memperoleh obyek yang memiliki kemampuan yang bervariasi, bertempur dengan musuh, dan lain-lain. Untuk membuat *game* ini, diperlukan perencanaan yang akurat sehingga memiliki alur cerita dari awal sampai tamat yang menarik bagi pemain. Contoh: *Assassins Creed* dan *Resident Evil.*

2. *Arcade*

   Arcade game adalah *genre game* yang tidak terfokus pada cerita, melainkan hanya dimainkan *"just for fun"* atau hanya untuk mengejar poin/*highscore*. Contoh: *Zuma* dan *Angry Bird.*

3. Pertarungan *(Fighting)*

   Pertarungan dua *player* dengan jurus-jurus yang bias dikeluarkan dengan menekan beberapa tombol pada *keyboard* dengan urutan tertentu. Pemain dapat mengeluarkan jurus-jurus ampuh dalam pertarungannya. *Genre fighting* biasanya one on one dalam sebuah arena yang sempit. Contoh: *Street Fighter II* dan *Tekken series*.

4. Balapan *(Racing)*

Racing *game* adalah *game* sejenis racing yg memungkinkan kita untuk mengendalikan sebuah kendaraan untuk memenangkan sebuah balapan. Pada *genre* racing ini biasanya pemain dapat memilih, membeli, mengupgrade mesin kendaraan, atau menambah asesoris kendaraan. Contoh: *Need For Speed* dan *Moto GP*.

5. Peran *(Role Playing)*

Game RPG adalah sebuah *game* di mana *player* memainkan suatu tokoh yang ada dalam *game*. Didalam *game* ini biasanya terdapat unsur seperti experience point, atau perkembangan karakter yang kita mainkan sehingga membuat karakter kita naik level dan semakin kuat. Unsur cerita dalam *game* RPG sangat kental. Ada yang akhir ceritanya bisa kita tentukan sendiri tergantung apa yang kita lakukan dalam *game*. Biasanya di *game* RPG terdapat juga sistem equipment, di mana untuk memperkuat karakter yang kita mainkan diperlukan kombinasi perlengkapan yang mempengaruhi dalam menjalankan *game* RPG. *Game* RPG di bagi menjadi 2 *genre* yaitu Action RPG dan Turn Based RPG. Contoh: *Final Fantasy* dan *The Last Remnant.*

6. Penembak *(Shooter)*

Jenis *game* ini difokuskan terhadap pertempuran yang kebanyakan menggunakan senjata militer seperti rudal, pistol dan lain-lain. *Genre game* shooter ini dibedakan menjadi 2 yaitu First Person Shooter (FPS) dan Third Person Shooter (TPS). First Person Shooter (FPS) menggunakan sudut pandang orang pertama untuk membidik atau membunuh musuh, sehingga yang tampak hanya tangannya saja, tidak

sampai melihat karakter yang dimainkan. Third Person Shooter (TPS) menggunakan sudit pandang orang ketiga, sehingga kita bisa melihat seluruh tubuh karakter yang dimainkan. Contoh: *Call of Duty (FPS)* dan *Gears of Wars (TPS).*

7. Simulasi *(Simulation)*

Genre tipe ini merupakan tipe *game* yang memberikan pengalaman atau interaksi sedekat mungkin dengan kendaraan yang aslinya. Segala faktor yang ada pada *game* ini sangat diperhatikan agar mirip dengan di dunia nyata. *Game* ini terbilang agak rumit karena di buat berdasarkan objek asli yang di simulasikan. Contoh: *Bus Simulator* dan *Flight Simulator*.

8. Olahraga *(Sport)*

Sports game adalah salah satu *genre game* yang di buat dari olahraga yang ada di kehidupan nyata. Dari segala faktor yang ada di dalam olahraga sampai strategi permainan diperhatikan disini. Tokoh dari *game* ini biasanya benar-benar ada di dunia nyata. Tapi ada juga yang merupakan hasil kreasi pembuat *game*. Sistem permainan tiap *game* berbeda tergantung jenis olahraga. Contoh: *Pro Evolution Soccer* dan *FIFA*.

9. Strategi *(Strategy)*

Genre tipe ini mempunyai *gameplay* mengatur suatu unit atau pasukan untuk membangun, menyerang musuh, atau bertahan dan juga diperlukan keahlian berpikir untuk memutuskan setiap gerakan secara hati-hati dan terencana untuk memenangkan permainan. *Game* ini

mengharuskan pemainnya menggunakan taktik dan strategi untuk melihat setiap peluang, kelemahan musuh, dan jeli dalam menggunakan sumber daya yang ada. Contoh: *Red Alert* dan *Stronghold Crusader.*

10. *Tycoon*

Dalam *game tycoon*, keahlian berpikir harus digunakan untuk memainkannya, karena *game* ini pemain akan menjadi seorang pengusaha yang menjalankan bisnis dengan memproduksi barang atau jasa agar laku di pasaran. Tantangan *game* ini adalah bagaimana kita mengolah modal kita untuk membangun sesuatu agar dapat memperoleh keuntungan. Contoh: *Roller Coaster Tycoon* dan *Zoo Tycoon*.

2.1.2. Tashrif

Shorof menurut bahasa adalah berubah atau mengubah. Mengubah dari bentuk aslinya kepada bentuk yang lain. Misalnya merubah bentuk bangunan rumah kuno menjadi bentuk bangunan rumah yang modern. Adapun menurut istilah, *shorof* adalah berubahnya bentuk asal pertama yang berupa *fi'il madhi,* menjadi *fi'il mudhori,* menjadi *mashdar, isim fa'il, isim maf'ul, fi'il amr, fi'il nahi, isim zaman, isim makan sampai isim alat*. Maksud dan tujuan dari perubahan ini adalah agar memperoleh makna atau arti yang berbeda. Dari perubahan satu bentuk ke bentuk lainnya di dalam ilmu *shorof* dinamakaan *shighot*.

Dari hal ini, ilmu yang mempelajari berbagai macam bentuk perubahan kata, asal usul kata atau keadaannya dinamakan dengan ilmu *Shorof*. Perbedaan yang mendasar antara *shorof* dan nahwu secara gampangannya adalah kalau *shorof* untuk membaca kitab atau tulisan yang gundul, sedangkan nahwu untuk

mengetahui makna dari kitab gundul tersebut. Sehingga antara nahwu dan *shorof* tidak boleh dipisahkan dalam penggunaannya.

Dalam ilmu *shorof*, para ulama telah membagi *tashrif* ini menjadi 2 macam, yaitu *Tashrif* Lughowi dan *Tashrif* Istilahi.

1. *Tashrif Lughowi* adalah perubahan kata dari satu bentuk ke bentuk lain dengan pelaku yang berbeda atau dengan kata lain berdasarkan dhomir.
   Contoh: **فعل ـ فعلا ـ فعلو ـ الخ...**
2. *Tashrif Istilahi* adalah perubahan kata yang digunakan untuk mengetahui bentuk *shighot* (bentuk kata) dari suatu kata dari fi'il madhi sampai isim 'alat.
   Contoh: **فعل ـ يفعل ـ فعلا ـ الخ...**

2.1.3. AI (Artificial Intellegences)

Artificial Intelligence atau kecerdasan buatan adalah suatu proses di perangkat lunak maupun perangkat keras buatan manusia yang mempunyai kecerdasan seperti manusia untuk membantu manusia dalam memecahkan suatu masalah yang lebih rumit dalam komputasi digital. Kecerdasan Buatan (AI) merupakan sebuah studi tentang bagaimana membuat komputer melakukan hal-hal yang pada saat ini dapat dilakukan lebih baik oleh manusia (Rich and Knight:1991). Sedangkan menurut H.A. Simon *Artificial Intelligence* atau kecerdasan buatan merupakan kawasan penelitian, aplikasi, dan instruksi yang terkait dengan pemrograman komputer untuk melakukan suatu hal yang dalam pandangan manusia adalah kecerdasan.

Ada 4 pendekatan mengenai *Artificial Intelligence* menurut Stuart Russel dan Peter Norvig pada tahun 2010 yaitu:

1. ***Thinking Humanly****: the cognitive modeling approach*

   Pendekatan ini dilakukan dengan 2 cara sebagai berikut:

   - Melalui introspeksi: mencoba menangkap pemikiran-pemikiran kita sendiri pada saat kita berfikir. Tetap seorang Psikolog barat mengatakan *"How do you know that you understand?"* Bagaimana anda tahu bahwa anda mengerti? Karena pada saat anda menyadari pemikiran anda, ternyata pemikiran tersebut sudah lewat dan digantikan oleh kesadaran anda. Sehingga definisi ini terlalu mengada-ada dan tidak mungkin dilakukan.
   - Melalui eksperimen-eksperimen psikologi.

2. ***Acting Humanly****: the turing test approach*

   Pada tahun 1950, Alan Turing merancang suatu ujian bagi komputer berinterlijensia untuk menguji apakah komputer tersebut mampu mengelabuhi seorang manusia yang menginterograsinya melalui *teletype* (komunikasi berbasis teks jarak jauh). Jika interrogator tidak dapat membedakan yang diinterograsi adalah manusia atau komputer, maka komputer berintelejensia tersebut lolos dari *Turing Test*. Komputer tersebut perlu memiliki kemampuan *Natural Language Processing, Knowledge Representation, Automated Reasoning, Machine Learning, Computer Vision, Robotics. Turing Test* sengaja menghindari kontak fisik antara interrogator dan komputer karena simulasi fisik manusia tidak memerlukan intelijensia.

3. ***Thinking Rationally***: *the laws of thought approach*

   Terdapat dua masalah dalam pendekatan ini, yaitu:

   - Tidak mudah untuk membuat pengetahuan informal dan menyatakan pengetahuan tersebut ke dalam *formal term* yang diperlukan oleh notasi logika, khususnya ketika pengetahuan tersebut memiliki kapasitas kurang dari 100%.
   - Terdapat perbedaan besar antara dapat memecahkan masalah "dalam prinsip" dan memecahkannya "dalam dunia nyata".

4. ***Acting Rationally***: *the rational agent approach*

   Membuat inferensi yang logis merupakan bagian dari suatu rational agent. Hal ini disebabkan satu-satunya cara untuk melakukan aksi secara rasional adalah dengan menalar secara logis. Dengan menalar secara logis, maka bisa didapatkan kesimpulan bahwa aksi yang diberikan akan mencapai tujuan atau tidak. Jika mencapai tujuan, maka agent dapat melakukan aksi berdasarkan kesimpulan tersebut.

Thinking Humanly dan Acting Humanly adalah dua definisi dalam arti yang sangat luas. Sampai saat ini, pemikiran manusia yang diluar rasio, yakni reflek dan intuitif (berhubungan dengan perasaan), belum dapat ditirukan sepenuhnya oleh komputer. Dengan demikian, kedua definisi ini dirasa kurang tepat untuk saat ini. Jika kita menggunakan definisi ini, maka banyak produk komputasi cerdas saat ini yang tidak layak disebut sebagai produk AI.

Definisi *Thinking Rationally* terasa lebih sempit dari pada *Acting Rationally*. Oleh karena itu, definisi AI yang paling tepat untuk saat ini adalah

Acting Rationally dengan pendekatan *Rational Agent*. Hal ini didasarkan pemikiran bahwa komputer bisa melakukan penalaran secara logis dan juga bisa melakukan aksi secara rasional berdasarkan hasil penalaran tersebut.

2.3 Game engine Unity3D

2.3.1 Unity Software

Unity merupakan *game engine* yang dikembangkan oleh *Unity* Technologies. Software ini pertama kali diluncurkan pada tahun 2005 dan menjadi salah satu dari sekian banyak *game engine* yang dipakai oleh banyak pengembang *game* profesional di dunia. *Unity* merupakan alat pengembang *game* dengan kemampuan rendering yang terintegrasi di dalamnya. Dengan mengunakan fitur-fiturnya dan juga kecepatan kerja yang tinggi, *Unity* daat menciptakan sebuah program interaktif tidak hanya dalam 2 dimensi, tetapi dalam bentuk 3 dimensi.

Unity tidak hanya didesain untuk membuat *game* di *Personal Computer* (PC) atau laptop saja, tetapi juga untuk *platform* yang berbeda seperti Android, iOS web*player*, PC, Mac & Linux Standalone, Xbox 360, PS3, dan juga Wii. Oleh karena itu, *Unity* sering disebut *game engine multiplatform* karena bisa digunakan untuk membuat *game* diberbagai macam *platform*.

Tentunya tidak hanya multiplatform yang ditonjolkan di dalam *Unity*, tetapi juga kemampuan untuk membuat berbagai macam permainan dengan *genre* yang berbeda-beda seperti *RPG*, *FPS*, *Adventure*, *Arcade*, *Racing*, *Action,* dan masih banyak lagi. Tentunya *software* ini akan cocok di setiap kalangan pemain *game* yang menyukai *genre game* yang berbeda-beda.

Di balik setiap kacanggihan yang dimiliki *Unity*, ternyata *game engine* ini memiliki tingkat kemudahan yang tidak jauh dari *game engine* yang lain. Tentunya hanya dengan kemauan untuk belajar, kita dapat membuat *game* sendiri sesuai dengan kreativias masing-masing. Yang menarik, untuk membuat *game* kita tidak harus menjadi *programmer* yang handal terlebih dahulu.

Unity memiliki 2 versi yang berbeda, yaitu versi gratis dan pro yang berbayar. Walaupun berbeda jika membandingkan keduanya, versi gratis juga memiliki fitur-fitur yang bagus untuk membuat *game* berkualitas tinggi. Perbedaan untuk versi gratis dan berbayar terlihat di beberapa fitur, diantaranya adalah *AAA visual fidelity, special effects, dan ambience*. *AAA Visual Fidelity* merupakan fitur yang memberikan tampilan yan lebih menarik dari pada versi yang gratis. *Special effects* adalah fitur yang memberikan efek-efek menarik seperti ledakan dan tabrakan. Sedangkan *ambience* adalah fitur yang berkaitan dengan shading aksesbilitas, yang menentukan penampilan *game*.

Unity secara rinci dapat digunakan untuk membuat *video game 3D, real time* animasi *3D* dan visualisasi arsitektur dan isi serupa yang interaktif lainnya. *Editor Unity* dapat menggunakan *plugin* untuk *web player* dan menghasilkan *game browser* yang didukung oleh *Windows* dan *Mac*. *Plugin* web *player* dapat juga dipakai untuk *widgets* Mac. *Unity* juga akan mendukung *console* terbaru seperti *PlayStation* 3 dan *Xbox* 360. Pada tahun 2010, telah memperoleh *Technology Innovation Award* yang diberikan oleh Wall Street Journal dan tahun 2009, *Unity Technology* menjadi 5 perusahaan *game* terbesar. Tahun 2006, menjadi juara dua pada *Apple Design Awards*.

Server aset dari *Unity* dapat digunakan semua *scripts* dan aset *game* sebagai solusi dari versi kontrol dan dapat mendukung proyek yang terdiri atas banyak *gigabytes* dan ribuan dari *file multi-megabyte*. *Editor Unity* dapat menyimpan metadata dan versi mereka, itu dapat berjalan, pembaharuan dan didalam perbandingan versi grafis. *Editor Unity* dapat diperbaharui dengan sesegera mungkin seperti file yang telah dimodifikasi. *Server* aset *Unity* juga cocok pada *Mac, Windows* dan *Linux* dan juga berjalan pada *PostgreSQL, database server OPENsource*

2.3.3. Fitur-fitur

- *Rendering*

  Graphics Engine yang digunakan adalah Direct*3D* (*Windows, Xbox 360), OPENgL (Mac, Windows, Linux, PS3), OPENgL ES (Android, iOS),* dan *proprietary APIs (Wii).* Ada pula kemampuan untuk *bump mapping, reflection mapping, parallax mapping, screen space ambient occlusion (SSAO), dynamic shadows using shadow maps, render-to-texture and full-screen post-processing effects*

  Unity dapat mengambil format desain dari *3Ds Max, Maya, Softimage, Blender, modo, ZBrush, Cinema 4D, Cheetah3D, Adobe Photoshop, Adobe Fireworks and Allegorithmic Substance*. *Asset* tersebut dapat ditambahkan ke *game project* dan diatur melalui *graphical user interface Unity*.

ShaderLab adalah bahasa yang digunakan untuk *shaders*, dimana mampu memberikan deklaratif "programming" dari *fixed-function pipeline* dan program *shader* ditulis dalam GLSL atau Cg. Sebuah *shader* dapat menyertakan banyak varian dan sebuah spesifikasi *fallback declarative*, dimana membuat *Unity* dapat mendeteksi berbagai macam *video card* terbaik saat ini, dan jika tidak ada yang kompatibel, maka akan dilempar menggunakan *shader* alternatif yang mungkin dapat menurunkan fitur dan performa.

Pada 3 Agustus 2013, seiring dengan diluncurkannya versi 4.2, *Unity* mengijinkan *developer* indie mengunakan *Realtime shadows* hanya untuk *Directional lights*, dan juga menambahkan kemampuan dari DirectX11 yang memberikan *shadows* dengan resolusi pixel yang lebih sempurna, textur untuk membuat objek *3D* dari *grayscale* dengan lebih grafik facial, animasi yang lebih halus dan mempercepat FPS.

- *Scripting*

Script game engine dibuat dengan Mono 2.6, sebuah implementasi *open-source* dari .NET *Frame*work. *Programmer* dapat menggunakan *UnityScript* (bahasa terkustomisasi yang terinspirasi dari sintax ECMA*Script*, dalam bentuk *JavaScript), C#*, atau Boo (terinspirasi dari sintax bahasa pemrograman phyton). Dimulai dengan dirilisnya versi 3.0, *Unity* menyertakan versi Mo*node*velop yang terkustomisasi untuk debug *script*.

- *Asset Tracking*

Unity juga menyertakan *Server Unity Asset* – sebuah solusi terkontrol untuk defeloper *game asset* dan *script*. *Server* tersebut menggunakan *PostgreSQL* sebagai *backend*, sistem audio dibuat menggunakan FMOD library (dengan kemampuan untuk memutar *Ogg Vorbis compressed audio*), *video playback* menggunakan *Theora codec*, *engine* daratan dan vegetasi (dimana mensuport *tree billboarding*, *Occlusion Culling dengan Umbra*), *built-in lightmapping dan global illumination* dengan *Beast*, *multiplayer networking* menggunakan RakNet, dan navigasi mesh pencari jalur built-in.

- *Platform*s

Unity support pengembangan ke berbagai *platform*. Didalam, *developer* memiliki kontrol untuk mengirim keperangkat mobile, web *browser*, *desktop*, and *console*. *Unity* juga mengijinkan spesifikasi kompresi tekstur dan pengaturan resolusi di setiap *platform* yang didukung.

Saat ini *platform* yang didukung adalah *BlackBerry 10, Windows 8, Windows Phone 8, Windows, Mac, Linux, Android, iOS, Unity Web Player, Adobe Flash, PlayStation 3, Xbox 360, Wii U and Wii*. Meskipun tidak semua terkonfirmasi secara resmi, *Unity* juga mendukung *PlayStation Vita* yang dapat dilihat pada *Game Escape Plan* dan *Oddworld: New 'n' Tasty.*

Rencana *platform* berikutnya adalah *PlayStation* 4 dan *Xbox One*. Dan juga rumor untuk kedepanya mengatakan HTML akan menjadi *platform*nya, dan *plug-in* Adobe baru dimana akan disubtitusikan ke *Flash Player*, juga akan menjadi *platform* berikutnya.

- *Asset Store*

  Diluncurkan November 2010, *Unity Asset Store* adalah sebuah *resource* yang hadir di *Unity editor*. *Asset store* terdiri dari koleksi lebih dari 4,400 *asset packages*, beserta *3D models, textures* dan *materials*, sistem *particle*, musik dan efek suara, tutorial dan *project*, *script*ing *package, editor extensions* dan servis *online*.

- Physics

  Unity juga memiliki *support built-in* untuk *PhysX physics engine* (sejak *Unity* 3.0) dari Nvidia (sebelumnya Ageia) dengan penambahan kemampuan untuk simulasi *real-time cloth* pada *arbitrary* dan *skinned meshes, thick ray cast,* dan *collision layers* (Rosikhana M, Aristiawan, 2013).

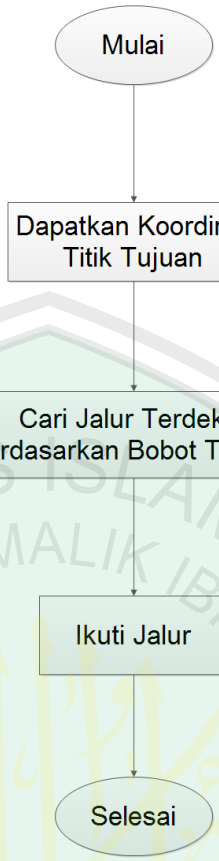
2.4 NPC (Non-Playable Character)

Yunifa Miftahul Arif (2010) dalam penelitiannya yang berjudul *Strategi Menyerang pada Game FPS Menggunakan Hierarchy Finite State Machine dan Logika Fuzzy* menjelaskan bahwa *NPC* adalah Jenis otonomous *agent* yang ditujukan untuk penggunaan komputer animasi dan media interaktif seperti *game* 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 *NPC (A Non-Player Character).*

2.4 Algoritma *Pathfinding*

Algoritma *pathfinding* adalah algoritma untuk menemukan jalur terbaik dari vetex awal ke vertex akhir. *Pathfinding* digunakan untuk menentukan arah pergerakan suatu objek dari suatu tempat ke tempat lain berdasarkan keadaan peta dan objek yang lainnya. Algoritma *pathfinding* memiliki tiga tahapan, mendapatkan koordinat titik tujuan, mencari jalur terdekat berdasarkan bobot terkecil dan membuat jalurnya, dan mengikuti jalur yang sudah tercipta. Peracangan algoritma pathfinding dijabarkan pada gambar dibawah ini:



Gambar 2.1 Flowchart Pencarian

Secara umum algoritma *pathfinding* digolongkan menjadi 2 jenis:

1. Algoritma *Blind/Un-informed Search*

   Pada algoritma ini digunakan isitilah *blind* atau buta karena memang tidak ada informasi awal yang digunakan dalam proses pencarian. Contoh algoritma ini adalah *Breadth-First-Search* (BFS), *Uniform Cost Search* (UCS), *Depth First Search* (DFS), *Depth-Limited Search* (DLS), *Iterative-Deepening Search* (IDS), dan *Bi-directional Search* (BDS).

2. Algoritma Pencarian Heuristik

   Pada algoritma ini, istilah heuristik diartikan sebagai fungsi yang memberikan suatu nilai berupa biaya perkiraan (estimasi) dari suatu solusi. Contoh algoritma ini adalah *Generate and Test*, *Hill Climbing* (*Simple Hill Climbing* dan *Steepest-Ascent Hill Climbing*), *Simulated Annealing*, *Best First Search* (*Greedy Best-First Search* dan A* dengan berbagai variasi).

   Dalam algoritma *pathfinding* sering terjadi *backtrack* bila tidak menemukan solusi. *Backtrack* merupakan suatu algoritma pelacakan yang mencoba mencari penyelesaian masalah yang menyeluruh dengan membangun solusi partial. Masalah yang akan diselesaikan dengan fungsi backtrack harus memenuhi suatu set kendala (*constraint*). Dalam prosesnya, backtrack akan mundur ke solusi partial sebelumnya, jika terdapat solusi yang cocok dengan tuntunan masalah.

2.5 Algoritma A*

Algoritma A* merupakan perbaikan dari metode *BFS (Best First Search)* dengan memodifikasi fungsi heuristiknya. Heuristik adalah sebuah teknik yang mengembangkan efisiensi dalam proses pencarian (pencarian yang lebih simpel). Namun kemungkinan juga dapat mengorbankan kelengkapan (*complateness*). Heuristik digunakan untuk mengevaluasi keadaan-keadaan problem individual dan menentukan seberapa jauh hal tersebut dapat digunakan untuk mendapatkan solusi yang diinginkan. Jenis-jenis pencarian heuristik diantaranya yaitu *Generate and Test, Hill Climbing, Best First Search, Alpha Beta Prunning,* dan *Simmulated Annealing.* Algoritma A* akan meminimumkan total biaya lintasan yang terdapat pada metode *BFS (Best First Search*). Pada kondisi tertentu algoritma A* akan

memberikan solusi yang terbaik dalam waktu yang optimal. Pada contoh kasus pencarian rute dimana tidak ada halangan pada peta, algoritma A* akan bekerja secepat dan seefisien metode BFS (*Best First Search*). Algoritma A* juga akan bekerja dengan baik dengan menemukan solusi *rute* meskipun pada peta terdapat halangan tanpa terjebak.

Algoritma A* mempunyai beberapa istilah dasar, diantaranya simpul (*node*), *start node*, *end node*, *current node*, *open list*, *closed list*, harga (*cost*), halangan (*unwalkable*).

- *Node* adalah petak-petak kecil sebagai representasi dari area *pathfinding*. Bentuknya dapat berupa persegi, maupun segitiga.
- *Start node* adalah sebuah terminologi posisi awal sebuah objek.
- *End node* adalah posisi terakhir dari sebuah objek.
- *Current Node* adalah simpul yang sedang dijalankan algoritma pencarian jalur terpendek.
- *Open list* adalah tempat penyimpanan data simpul yang mungkin diakses dari *start node* maupun simpul yang sedang dijalankan.
- *Closed list* adalah tempat penyimpanan data simpul sebelum *current mode* yang juga merupakan bagian dari jalur terpendek yang telah berhasil didapatkan.
- Harga (*cost*) adalah nilai yang diperoleh dari penjumlahan nilai G, jumlah nilai tiap simpul dalam jalur terpendek dari *start node* ke *current node*, dan H, jumlah nilai perkiraan dari sebuah simpul ke simpul tujuan.
- Simpul tujuan adalah simpul yang dituju.

- Halangan (*unwalkable*) adalah sebuah atribut yang menyatakan bahwa simpul tidak dapat dilalui oleh *current mode.*

A star memiliki 2 fungsi utama dalam menentukan solusi terbaik. Fungsi pertama disebut sebagai g(n) merupakan fungsi yang digunakan untuk menghitung total *cost* yang dibutuhkan dari *start*ing point menuju *node* tertentu. Fungsi kedua yang biasa disebut sebagai h(n) merupakan fungsi perkiraan *total cost* yang diperkirakan dari suatu *node* ke *node* akhir.

Pada A star, setiap *node* dari *node* awal ditelusuri kemudian dihitung *cost* dari tiap-tiap *node* dan dimasukkan ke tabel prioritas. *Node* dengan *cost* paling rendah akan diberikan tingkat prioritas paling tinggi. Kemudian pencarian dilanjutkan pada *node* dengan nilai prioritas tertinggi pada tabel. Algoritma A* dapat dijelaskan di bawah ini:

1. Masukkan *node* awal ke *open list.*
2. *Loop* langkah-langkah dibawah ini:
    a. Cari *node* (*n*) dengan nilai f(n) yang paling rendah dalam *open list. Node* ini sekarang menjadi *current list node.*
    b. Keluarkan *current node* dari *open list* dan masukkan ke *closed list*
    c. Untuk setiap tetangga dari *current node* lakukan langkah sebagai berikut:
        - Jika tidak dapat dilalui atau sudah ada dalam *closed list*, maka abaikan.

- Jika belum ada di *open list*, buat *current node parrent* dari *node* tetangga ini. Simpan nilai f, g, dan h dari *node* ini.
- Jika sudah ada di *open list*, cek bila *node* tetangga ini lebih baik dengan menggunakan nilai g sebagai ukuran. Jika lebih ganti *parrent* dari *node* ini di *open list* menjadi *current node*, lalu kalkulasi ulang nilai g dan f dari *node* ini.

d. Hentikan *loop* jika:

- *Node* tujuan telah ditambahkan ke *open list*, yang berarti rute telah ditemukan.
- Belum menemukan *node goal* sementara *open list* kosong atau rute belum ditemukan.

3. Simpan rute secara *backward*, urut mulai *node goal parrent* sampai mencapai *node* awal sambil menyimpan *node* ke dalam *array*.

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

dengan:

n = *posisi koordinat node*

f(n) = *fungsi evaluasi*

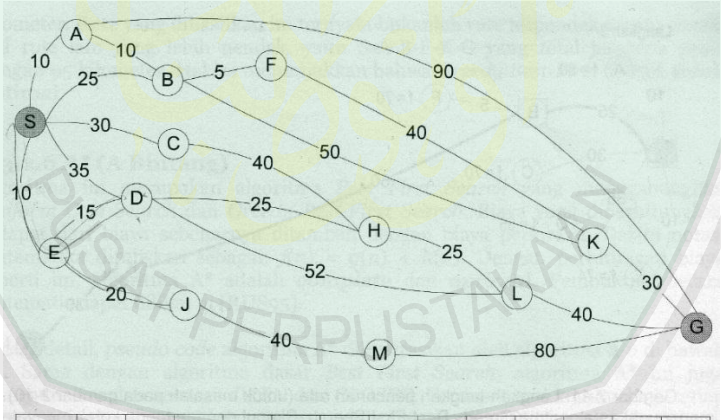
g(n) = *biaya yang sudah dikeluarkan dari keadaan sampai keadaan* ***n***

h(n) = *estimasi biaya untuk sampai pada suatu tujuan mulai dari* ***n***

Nilai F adalah *cost* perkiraan suatu *node* yang teridentifikasi. Nilai F merupakan hasil dari f(n). Nilai G hasil dari fungsi g(n), adalah banyaknya langkah yang diperlukan untuk menuju ke *node* sekarang. Setiap *node* (*node*) harus memiliki informasi nilai h(n), yaitu estimasi harga *node* tersebut dihitung dari *node* tujuan yang hasilnya menjadi nilai H.

Node dengan nilai terendah merupakan solusi terbaik untuk diperiksa pertama kali pada g(n) + h(n). Dengan fungsi heuristik yang memenuhi kondisi tersebut, maka pencarian dengan algoritma A star dapat optimal.

Contoh algoritma A* dapat dicontohkan pada gambar dibawah ini:



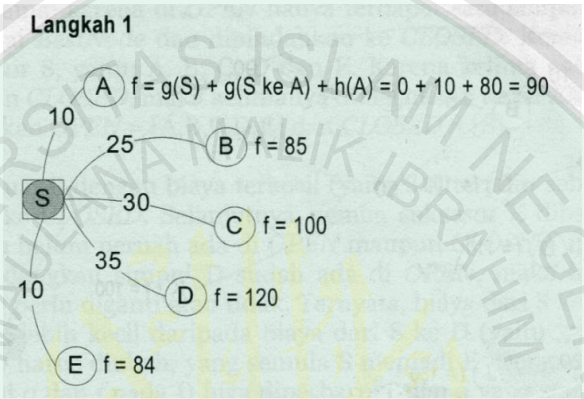
n	S	A	B	C	D	E	F	G	H	J	K	L	M
h(*n*)	80	80	60	70	85	74	70	0	40	100	30	20	70

Gambar 2.2 Algoritma A*

Pada gambar 2.2, masalah pencarian rute dalam suatu daerah yang direpresentasikan dalam suatu graph dua arah. Setiap simpul menyatakan suatu

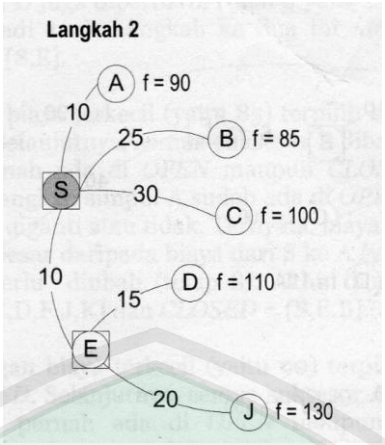
kota. Busur menyatakan jarak sebenarnya antara satu kota dengan kota lainnya dan h(n) menyatakan biaya perkiraan (jarak garis lurus) dari simpul n menuju G. Kemudian mencari rute terpendek dari S menuju G.

Berikut dibawah ini akan dijelaskan langkah-langkah dalam menyelesaikan algoritma A*:



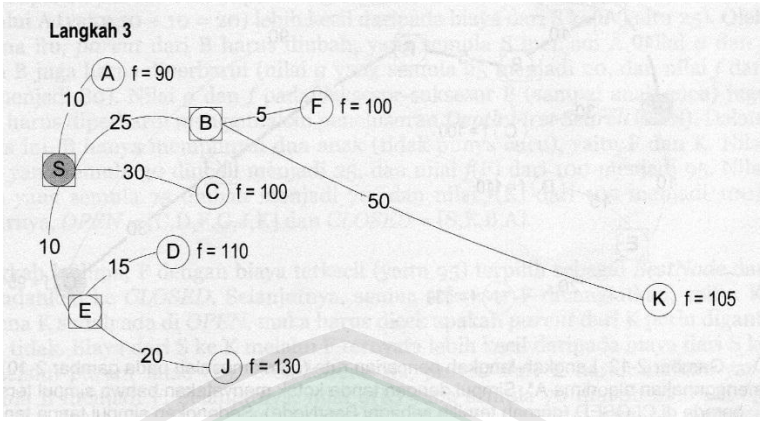
Gambar 2.3 Langkah Pertama

Pada gambar 2.3 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].



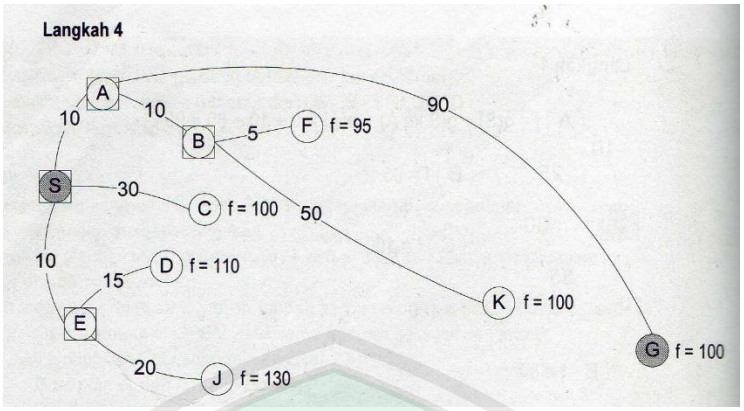
Gambar 2.4 Langkah Kedua

Pada gambar 2.4 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 dari pada biaya dari S ke D yaitu 35. Oleh karena itu, parent dari D harus diubah, yang semula S menjadi E. Dengan perubahan parent ini, maka nilai g dan f pada D juga diperbarui (nilai g semula 35 menjadi 25, dan nilai f dari 120 menjadi 110). Langkah kedua ini menjadi OPEN = [A, B, C, D, J] dan CLOSED = [S, E].



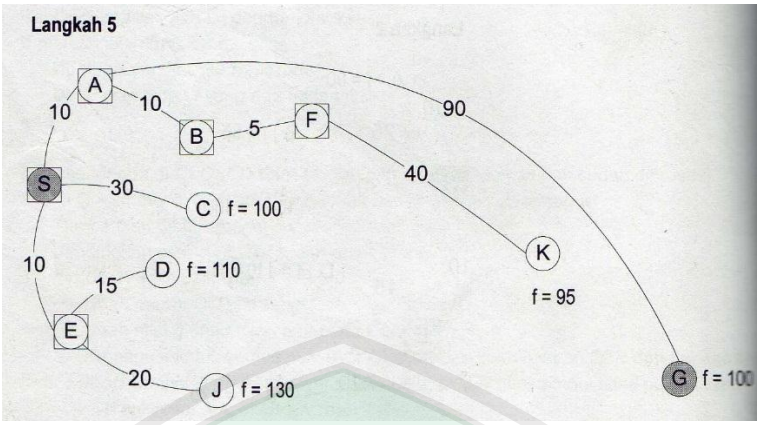
Gambar 2.5 Langkah Ketiga

Pada gambar 2.5 langkah ketiga, B dengan biaya terkecil yaitu 85 terpilih sebagai BestNode dan dipidahkan 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 dari pada biaya dari S ke A yaitu 10. Oleh karena itu, parent dari A tidak perlu diganti (tetap S). Akhir dari langkah ketiga ini menghasilkan OPEN = [A, C, D, F, J, K] dan CLOSED = [S, E, B].



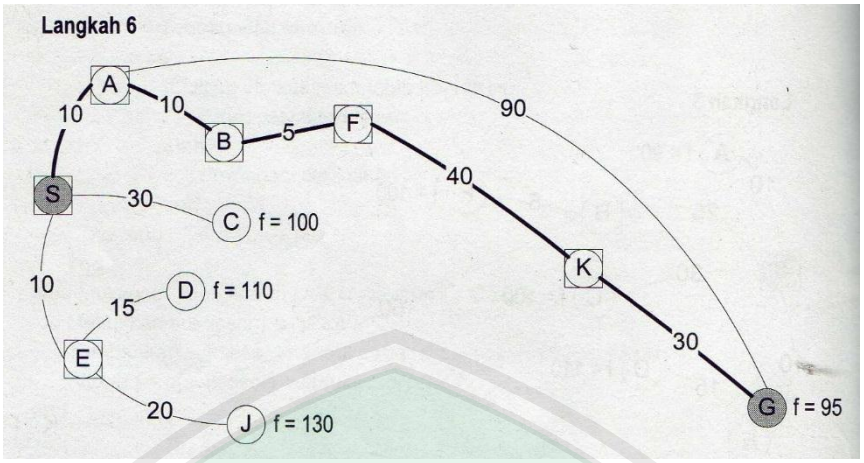
Gambar 2.6 Langkah Keempat

Pada gambar 2.6 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 dan 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 dari pada biaya dari S ke B yaitu 25. Oleh karena itu parent dari B harus diubah, yang semula S menjadi A. Nilai g dan f pada B juga harus diperbarui (nilai g yang semula 25 menjadi 20, dan nilai f dari 85 menjadi 80). Nilai g dan f pada suksesor-suksesor B (sampai anak cucu) juga harus diperbarui menggunakan penelusuran *Depth First Search* (DFS). Dalam kasus ini, B hanya mempunyai dua anak (tidak punya cucu) yaitu F dan K. Nilai g(F) yang semula 30 diubah menjadi 25, dan nilai f(F) dari 100 menjadi 95. Nilai g(K) yang semula 75 diubah menjadi 70, dan nilai f(K) dari 105 menjadi 100. Akhirnya, OPEN = [C, D, F, G, J, K] dan CLOSED = [S, E, B, A].



Gambar 2.7 Langkah Kelima

Pada gambar 2.7 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 di OPEN, maka harus dicek apakah parent dari K perlu diganti atau tidak. Biaya dari S ke K melalui F ternyata lebih kecil dari pada biaya dari S ke K melalui parent lama (B). Oleh karena itu, parent 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. Akhirnya, OPEN = [C, D, F, G, J, K] dan CLOSED = [S, E, B, A, F].



Gambar 2.8 Langkah Keenam

Pada gambar 2.8 langkah keenam, K dengan nilai 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. Biaya dari S ke G melalui K ternyata lebih kecil dari pada biaya dari 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, OPEN = [C, D, G, J] dan CLOSED = [S, E, B, A, F, K].

Selanjutnya, G dengan biaya terkecil yaitu 95 terpilih sebagai BestNode. Karena 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 dan setiap simpul memiliki informasi biaya sebenarnya (g). Penelusuran balik menghasilkan rute S-A-B-F-K dengan total jarak sama dengan 95 kilometer. Rute ini merupakan rute terpendek yang ada di graph tersebut. Jadi, algoritma A* adalah optimal. Tanpa ada batasan waktu dan memori, A* adalah complete (selalu

menemukan solusi jika ada). Pada kasus tersebut, A* membangkitkan dan menyimpan 10 simpul (dari 13 simpul yang ada pada graph). Untuk masalah yang lebih kompleks, misalkan pencarian rute terpendek pada graph yang terdiri dari 100 juta simpul, A* akan menghadapi masalah waktu proses dan memori yang dibutuhkan. Untuk menyelesaikan kedua masalah tersebut, telah diusulkan berbagai variasi A* dengan karakteristik yang sesuai dengan permasalahan tertentu.

2.6 Algoritma Modified Bi-directional A*

Algoritma *Modified Bi-directional A** merupakan salah satu variasi atau modifikasi dari agoritma A* yang dapat digunakan untuk penyelesaian masalah *shortest path* dengan menggunakan fungsi heuristik. Algoritma tersebut mampu menghasilkan performa yang bagus dalam menyelesaikan masalah *shortest path* dibandingkan dengan A* dalam pencarian yang lebih kompleks dan jumlah *node* yang lebih besar (Hutari Laksono, 2012). Fungsi heuristik untuk simpul n pada pencarian maju (dari S ke G) yaitu:

$$f = g(S,n) + \frac{1}{2}\left[h_s(n) - h_g(n)\right]$$

Gambar 2.9 Pencarian Maju dari S ke G

Sedangkan fungsi heuristik untuk simpul n pada pencarian mundur (dari G ke S) yaitu:

$$f = g(G,n) + \frac{1}{2}\left[h_g(n) - h_s(n)\right]$$

Gambar 2.10 Pencarian Maju dari G ke S

Dimana:

S : simpul asal atau *initial state*

G : simpul tujuan atau *goal state*

m : *parent* dari n

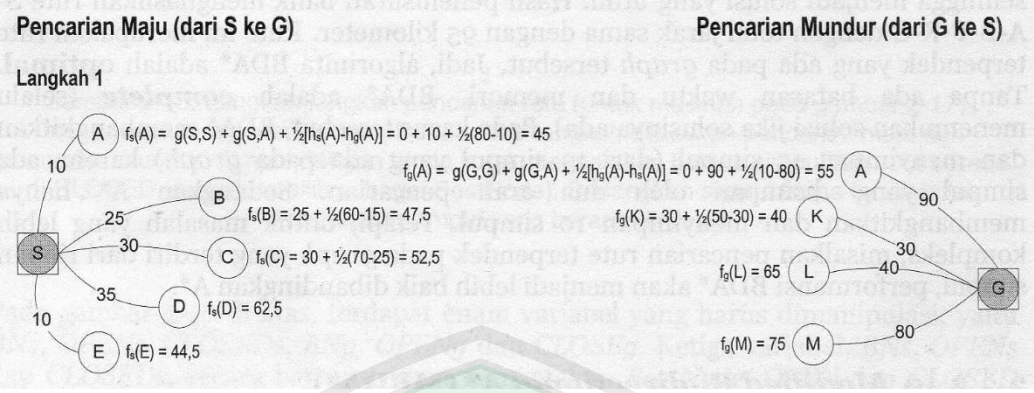
g(S,n) : biaya sebenarnya dari S ke n

g(G,n) : biaya sebenarnya dari G ke n

$h_s(n)$: biaya sebenarnya dari n ke G

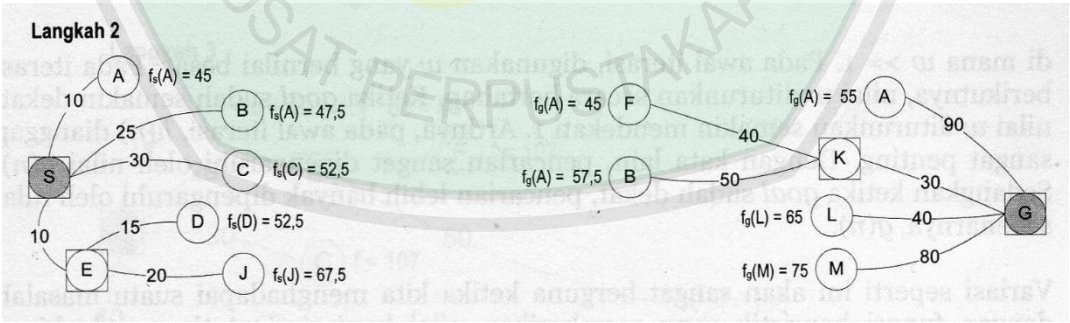
$h_g(n)$: biaya sebenarnya dari n ke S

Di bawah ini adalah langkah-langkah algoritma *Modified Bi-directional A** dalam menyelesaikan masalah:



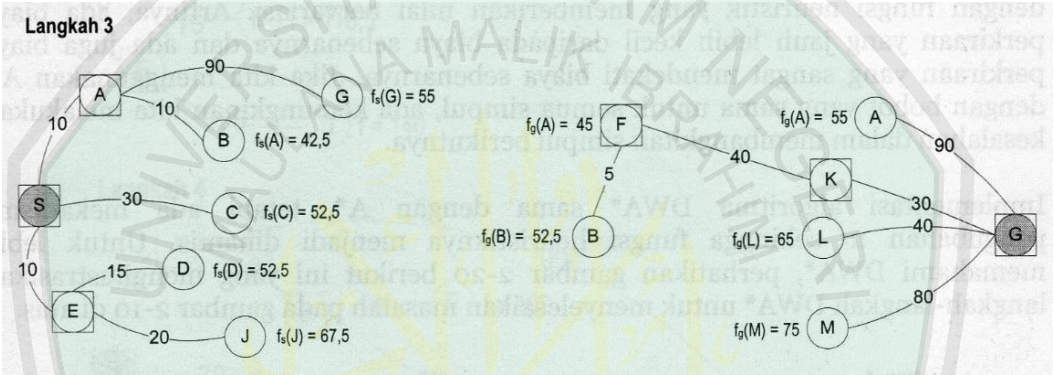
Gambar 2.11 MBDA* Langkah Pertama

Langkah pertama adalah pencarian maju untuk menghasilkan *BNs* = S dan *OPENs* = [A, B, C, D, E] kemudian *CLOSEDs* = S. Karena *best node* tidak berada pada *CLOSEDg*, maka dilanjutkan untuk pencarian mundur yang menghasilkan *BNg* = G dan *OPENg* = [A, K, L, M] kemudian *CLOSEDg* adalah [G]. Karena *BNg* tidak berada pada *CLOSEDs*, maka kembali ke *loop* utama untuk iterasi berikutnya.



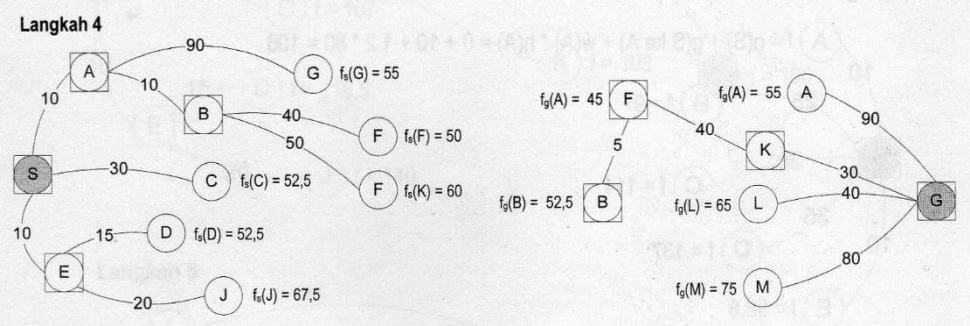
Gambar 2.12 MBDA* Langkah Kedua

Langkah kedua adalah pencarian maju untuk menghasilkan *BNs* = E dan *OPENs* = [A, B, C, D, J]. Kemudian *CLOSEDs* = [S, E]. Karena *BNs* tidak berada pada *CLOSEDg*, maka dilanjutkan pada pencarian mundur yang menghasilkan *BNg* = K dan *OPENg* = [A, L, M, F, B]. Kemudian *CLOSEDg* adalah [G, K]. Karena *BNg* tidak berada pada *CLOSEDs*, maka kembali ke *loop* utama untuk iterasi berikutnya.



Gambar 2.13 MBDA* Langkah Ketiga

Langkah ketiga adalah pencarian maju yang menghasilkan *BNs* = B dan *OPENs* = [A, C, D, F, J, K]. Kemudian untuk *CLOSEDs* = [S, E, B]. Karena *BNs* tidak berada pada *CLOSEDg*, maka dilanjutkan pada pencarian mundur yang menghasilkan *BNg* = F dan *OPENg* adalah [A, L, M, B]. Kemudian untuk *CLOSEDg* = [G, K, F]. Karena *BNg* tidak berada pada *CLOSEDs*, maka kembali ke *loop* utama untuk melakukan iterasi berikutnya.



Gambar 2.14 MBDA* Langkah Keempat

Pada langkah keempat atau langkah yang terakhir, pencarian maju menghasilkan *BNs* = A. OPENs = [C, D, F, J, K, G] dan CLOSEDs = [S, E, B, A]. Karena BNs tidak berada pada CLOSEDg, maka dilanjut pada pencarian mundur yang menghasilkan BNg = B, OPENg = [A, L, M], dan CLOSEDg = [G, K, F, B]. Karena BNg = B berada pada CLOSEDs, berarti solusi telah ditemukan. Kemudian dicari suksesor dari B yang sudah ada di dalam CLOSEDs. A adalah suksesor dari B dan berada pada CLOSEDs. Ternyata g(S, B) melalui A lebih kecil dari pada g(S, B), maka parent dari B diubah, yang tadinya G menjadi A. Nilai g dan f pada B juga diubah. Nilai g yang tadinya 25 diubah menjadi 20 dan nilai f yang tadinya 85 diubah menjadi 80. Selanjutnya rute dan total biaya bisa ditelusuri balik dari G menuju S dan dari S menuju G. Kemudian, hasil penelusuran balik dari kedua arah pencarian digabungkan sehingga menjadi solusi yang utuh. Hasil penelusuran balik menghasilkan rute S-A-B-F-K-G dengan total jarak 95 kilometer. Rute ini merupakan rute terpendek yang ada pada graph tersebut.

2.7 Penelitian Terkait

- *Game Casual Mobile Benthik* / Patil Lele untuk Pembelajaran Ilmu Fiqih Menggunakan Algoritma *Modified Bi-directional A** (*MBDA**) dan Algoritma *Multiplicative Congruential Random Number Generator* (MCRNG)

  Penelitian yang memfokuskan dalam dunia *game console* bernuansa tradisional ini telah berhasil diselesaikan oleh Catur Priyo Wibowo dari Universitas Islam Negeri Maulana Malik Ibrahim Malang. Penelitian tersebut memadukan algoritma *MBDA** dan *MCNRG* sebagai inti dari permainan *benthik*.

- Analisis Performansi Algoritma *MBDA** pada Penyelesaian *Shortest Path* dengan Membagi Graf Menjadi Dua Bagian

  Penelitian ini dibuat oleh Hutari Laksono dari Institut Teknologi Telkom (ITT) Bandung sebagai tugas akhirnya. Pada tugas akhir ini, diterapkan algoritma *MBDA** untuk menyelesaikan masalah pencarian *shortest path* namun dengan menambahkan fungsi lain, yaitu untuk menemukan *middle node*.

  Pencarian solusi akan dibangkitkan dari *start* menuju *middle node* dan dilanjutkan dengan pencarian dari *middle node* menuju *goal node*. Penulis memberi nama algoritma pencarian tersebut dengan *Dgraph-MBDA**. Diharapkan dengan penambahan fungsi tersebut, waktu proses pencarian solusi semakin lebih cepat, dibandingkan dengan *MBDA** yang dilakukan dengan cara biasa.

- Aplikasi Permainan *Meteor Shooter* Menggunakan *MCRNG* dan A* Sebagai Algoritma *Randoming Spawn* dan Pencarian *User* Berbasis *Mobile*.

  Penelitian yang berlatar luar angkasa tersebut diajukan oleh Juniardi Nur Fadila dari Universitas Islam Negeri Maulana Malik Ibrahim Malang sebagai tugas akhir dalam meraih gelar sarjana. Penelitian tersebut menggunakan algoritma *MCRNG* sebagai Algoritma *Randoming Spawn* dan A* sebagai metode pencarian.

BAB III

METODOLOGI PENELITIAN

3.1 Analisi dan perancangan sistem

Game ini adalah *game* yang bergenre *Adventure Game* yang dimainkan secara *single player*. Pada *game* ini terdapat karakter sebagai pemain utama yang akan dimainkan oleh pengguna. *Game* ini terdapat 2 *NPC*, yaitu *NPC Army* sebagai karakter sekutu atau bukan lawan dan karakter *Enemy* yang merupakan karakter lawan yang akan dijalankan secara otomatis oleh komputer sesuai dengan perintah program yang dijalankan. Penerapan algoritma pada *game* ini adalah sebuah algoritma pencarian yang digunakan *NPC Army* untuk mencari dan mendekati objek yang telah ditentukan untuk membantu *player* dalam menyelesaikan misi.

3.1.1 Keterangan Umum *Game*

Game Tashrif "Si Zaid" ini merupakan *game* edukasi bergenre *adventure game* berbasis *dekstop* yang dijadikan media untuk pengenalan bentuk-bentuk perubahan kata pada bahasa arab. Sistem kemenangan pada *game* ini akan ditentukan dengan telah ditemukannya sebuah bintang sebagai tanda kalau semua objek yang dicari telah ditemukan. *Game* ini berisi misi yang akan memandu pemain lebih mengenal perubahan kata pada bahasa arab atau biasa disebut *tashrif* dengan mengumpulkan kata bahasa arab sesuai dengan *wazan*-nya.

Selama permainan ini dimainkan, *player* akan mengumpulkan beberapa kata dalam bahasa arab. Namun ada 2 *NPC* yang terlibat selama permainan berlangsung, yaitu *NPC Army* dan *NPC Enemy*. *NPC Army* adalah *NPC* sekutu

atau *NPC* yang membantu kita selama permainan belangsung. Dalam *game* terdapat 9 kata bahasa arab yang benar sesuai dengan perintah awal untuk menyelesaikan permainan. Tugas dari *NPC Army* ini adalah menemukan kata bahasa arab yang salah secara acak yang telah ditentukan yang tidak sesuai dengan perintah sehingga memudahkan *player* untuk menemukan kata bahasa arab uang benar sesuai perintah. Sedangkan *NPC Enemy* adalah *NPC* musuh yang selalu mengejar *player* kemanapun *player* bergerak. Tugas dari *NPC Enemy* ini adalah mengganggu *player* dalam misi menemukan kata bahasa arab, sehingga pemain diharuskan untuk terus bergerak menghindari *NPC Enemy*. *NPC Enemy* dan nyawa *player* mempunyai jumlah yang sama yaitu 3. Jika *player* terkena *NPC Enemy*, maka nyawa atau kesehatan dari *player* akan berkurang dan juga *NPC Enemy* akan mati atau menghilang dan kemudian untuk tugas mengganggu dan mengejar *player* akan diteruskan oleh *NPC Enemy* yang tersisa yang harus dihindari *player*. Apabila *player* sudah terkena *NPC Enemy* sebanyak 3 kali dalam artian nyawa atau kesehatan *player* sudah habis, maka akan *game* over dan permainan akan kembali dari awal.

Karakter *NPC Army* seperti yang telah dijelaskan diatas adalah karakter *NPC* yang membantu *player* dalam menjalankan misi. Tugasnya adalah mencari dan menemukan kata bahasa arab yang salah untuk memudahkan *player* dalam menemukan kata bahasa arab yang benar. Peletakan *NPC Army* berada di tengah arena permainan dan letak dari kata bahasa arab yang salah yang menjadi tujuan atau tugas dari *NPC Army* ini adalah berada di tempat yang acak dan jauh dari *NPC Army*. Oleh karena itu, pergerakan *NPC Army* ini mengimplementasikan algoritma *Modified Bi-directioanl A** (*MBDA**). Ketika permainan dimulai, setiap

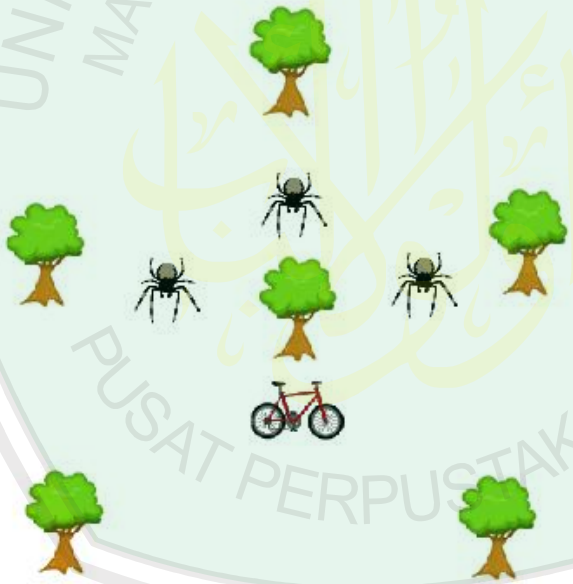
NPC Army akan mencari rute terdekat menuju lokasi kata bahasa arab yang telah diacak. Setelah rute terpendek berhasil ditemukan, maka *NPC Army* akan mulai berjalan dan mengambil kata bahasa arab tersebut dan tugasnya selesai. Kemudian *NPC Army* akan tetap berdiam diri di lokasi ditemukannya lokasi kata bahasa arab yang salah tersebut sebagai tanda untuk *player* kalau tempat tersebut pernah ada kata bahasa arab yang salah yang telah diambil oleh *NPC Army*.

Jika seluruh kata bahasa arab sudah terambil semua oleh *player*, maka objek bintang akan muncul. Seketika itu *player* diharuskan untuk segera mengambil objek bintang tersebut sebagai tanda bahwa *player* telah berhasil menyelesaikan misi pada permainan ini dan *player* dinyatakan menang dan dapat lanjut ke level selanjutnya.

3.1.2 Story board

Berikut ini adalah gambar *storyboard* dari *game*:

No.	Storyboard	Deskripsi
1.		Kondisi awal ketika permainan ini dimulai, *player* siap untuk bermain. Selain *player* terdapat 2 *NPC*, yaitu *NPC Army* (jamur) dan *NPC Enemy* (laba-laba). Kemudian ada kata bahasa arab yang harus diambil *player*.
2.		*Player* harus mencari dan mengumpulkan *tashrif* sesuai dengan kata yang diperintahkan. Pada permainan awal yang harus dikumpulkan *player* adalah kata *tashrif* فعل

3.		Apabila *player* salah mengambil kata, maka poin akan berkurang.
4.		Selama perjalanan *player* mengumpulkan kata *tashrif*, *NPC Enemy* terus mengejar *player* kemanapun *player* bergerak. *NPC Enemy* diperankan oleh laba-laba. *Player* harus berlari menghindar agar tidak terkena *NPC Enemy*. Apabila *player* terkena *NPC Enemy*, maka kesehatan/nyawa dari *player* akan berkurang dan juga *NPC Enemy* akan mati dan menghilang.

5.		Selain terdapat *NPC Enemy*, dalam permainan juga terdapat *NPC Army*. *NPC Army* disini diperankan oleh karakter jamur. *NPC Army* ini bertugas untuk mengambil kata *tashrif* yang salah.
6.		Dengan diambilnya kata *tashrif* yang salah oleh *NPC Army*, maka tugas *player* untuk mengumpulkan kata *tashrif* akan lebih mudah. Pergerakan *NPC Army* untk menuju target menggunakan implementasi dari algoritma *Modified Bi-directional A** (*MBDA**).

7.		Apabila waktu atau nyawa *player* sudah habis, maka *player* dinyatakan kalah atau *game* over.
8.		Ketika kata *tashrif* sudah dikumpulkan, maka akan muncul tanda bintang sebagai tanda bahwa permainan sudah selesai.

9.		Apabila *player* sudah berhasil mengambil bintang, maka *player* telah berhasil menyelesaikan misi permainan ini. *Player* dinyatakan menang dan bisa kembali ke menu utama atau ke level yang selanjutnya.

Tabel 3.1 Storyboard Game

3.1.3 Penampilan Umum *Game*

Game ini dibangun dengan secara umum dibangun menggunakan grafis 3 dimensi sehingga membuat pemain semakin tertarik dan juga membuat permainan ini menjadi lebih bagus. Objek-objek dalam *game* ini yang dibangun menggunakan grafis 3 dimensi diantaranya karakter pemain, pohon, rumah, rumput, gunung, perbukitan, batu, musuh atau *NPC (Non-Playable Character)* dan lain-lain.

3.1.4 Deskripsi Karakter

Berikut ini akan dijelaskan karakter-karakter dalam *game*, yaitu karakter utama adalah Si Zaid dan *NPC (Non-Playable Character)* adalah kancil:

a) Karakter Utama (Si Zaid)

Karakter utama dalam *game* ini adalah Si Zaid. Dalam permainan Si Zaid akan menaiki sepeda. Karakter utama *game* ini menggunakan sudut pandang orang pertama atau *First Person Shooter*. Oleh karena itu yang tampak pada layar hanyalah tangan dan setir sepeda. Pemain mempunyai misi untuk mencari dan menyusun *wazan tashrif* sesuai dengan urutannya. Untuk mencari dan mengambil *wazan tashrif* tersebut, Si Zaid akan menghadapi rintangan dari *NPC*.

Gambar 3.1 Karakter Utama

b) *NPC Army* (Jamur)

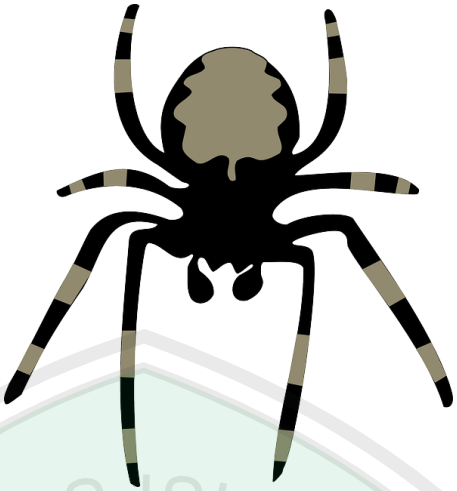
NPC Army adalah karakter yang bertugas mencari dan mengambil kata *tashrif* yang salah. Dengan diambilnya kata *tashrif* yang salah, maka tugas *player* akan semakin mudah untuk menjalankan misi permainan. Pergerakan *NPC Army* ini mengimplementasikan algoritma Modified Bi-directional A* (*MBDA**) untuk mencari rute terdekat dari lokasi awal *NPC Army* ke lokasi kata *tashrif* yang salah. Sebelum *NPC Army* berjalan, algoritma Modified Bi-directional A* akan mencari rute tersebut. Setelah

itu *NPC Army* akan berjalan mengambilnya dengan rute yang sudah ditemukan.

Gambar 3.2 Karakter NPC Army

c) *NPC Enemy* (Laba-laba)

NPC Enemy adalah karakter musuh yang mengganggu *player*. Karakter ini akan terus mengejar *player* kemanapun *player* bergerak. Oleh karena itu pemain harus terus bergerak dan menghindarinya karena apabila *NPC Enemy* mengenai *player* maka kesehatan/nyawa *player* akan berkurang. Apabila terkena 3 kali *player* akan mati dan harus mulai dari awal.



Gambar 3.3 NPC Enemy

3.1.5 Deskripsi Item

Berikut ini akan dijelaskan komponen-komponen item dalam *game* yaitu senjata panah dan item darah:

a) Kata *Tashrif*

Kata *Tashrif* adalah item yang dicari oleh *player*. Kata *Tashrif* ini sendiri objek yang dicari dan diambil oleh *player*. Namun pada permainan ini ada 2 kata kata dasar *tashrif*, yang satu benar dan yang satu salah. *Player* diharuskan untuk mengambil kata *tashrif* yang benar agar dapat menyelesaikan misi. Sedangkan untuk kata *tashrif* yang salah akan dicari dan diambil oleh *NPC Army* yaitu jamur agar misi *player* lebih mudah.

فعل

Gambar 3.4 Kata Shorof

b) Bintang

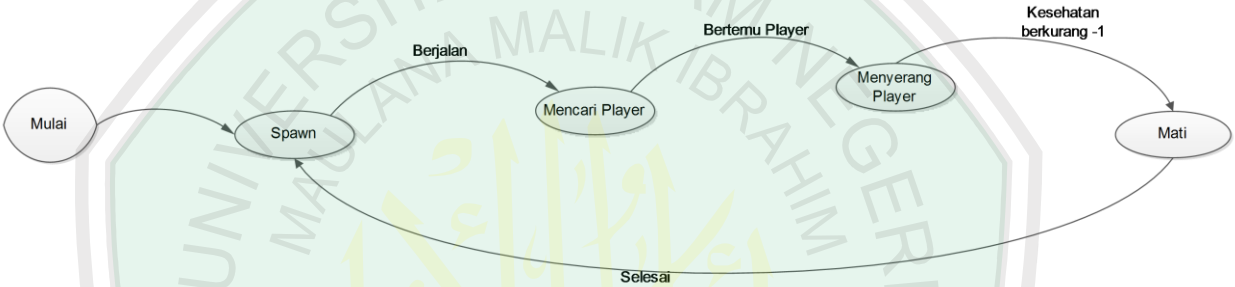
Apabila *player* sudah mengumpulkan semua kata *tashrif* yang sudah ditentukan, maka item bintang ini akan muncul. Item bintang ini menjadi tanda kalau *player* sudah menyelesaikan misi. Oleh karena itu *player* harus mengambil item ini agar bisa lanjut ke level yang selanjutnya.

Gambar 3.5 Item Bintang

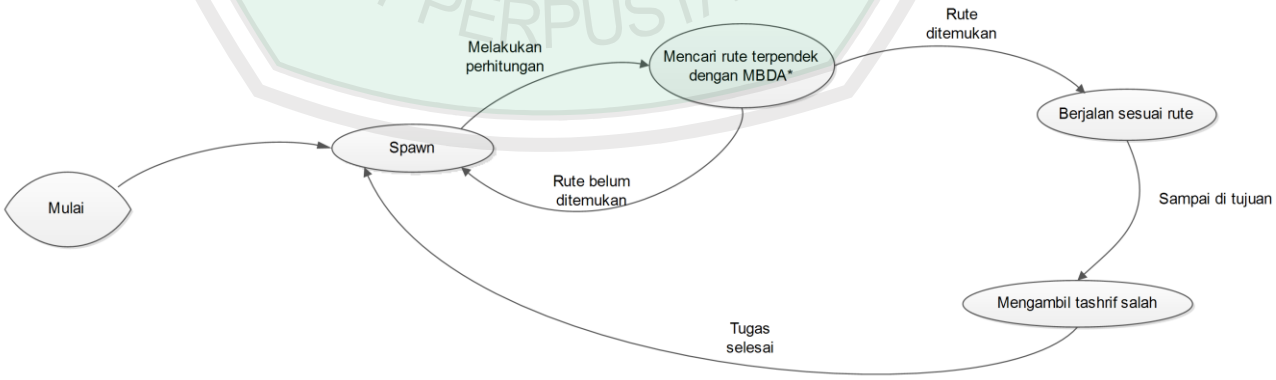
3.2 Finite State Machine

Implementasi *Finite State Machine* pada *game* ini adalah untuk mengatur perilaku *NPC*. Sedangkan perilaku karakter utama pada *game* ini pengikuti

perintah dari yang memainkan permainan ini. Pada permainan ini terdapat 2 buah *NPC*, yaitu *NPC Army* dan *NPC Enemy*. Jumlah *NPC* ini berbeda sesuai dengan ketentuan. Untuk pergerakan *NPC Army* menggunakan implementasi dari algoritma Modified Bi-directional A* (*MBDA**), sedangkan untuk pergerakan *NPC Enemy* berdasarkan perintah program yang dijalankan. Berikut ini adalah FSM untuk kedua *NPC*:



Gambar 3.6 FSM NPC Enemy



Gambar 3.7 FSM NPC Army

Pada gambar 3.7 akan dijelaskan *FSM NPC Enemy*:

1. *Spawn/start*

   Merupakan posisi awal musuh

2. Mencari *Player*

   Musuh berjalan mencari *player*

3. Menyerang *Player*

   Jika jarak sudah dekat dengan pemain, maka musuh akan menyerang pemain.

4. Mati

   Apabila *NPC Enemy* sudah menyerang *player*, maka *NPC Enemy* langsung mati.

Daftar *state transision* pada *FSM Enemy* adalah sebagai berikut:

1. Berjalan
2. Bertemu *player*
3. Kesehatan berkurang -1
4. Selesai

Pada gambar 3.8 akan dijelaskan *NPC Army*:

1. *Spawn*/*start*

   Merupakan posisi awal *NPC Army*.

2. Mencari rute terpendek dengan *MBDA**

   Kemudian itu mencari rute terpedek tujuan dengan algoritma *MBDA**.

3. Berjalan sesuai dengan rute

   Setelah rute berhasil ditemukan, maka *NPC Army* akan berjalan sesuai dengan rute.

4. Mengambil *tashrif* salah

   NPC Army bertugas untuk mengambil *tashrif* yang salah dan untuk memudahkan misi *player*.

   Daftar *state transision* pada *FSM Enemy* adalah sebagai berikut:

1. Melakukan perhitungan
2. Rute belum ditemukan
3. Rute telah ditemukan
4. Sampai di tujuan
5. Tugas selesai

3.3 Perancangan Algoritma Modified Bi-directional A*

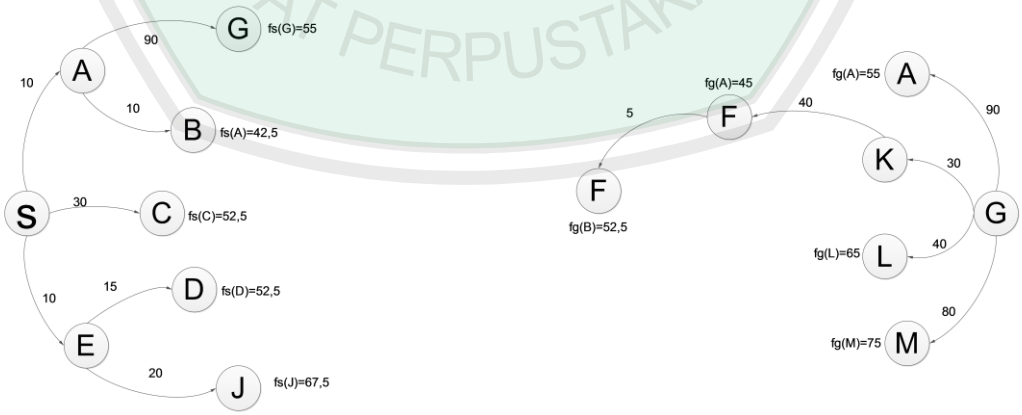
Penghitungan algoritma *Modified Bi-directional A**

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

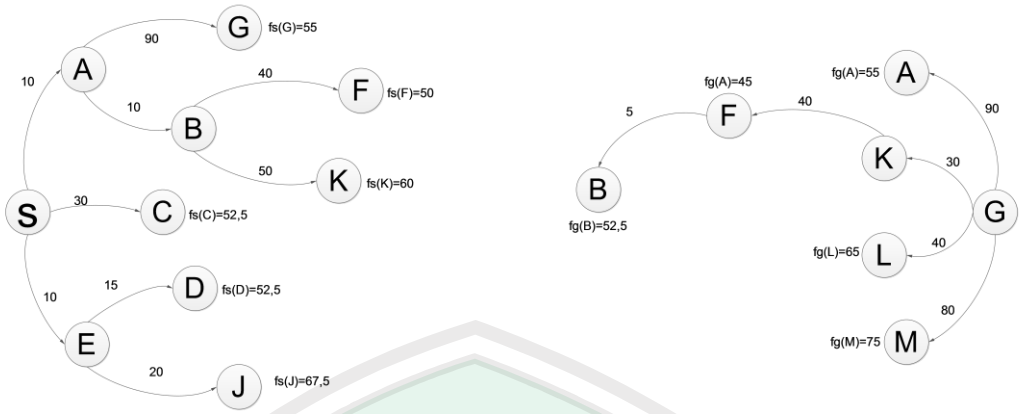
Tabel 3.2 Penghitungan Manual MBDA*

Gambar 3.8 Penghitungan algoritma *MBDA** langkah pertama

Gambar 3.9 Penghitungan algoritma *MBDA** langkah kedua



Gambar 3.10 Penghitungan algoritma *MBDA** langkah ketiga



Gambar 3.11 Penghitungan algoritma *MBDA** langkah keempat

Penghitungan algoritma *Modified Bi-directional A** berdasarkan gambar diatas.

1. fs(A) = g(S,S) + g(S,A) + ½ [hs(A) - hg(A)] = 10 + ½ [80 - 10] = 45

   fs(B) = g(S,S) + g(S,B) + ½ [hs(B) - hg(B)] = 25 + ½ [60 - 15] = 47,5

   fs(C) = g(S,S) + g(S,C) + ½ [hs(C) - hg(C)] = 30 + ½ [70 - 25] = 52,5

   fs(D) = g(S,S) + g(S,D) + ½ [hs(D) - hg(D)] = 35 + ½ [85 - 30] = 62,5

   fs(E) = g(S,S) + g(S,E) + ½ [hs(E) - hg(E)] = 10 + ½ [74 - 5] = 44,5

   fg(A) = g(G,G) + g(G,A) + ½ [hg(A) - hs(A)] = 90 + ½ [10 - 80] = 55

   fg(K) = g(G,G) + g(G,K) + ½ [hg(K) - hs(K)] = 30 + ½ [50 - 30] = 40

   fg(L) = g(G,G) + g(G,L) + ½ [hg(L) - hs(L)] = 40 + ½ [50 - 20] = 65

   fg(M) = g(G,G) + g(G,M) + ½ [hg(M) - hs(M)] = 80 + ½ [60 - 70] = 75

2. **fs(A) = g(S,S) + g(S,A) + ½ [hs(A) - hg(A)] = 10 + ½ [80 - 10] = 45**

   fs(B) = g(S,S) + g(S,B) + ½ [hs(B) - hg(B)] = 25 + ½ [60 - 15] = 47,5

   fs(C) = g(S,S) + g(S,C) + ½ [hs(C) - hg(C)] = 30 + ½ [70 - 25] = 52,5

   fs(D) = g(S,S) + g(S,D) + ½ [hs(D) - hg(D)] = 35 + ½ [85 - 30] = 62,5

fs(J) = g(S,S) + g(S,J) + ½ [hs(J) - hg(J)] = 30 + ½ [100 - 25] = 67,5

fg(A) = g(G,G) + g(G,A) + ½ [hg(A) - hs(A)] = 90 + ½ [10 - 80] = 55

fg(F) = g(G,G) + g(G,F) + ½ [hg(F) - hs(F)] = 70 + ½ [20 - 70] = 45

fg(B) = g(G,G) + g(G,B) + ½ [hg(B) - hs(B)] = 80 + ½ [15 - 60] = 57,5

fg(L) = g(G,G) + g(G,L) + ½ [hg(L) - hs(L)] = 40 + ½ [50 - 20] = 65

fg(M) = g(G,G) + g(G,M) + ½ [hg(M) - hs(M)] = 80 + ½ [60 - 70] = 75

3. fs(G) = g(S,S) + g(S,G) + ½ [hs(G) - hg(G)] = 100 + ½ [0 - 90] = 55

   fs(B) = g(S,S) + g(S,B) + ½ [hs(B) - hg(B)] = 20 + ½ [60 - 15] = 42,5

   fs(C) = g(S,S) + g(S,C) + ½ [hs(C) - hg(C)] = 30 + ½ [70 - 25] = 52,5

   fs(D) = g(S,S) + g(S,D) + ½ [hs(D) - hg(D)] = 25 + ½ [85 - 30] = 52,5

   fs(J) = g(S,S) + g(S,J) + ½ [hs(J) - hg(J)] = 30 + ½ [100 - 25] = 67,5

   fg(A) = g(G,G) + g(G,A) + ½ [hg(A) - hs(A)] = 90 + ½ [10 - 80] = 55

   fg(B) = g(G,G) + g(G,B) + ½ [hg(B) - hs(B)] = 75 + ½ [15 - 60] = 52,5

   fg(L) = g(G,G) + g(G,L) + ½ [hg(L) - hs(L)] = 40 + ½ [50 - 20] = 65

   fg(M) = g(G,G) + g(G,M) + ½ [hg(M) - hs(M)] = 80 + ½ [60 - 70] = 75

4. fs(G) = g(S,S) + g(S,G) + ½ [hs(G) - hg(G)] = 100 + ½ [0 - 90] = 55

   fs(F) = g(S,S) + g(S,B) + ½ [hs(F) - hg(F)] = 60 + ½ [70 - 90] = 50

   fs(K) = g(S,S) + g(S,B) + ½ [hs(K) - hg(K)] = 70 + ½ [30 - 50] = 60

   fs(C) = g(S,S) + g(S,C) + ½ [hs(C) - hg(C)] = 30 + ½ [70 - 25] = 52,5

   fs(D) = g(S,S) + g(S,D) + ½ [hs(D) - hg(D)] = 25 + ½ [85 - 30] = 52,5

   fs(J) = g(S,S) + g(S,J) + ½ [hs(J) - hg(J)] = 30 + ½ [100 - 25] = 67,5

fg(A) = g(G,G) + g(G,A) + ½ [hg(A) - hs(A)] = 90 + ½ [10 - 80] = 55

fg(B) = g(G,G) + g(G,B) + ½ [hg(B) - hs(B)] = 75 + ½ [15 - 60] = 52,5

fg(L) = g(G,G) + g(G,L) + ½ [hg(L) - hs(L)] = 40 + ½ [50 - 20] = 65

fg(M) = g(G,G) + g(G,M) + ½ [hg(M) - hs(M)] = 80 + ½ [60 - 70] = 75

Sebelum mengimplementasikan algoritma ke dalam *game*, akan dilakukan simulasi penghitungan manual algoritma *MBDA**. Berikut contoh arena yang digunakan untuk simulasi perhitungan manual pencarian rute terdekat dengan *MBDA**. Warna hijau adalah *starting point*/*start node*, warna biru adalah *goal*/*end point*/*end node*, dan merah adalah penghalang/*wall*. *Goal* dari aplikasi ini adalah mencari rute dari titik hijau ke biru tanpa melewati penghalang merah.

(0,0)	(0,1)	(0,2)	(0,3)	(0,4)	(0,5)	(0,6)	(0,7)
(1,0)	(1,1)	(1,2)	(1,3)	(1,4)	(1,5)	(1,6)	(1,7)
(2,0)	(2,1)	(2,2)	(2,3)	(2,4)	(2,5)	(2,6)	(2,7)
(3,0)	(3,1)	(3,2)	(3,3)	(3,4)	(3,5)	(3,6)	(3,7)
(4,0)	(4,1)	(4,2)	(4,3)	(4,4)	(4,5)	(4,6)	(4,7)

Tabel 3.3 Arena untuk simulasi *MBDA**

Ketika penghitungan dimulai, warna abu-abu adalah *open list* dan warna kuning adalah *closed list*. Di dalam satu kotak atau *node* ada nilai Hg, Hs, G, S koordinat *node* dan arah direction yang ditunjuk merupakan parent *node* tersebut. Untuk lebih jelasnya akan dijelaskan pada gambar dibawah ini.

Hg	(Koordinat)	**Hg**	(Koordinat)
Hs	**Gs**	**Hs**	**Gs**
ARAH PARENT	**Fs**	ARAH PARENT	**Fs**

Tabel 3.4 Pencarian Maju

Hs	(Koordinat)	**Hs**	(Koordinat)
Hg	**Gg**	**Hg**	**Gg**
ARAH PARENT	**Fg**	ARAH PARENT	**Fg**

Tabel 3.5 Pencarian Mundur

Skor atau biaya disetiap *node* dilambangkan dengan F. Pada algoritma *MBDA** nilai F dapat diperoleh dengan rumus sebagai berikut, fungsi heuristik untuk simpul n pada pencarian maju (dari S ke G) adalah:

$$\mathbf{Fs = Gs + ½ [h_s - h_g]}$$

Sedangkan fungsi heuristik untuk simpul n pada pencarian mundur (dari G ke S) adalah:

$$\mathbf{Fg = Gg + ½ [h_g - h_s]}$$

Di mana:

Gs : jarak dari *start* ke *node* saat ini

Gg : jarak dari *goal* ke *node* saat ini

h_s : jarak *node* saat ini ke *goal*

h_g : jarak dari *node* saat ini ke *start*

Untuk nilai G diasumsikan setiap langkah dari hijau 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 nilai 14 untuk langkah diagonal. Nilai 14 diperoleh dari nilai perhitungan pitagoras dimana 14,1421 = sqrt(sqr(10)+sqr(10)).

Berikut ini adalah simulasi perghitungan manual algoritma *MBDA**. Pada simulasi ini permasalahan utamanya adalah untuk mencari rute terpendek yang dimualai dari koordinat (3,1) yang merupakan *start node* kemudian menuju koordinat (1,6) yang merupakan end *node*/*goal node*. Untuk langkah-langkah penghitungan manual algoritma *MBDA** dan penjelasannya akan dijelaskan pada gambar dibawah ini.

(0,0) 14 70 20 39	(0,1) 10 60 10 35	(0,2) 14 50 20 29	(0,3)	(0,4)	(0,5)	(0,6)	(0,7)
(1,0) 10 60 10 35	(1,1) START	(1,2) 10 40 10 25	(1,3) WALL	(1,4) WALL	(1,5)	(1,6) GOAL	(1,7)
(2,0) 14 70 20 39	(2,1) 10 60 10 35	(2,2) 14 50 20 29	(2,3) WALL	(2,4) WALL	(2,5)	(2,6)	(2,7)
(3,0)	(3,1)	(3,2)	(3,3)	(3,4)	(3,5)	(3,6)	(3,7)

Gambar 3.12 Langkah Pertama Pencarian Maju

Node (0,0)
F = Gs + ½ [Hs-Hg]
F = 14 + ½ [70-20]
F = 39

Node (0,1)
F = Gs + ½ [Hs-Hg]
F = 10 + ½ [60-10]
F = 35

Node (0,1)
F = Gs + ½ [Hs-Hg]
F = 14 + ½ [50-20]
F = 29

Node (1,0)
F = Gs + ½ [Hs-Hg]
F = 10 + ½ [60-10]
F = 35

Node (1,2)
F = Gs + ½ [Hs-Hg]
F = 10 + ½ [40-10]
F = 25

Node (2,0)
F = Gs + ½ [Hs-Hg]
F = 14 + ½ [70-20]
F = 39

Node (2,1)
F = Gs + ½ [Hs-Hg]
F = 10 + ½ [60-10]
F = 35

Node (2,2)
F = Gs + ½ [Hs-Hg]
F = 14 + ½ [50-20]
F = 29

(0,0)	(0,1)	(0,2)	(0,3)	(0,4)	(0,5) 14 50 20 29	(0,6) 10 60 10 35	(0,7) 14 70 20 39
(1,0)	(1,1) START	(1,2)	(1,3) WALL	(1,4) WALL	(1,5) 10 40 10 25	(1,6) GOAL	(1,7) 10 60 10 35
(2,0)	(2,1)	(2,2)	(2,3) WALL	(2,4) WALL	(2,5) 14 50 20 29	(2,6) 10 60 10 35	(2,7) 14 70 20 39
(3,0)	(3,1)	(3,2)	(3,3)	(3,4)	(3,5)	(3,6)	(3,7)

Gambar 3.13 Langkah Pertama Pencarian Mundur

Node (0,5)
F = Gg + ½ [Hg-Hs]
F = 14 + ½ [50-20]
F = 29

Node (0,6)
F = Gg + ½ [Hg-Hs]
F = 10 + ½ [60-10]
F = 35

Node (0,7)
F = Gg + ½ [Hg-Hs]
F = 14 + ½ [70-20]
F = 39

Node (1,5)
F = Gg + ½ [Hg-Hs]
F = 10 + ½ [40-10]
F = 25

Node (1,7)
F = Gg + ½ [Hg-Hs]
F = 10 + ½ [60-10]
F = 35

Node (2,5)
F = Gg + ½ [Hg-Hs]
F = 14 + ½ [50-20]
F = 29

Node (2,6)
F = Gg + ½ [Hg-Hs]
F = 10 + ½ [60-10]
F = 35

Node (2,7)
F = Gg + ½ [Hg-Hs]
F = 14 + ½ [70-20]
F = 39

Langkah pertama adalah pencarian maju untuk menghasilkan *BNs* = *node* (1,1) dan *OPENs* adalah koordinat (0,0), (0,1), (0,2), (1,0), (1,2), (2,0), (2,1), dan (2,2) kemudian *CLOSEDs* adalah (1,1). Karena *best node* tidak berada pada closed G, maka dilanjutkan untuk pencarian mundur yang menghasilkan *BNg* =

(1,6) dan *OPENg* adalah koordinat (0,5), (0,6), (0,7), (1,5), (1,7), (2,5), (2,6), dan (2,7) kemudian *CLOSEDg* adalah (1,6). Karena *BNg* tidak berada pada *CLOSEDs*, maka kembali ke *loop* utama untuk iterasi berikutnya.

(0,0) 14 70 20 39	(0,1) 10 60 10 35	(0,2) 14 50 20 29	(0,3) 24 40 30 29	(0,4)	(0,5)	(0,6)	(0,7)
(1,0) 10 60 10 35	(1,1) START	(1,2) 10 40 10 25	(1,3) WALL	(1,4) WALL	(1,5)	(1,6) GOAL	(1,7)
(2,0) 14 70 20 39	(2,1) 10 60 10 35	(2,2) 14 50 20 29	(2,3) WALL	(2,4) WALL	(2,5)	(2,6)	(2,7)
(3,0)	(3,1)	(3,2)	(3,3)	(3,4)	(3,5)	(3,6)	(3,7)

Gambar 3.14 Langkah Kedua Pencarian Maju

Node (0,1)
F = Gs + ½ [Hs-Hg]
F = 10 + ½ [60-10]
F = 35

Node (0,2)
F = Gs + ½ [Hs-Hg]
F = 14 + ½ [50-20]
F = 29

Node (0,3)
F = Gs + ½ [Hs-Hg]
F = Gs + ½ [40-30]
F = 29

Node (2,1)
F = Gs + ½ [Hs-Hg]
F = 10 + ½ [60-10]
F = 35

Node (2,2)
F = Gs + ½ [Hs-Hg]
F = 14 + ½ [50-20]
F = 29

(0,0)	(0,1)	(0,2)	(0,3)	(0,4) 24 40 30 29	(0,5) 14 50 20 29	(0,6) 10 60 10 35	(0,7) 14 70 20 39
(1,0)	(1,1) START	(1,2)	(1,3) WALL	(1,4) WALL	(1,5) 10 40 10 25	(1,6) GOAL	(1,7) 10 60 10 35
(2,0)	(2,1)	(2,2)	(2,3) WALL	(2,4) WALL	(2,5) 14 50 20 29	(2,6) 10 60 10 35	(2,7) 14 70 20 39
(3,0)	(3,1)	(3,2)	(3,3)	(3,4)	(3,5)	(3,6)	(3,7)

Gambar 3.15 Langkah Kedua Pencarian Mundur

Node (0,4)
F = Gg + ½ [Hg-Hs]
F = 24 + ½ [40-30]
F = 29

Node (0,5)
F = Gg + ½ [Hg-Hs]
F = 14 + ½ [50-20]
F = 29

Node (0,6)
F = Gg + ½ [Hg-Hs]
F = 10 + ½ [60-10]
F = 35

Node (2,5)
F = Gg + ½ [Hg-Hs]
F = 14 + ½ [50-20]
F = 29

Node (2,6)
F = Gg + ½ [Hg-Hs]
F = 10 + ½ [60-10]
F = 35

Langkah kedua adalah pencarian maju untuk menghasilkan *BNs* = *node* (1,2), dan *OPENs* adalah *node* (0,1), (0,2), (0,3), (2,1), dan (2,2). Kemudian *CLOSEDs* adalah *node* (1,1) dan (1,2). Karena *BNs* tidak berada pada *CLOSEDg*, maka dilanjutkan pada pencarian mundur yang menghasilkan *BNg* = *node* (1,5), dan *OPENg* adalah *node* (0,4), (0,5), (0,6), (2,5), dan (2,6). Kemudian *CLOSEDg*

adalah *node* (1,6) dan (1,5). Karena *BNg* tidak berada pada *CLOSEDs*, maka kembali ke *loop* utama untuk iterasi berikutnya.

(0,0) 14 70 20 39	(0,1) 10 60 10 35	(0,2) 14 50 20 29	(0,3) 24 40 30 29	(0,4) 34 30 40 24	(0,5)	(0,6)	(0,7)
(1,0) 10 60 10 35	(1,1) START	(1,2) 10 40 10 25	(1,3) WALL	(1,4) WALL	(1,5)	(1,6) GOAL	(1,7)
(2,0) 14 70 20 39	(2,1) 10 60 10 35	(2,2) 14 50 20 29	(2,3) WALL	(2,4) WALL	(2,5)	(2,6)	(2,7)
(3,0)	(3,1)	(3,2)	(3,3)	(3,4)	(3,5)	(3,6)	(3,7)

Gambar 3.16 Langkah Ketiga Pencarian Maju

Node (0,2)

F = Gs + ½ [Hs-Hg]

F = 14 + ½ [50-20]

F = 29

Node (0,4)

F = Gs + ½ [Hs-Hg]

F = 34 + ½ [30-40]

F = 24

(0,0)	(0,1)	(0,2)	(0,3) 34 30 40 24	(0,4) 24 40 30 29	(0,5) 14 50 20 29	(0,6) 10 60 10 35	(0,7) 14 70 20 39
(1,0)	(1,1) START	(1,2)	(1,3) WALL	(1,4) WALL	(1,5) 10 40 10 25	(1,6) GOAL	(1,7) 10 60 10 35
(2,0)	(2,1)	(2,2)	(2,3) WALL	(2,4) WALL	(2,5) 14 50 20 29	(2,6) 10 60 10 35	(2,7) 14 70 20 39
(3,0)	(3,1)	(3,2)	(3,3)	(3,4)	(3,5)	(3,6)	(3,7)

Gambar 3.17 Langkah Ketiga Pencarian Mundur

Node (0,3)
F = Gg + ½ [Hg-Hs]
F = 34 + ½ [30-40]
F = 24

Node (0,5)
F = Gg + ½ [Hg-Hs]
F = 14 + ½ [50-20]
F = 29

Langkah ketiga adalah pencarian maju yang menghasilkan *BNs* = *node* (0,3) dan *OPENs* adalah *node* (0,2) dan (0,4). Kemudian untuk *CLOSEDs* adalah *node* (1,1), (0,2), dan (0,3). Karena *BNs* tidak berada pada *CLOSEDg*, maka dilanjutkan pada pencarian mundur yang menghasilkan *BNg* = *node* (0,4) dan *OPENg* adalah *node* (0,3) dan (0,5). Kemudian untuk *CLOSEDg* adalah *node* (1,6), (1,5), dan (0,4). Karena *BNg* tidak berada pada *CLOSEDs*, maka kembali ke *loop* utama untuk melakukan iterasi berikutnya.

(0,0) 14 70 20 39	(0,1) 10 60 10 35	(0,2) 14 50 20 29	(0,3) 24 40 30 29	(0,4) 34 30 40 24	(0,5)	(0,6)	(0,7)
(1,0) 10 60 10 35	(1,1) START	(1,2) 10 40 10 25	(1,3) WALL	(1,4) WALL	(1,5)	(1,6) GOAL	(1,7)
(2,0) 14 70 20 39	(2,1) 10 60 10 35	(2,2) 14 50 20 29	(2,3) WALL	(2,4) WALL	(2,5)	(2,6)	(2,7)
(3,0)	(3,1)	(3,2)	(3,3)	(3,4)	(3,5)	(3,6)	(3,7)

Gambar 3.18 Langkah Keempat Pencarian Maju

(0,0)	(0,1)	(0,2)	(0,3) 34 30 40 24	(0,4) 24 40 30 29	(0,5) 14 50 20 29	(0,6) 10 60 10 35	(0,7) 14 70 20 39
(1,0)	(1,1) START	(1,2)	(1,3) WALL	(1,4) WALL	(1,5) 10 40 10 25	(1,6) GOAL	(1,7) 10 60 10 35
(2,0)	(2,1)	(2,2)	(2,3) WALL	(2,4) WALL	(2,5) 14 50 20 29	(2,6) 10 60 10 35	(2,7) 14 70 20 39
(3,0)	(3,1)	(3,2)	(3,3)	(3,4)	(3,5)	(3,6)	(3,7)

Gambar 3.19 Langkah Keempat Pencarian Mundur

Pada langkah keempat atau langkah yang terakhir, pencarian maju menghasilkan *BNs* = *node* (0,4). Karena *BNs* sudah berada pada *CLOSEDg*, maka *goal* sudah ditemukan. Kemudian dicari dari *node* suksesor (0,4) yang sudah ada di *CLOSEDg* sampai kepada *goal*. Selanjutnya rute bisa ditelusuri balik dari G menuju S dan dari S menuju G. Kemudian hasil penelusuran balik dari kedua arah pencarian digabungkan sehingga menjadi solusi yang utuh. Hasil penelusuran balik menghasilkan rute (1,1), (1,2), (0,3), (0,4), (1,5), dan (1,6) sebagai *goal node*. Rute ini merupakan rute terpendek yang ada pada simulasi tersebut.

(0,0)	(0,1)	(0,2)	(0,3)	(0,4)	(0,5)	(0,6)	(0,7)
(1,0)	(1,1)	(1,2)	(1,3)	(1,4)	(1,5)	(1,6)	(1,7)
(2,0)	(2,1)	(2,2)	(2,3)	(2,4)	(2,5)	(2,6)	(2,7)
(3,0)	(3,1)	(3,2)	(3,3)	(3,4)	(3,5)	(3,6)	(3,7)
(4,0)	(4,1)	(4,2)	(4,3)	(4,4)	(4,5)	(4,6)	(4,7)

Tabel 3.6 *Start Node* lurus dengan *Goal Node*

Apabila *start node* berada pada koordinat (2,1) dan *goal node* berada pada koordinat (2,6), maka akan membentuk sebuah garis lurus dan juga simetris antara *start node* dan *goal node*. Setelah ditemukan node (2,2) dan node (2,5) menjadi *CLOSED*, maka akan terjadi nilai F yang sama pada koordinat *OPENs node* (1,2)

dan *node* (3,2) dan *CLOSEDg node* (1,5) dan *node* (3,5). Dengan nilai F yang sama tersebut, maka sistem akan memilih secara *random node* mana yang akan terpilih. Pada akhirnya akan tetap ditemukan rute terdekat antara *start node* dan *goal node.*

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 *game* untuk mengeahui apakah *game* tersebut telah berjalan sesuai dengan tujuan penelitian yang ingin dicapai.

4.1.1 **Kebutuhan** Perangkat Keras

Perangkat keras yang diperlukan untuk mengimplementasikan software dari *game* ini adalah sebagai berikut:

No.	Perangkat Keras	Spesifikasi
1.	*Processor*	*Intel (R) Celeron (R)* 1,70 GHz
2.	RAM	2 GB
3.	VGA	*Intel* (R) *HD Graphics*
4.	HDD	500 GB
5.	Monitor	14’
6.	*Speaker*	*On*
7.	*Mouse & Keyboard*	*On*

Tabel 4.1 Kebutuhan Perangkat Keras

4.1.2 Kebutuhan Perangkat Lunak

Perangkat lunak yang diperlukan untuk mengimplementasikan dari aplikasi *game* ini, sebagai berikut:

No.	Perangkat Lunak	Spesifikasi
1.	Sistem Operasi	*Windows 7 64 bit*
2.	*Game Engine*	*Unity 3D* 4.6.1
3.	Desain *3D*	*Blender 2.74*
4.	*Script Writer*	*Mono Develop*

Tabel 4.2 Kebutuhan Perangkat Lunak

4.1.3 Implementasi Algoritma Modified Bi-directional A* pada Pathfinding NPC

Proses implementasi adalah proses penerapan komponen sistem utama yang dibangun berdasarkan rancangan yang telah dibuat dan diajukan pada bab sebelumnya. Implementasi *Artificial Intelligence* pada penelitian ini diterapkan pada *pathfinding NPC* kepada *player* dengan memanfaatkan metode *Modified Bi-directional A**. Metode *Modified Bi-directional A** digunakan sebagai pembangkit pencarian jarak terpendek *NPC Army* kepada huruf *shorof* yang salah. Posisi *NPC Army* dijadikan sebagai *starting point* dan huruf *shorof* salah diinisiasikan sebagai tujuan. Algoritma *MBDA** ini diimplementasikan menggunakan bahasa C#, method/fungsi yang digunakan adalah sebagai berikut:

No.	Method/fungsi	Keterangan

1.	void Awake() {
	nodeDiameter = nodeRadius*2;
	gridSizeX = Mathf.RoundToInt(gridWorldSize.x/nodeDiameter);
	gridSizeY = Mathf.RoundToInt(gridWorldSize.y/nodeDiameter);
	CreateGrid();
	}
		Membuat Grid

2.	public int MaxSize {
	get {
	return gridSizeX * gridSizeY;
	}
	}
		Membuat ukuran Grid

3.	public bool walkable; public Vector3 worldPosition; public int gridX; public int gridY; public double gCost; public double hCost; public double wCost; public Node parent; public double Hg;	Deklarasi Variabel

4.	public double fCost {
	get {
	return gCost + 0.5*(hCost-Hg);
	}
	}
		Menghitung Pencarian Pencarian Maju

5.	public double fCost {	Menghitung

	get {
	return gCost + 0.5*(Hg-hCost);
	}
	}
		Pencarian Mundur

6.	void Awake() {
	requestManager = GetComponent<PathRequestManager>();
	grid = GetComponent<Grid>();
	}
		Memanggil komponen Grid dan PathRequestManager

7.	Node startNode = grid.NodeFromWorldPoint(startPos); Node targetNode = grid.NodeFromWorldPoint(targetPos);	Mengambil koordinat node awal dan koordinat node tujuan, kemudian memasukkan ke variabel sNode dan tNode
8.	if (startNode.walkable && targetNode.walkable) {	Cek untuk lokasi awal dan lokasi tujuan dapat dilalui

9.	while (openSet.Count > 0) {
	Node currentNode = openSet.RemoveFirst();
	closedSet.Add(currentNode);
		Membuat Best Node yang akan dipindah ke closed

10.	double newMovementCostToNeighbour = currentNode.gCost + GetDistance(currentNode, neighbour);	Menghitung nilai g dari tetangga current node

11	`if (newMovementCostToNeighbour < neighbour.gCost || !openSet.Contains(neighbour)) {`
	
	`neighbour.gCost = newMovementCostToNeighbour;`
	`neighbour.hCost = GetDistance(neighbour, targetNode);`
	
	`neighbour.Hg = GetDistance(neighbour, startNode);`
	`eighbour.parent = currentNode;`
		Jika node tetangga belum ada di open maka hitung nilai g, hs, hg, dan f kemudian set parent node tetangga adalah current node

12.	`if (!openSet.Contains(neighbour))`
	
	`openSet.Add(neighbour);`
		Jika node tetangga belum ada di open maka masukkan node tetangga ke dalam open

13.	`void Start() {` `PathRequestManager.RequestPath (transform.position,target.position, OnPathFound);`	Ketika permainan dimulai terlebih dahulu mengambil lokasi awal, dalam game ini NPC Army dan lokasi tujuan

14.	`public void OnPathFound(Vector3[] newPath, bool pathSuccessful) {`
	`if (pathSuccessful) {`
	`path = newPath;`
		Method yang digunakan untuk menentukan kapan NPC akan

	StopCoroutine("FollowPath"); StartCoroutine("FollowPath"); } }	dijalankan, yaitu jika pathSuccesfull bernilai true atau rute terpendek telah ditemukan, maka mulai jalankan method untuk menjalankan NPC Army

Tabel 4.3 Tabel Method/fungsi

4.1.4 Implementasi Aplikasi *Game*

Aplikasi *game* ini yang telah selesai dibuat dengan *unity 3D* kemudian Implementasi *game* merupakan proses pembangunan komponen-komponen pokok suatu sistem. Komponen tersebut dibangun berdasarkan desain dan rancangan yang telah dibuat sebelumnya.

Gambar 4.1 *Unity Splash screen*

Gambar 4.1 adalah tampilan *unity splash screen* yang menunjukkan bahwa aplikasi *game* ini dibuat menggunakan *game* engine *unity*. Versi yang digunakan untuk membuat *game* ini adalah versi *unity* 4.6.1.

Gambar 4.2 Menu Utama

Gambar 4.2 adalah tampilan menu utama yang terdapat 2 tombol yaitu tombol Main dan tombol Keluar. Tombol Main berfungsi untuk memulai permainan dan tombol Keluar berfungsi untuk menutup aplikasi *game*.

Gambar 4.3 Permainan Dimulai

Gambar 4.3 adalah tampilan ketika permainan dimulai. Di pojok kanan atas terdapat informasi yang harus diketahui *player* yaitu kesehatan dan waktu. Pada permainan ini *player* mempunyai kesehatan 3 dan mempunyai waktu 3 menit atau 180 detik. Pada awal mula permainan, lokasi permainan berada pada hutan, dimana terdapat banyak pohon, batu, dan pegunungan. Tugas dari *player* adalah mencari kata *tashrif* yang sudah ditentukan.

Gambar 4.4 *Player* akan mendapatkan kata *tashrif* yang sesuai

Gambar 4.4 adalah tampilan ketika *player* akan mendapatkan kata *tashrif* yang sesuai dengan ketentuan. Pada permainan kali ini, kata *tashrif* yang harus dikumpulkan adalah فعل yang menjadi kata dasar.

Gambar 4.5 *Player* akan mendapatkan kata *tashrif* yang salah

Gambar 4.5 adalah tampilan dimana *player* akan mendapatkan kata *tashrif* yang salah. Seperti yang diketahui bahwasannya kata yang sudah ditentukan

adalah فعل . Sedangkan kata diatas bukanlah kata yang berasal dari فعل . Ketika pemain mengambil kata *tashrif* yang salah, maka kesehatan *player* akan berkurang 1. Setiap permainan *player* mempunyai kesehatan 3.

Gambar 4.6 NPC Army

Gambar 4.6 adalah tampilan dari *NPC Army*. *NPC Army* ini diperankan oleh karakter jamur yang mempunyai jumlah 7 dan bertugas untuk mengambil kata *tashrif* yang salah. Sehingga tugas dari *player* menjadi lebih mudah untuk mengambil kata tasrif yang telah ditentukan.

Gambar 4.7 *NPC Enemy*

Gambar 4.7 adalah tampilan dari *NPC Enemy*. *NPC Enemy* ini diperankan oleh karakter laba-laba yang mempunyai jumlah 3. *NPC Enemy* ini mempunyai tugas untuk mengejar dan mengganggu *player* kemanapun *player* bergerak. Ketika *NPC Enemy* ini mengenai *player*, maka kesehatan *player* akan berkurang 1 dan juga karakter *NPC Enemy* yang mengenai *player* tersebut akan hilang.

Gambar 4.8 *Player* mengambil bintang

Gambar 4.8 adalah tampilan objek bintang. Ketika *player* sudah menemukan semua kata *tashrif* yang sudah ditentukan, maka objek bintang ini akan muncul sebagai tanda *player* telah menyelesaikan misi dan siap untuk berlanjut untuk level yang selanjutnya.

Gambar 4.9 *Game Over*

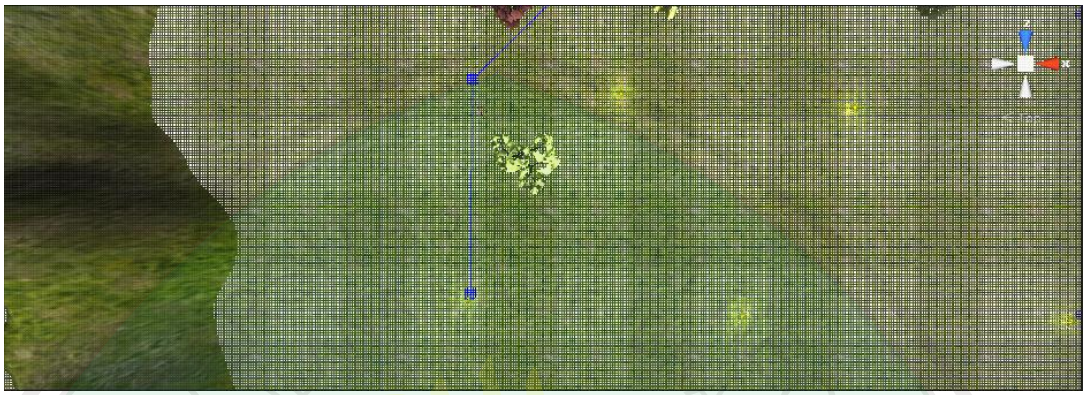
Gambar 4.9 adalah tampilan *game* over. Halaman ini muncul setelah *player* dinyatakan kalah karena kesehatan habis atau karena waktu habis. Halaman *game* over ini akan ditampilkan selama 5 detik, setelah itu akan ditampilkan kembali halaman menu utama agar *player* bisa memulai permainan baru.

4.2 Uji Coba

Pada subbab ini akan membahas tentang uji coba yang telah dilakukan terhadap aplikasi yang telah dibuat. Pembahasan kali ini terdapat 3 uji coba yaitu uji coba algoritma *Modified Bi-directional A** dan uji coba aplikasi game. Berikut ini pembahasan uji coba tersebut.

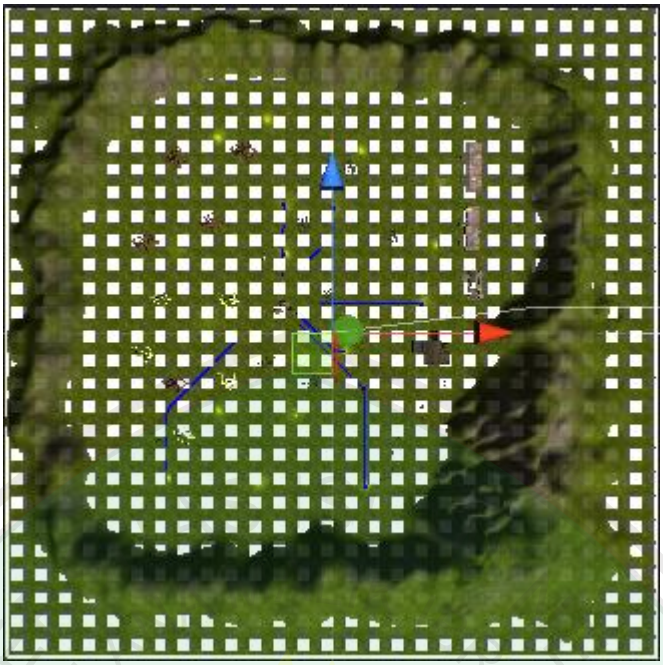
4.2.1 Uji Coba Algoritma Modified Bi-directional A*

Uji coba algoritma Modified Bi-directional A* dilakukan untuk melihat hasil dari implementasi algoritma tersebut kepada game yang telah dibuat di unity 3D.



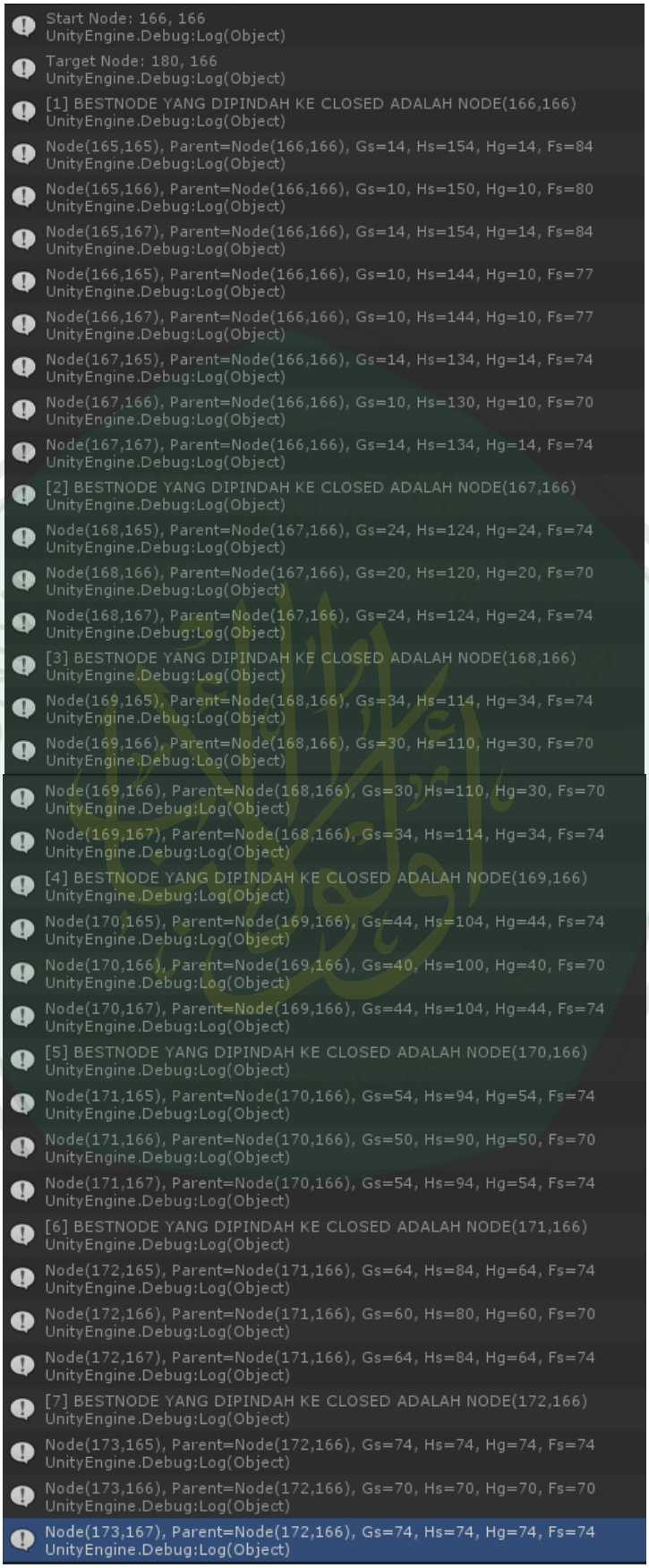
Gambar 4.10 Hasil implementasi Algoritma MBDA* *Grid View* pada salah satu NPC

Uji coba algoritma Modified Bi-directional A* di atas merupakan salah satu contoh pergerakan NPC Army untuk mengambil sorof yang salah. Sedangkan pada aplikasi game yang telah dibuat ini NPC Army akan mengambil beberapa kata shorof yang salah untuk membantu misi player yang ada di arena permainan. Gambar dibawah ini akan ditampilkan semua NPC Army beserta rute terpendek yang akan dilalui untuk mengambil kata sorof yang salah.



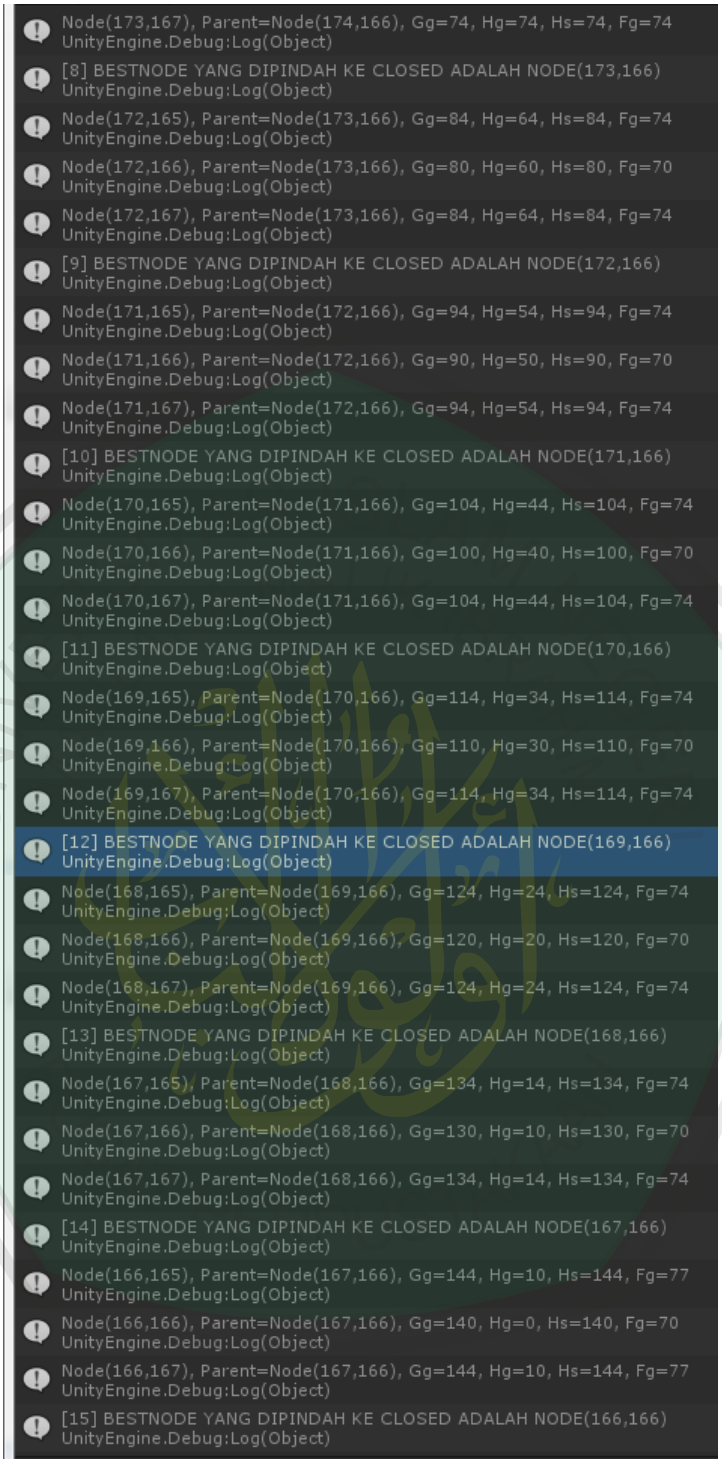
Gambar 4.11 Hasil Implementasi Algoritma MBDA* *Grid View* Semua NPC Army

Di bawah ini akan dilakukan uji coba untuk mengetahui implementasi algoritma Modified Bi-directional A* pada game yang telah dibuat. Penghitungan pencarian maju ini mengambil sampel node start (166, 166) dan goal node (179, 166) yang ditampilkan pada *console unity3D*.


Start Node: 166, 166
UnityEngine.Debug:Log(Object)
Target Node: 180, 166
UnityEngine.Debug:Log(Object)
[1] BESTNODE YANG DIPINDAH KE CLOSED ADALAH NODE(166,166)
UnityEngine.Debug:Log(Object)
Node(165,165), Parent=Node(166,166), Gs=14, Hs=154, Hg=14, Fs=84
UnityEngine.Debug:Log(Object)
Node(165,166), Parent=Node(166,166), Gs=10, Hs=150, Hg=10, Fs=80
UnityEngine.Debug:Log(Object)
Node(165,167), Parent=Node(166,166), Gs=14, Hs=154, Hg=14, Fs=84
UnityEngine.Debug:Log(Object)
Node(166,165), Parent=Node(166,166), Gs=10, Hs=144, Hg=10, Fs=77
UnityEngine.Debug:Log(Object)
Node(166,167), Parent=Node(166,166), Gs=10, Hs=144, Hg=10, Fs=77
UnityEngine.Debug:Log(Object)
Node(167,165), Parent=Node(166,166), Gs=14, Hs=134, Hg=14, Fs=74
UnityEngine.Debug:Log(Object)
Node(167,166), Parent=Node(166,166), Gs=10, Hs=130, Hg=10, Fs=70
UnityEngine.Debug:Log(Object)
Node(167,167), Parent=Node(166,166), Gs=14, Hs=134, Hg=14, Fs=74
UnityEngine.Debug:Log(Object)
[2] BESTNODE YANG DIPINDAH KE CLOSED ADALAH NODE(167,166)
UnityEngine.Debug:Log(Object)
Node(168,165), Parent=Node(167,166), Gs=24, Hs=124, Hg=24, Fs=74
UnityEngine.Debug:Log(Object)
Node(168,166), Parent=Node(167,166), Gs=20, Hs=120, Hg=20, Fs=70
UnityEngine.Debug:Log(Object)
Node(168,167), Parent=Node(167,166), Gs=24, Hs=124, Hg=24, Fs=74
UnityEngine.Debug:Log(Object)
[3] BESTNODE YANG DIPINDAH KE CLOSED ADALAH NODE(168,166)
UnityEngine.Debug:Log(Object)
Node(169,165), Parent=Node(168,166), Gs=34, Hs=114, Hg=34, Fs=74
UnityEngine.Debug:Log(Object)
Node(169,166), Parent=Node(168,166), Gs=30, Hs=110, Hg=30, Fs=70
UnityEngine.Debug:Log(Object)
Node(169,166), Parent=Node(168,166), Gs=30, Hs=110, Hg=30, Fs=70
UnityEngine.Debug:Log(Object)
Node(169,167), Parent=Node(168,166), Gs=34, Hs=114, Hg=34, Fs=74
UnityEngine.Debug:Log(Object)
[4] BESTNODE YANG DIPINDAH KE CLOSED ADALAH NODE(169,166)
UnityEngine.Debug:Log(Object)
Node(170,165), Parent=Node(169,166), Gs=44, Hs=104, Hg=44, Fs=74
UnityEngine.Debug:Log(Object)
Node(170,166), Parent=Node(169,166), Gs=40, Hs=100, Hg=40, Fs=70
UnityEngine.Debug:Log(Object)
Node(170,167), Parent=Node(169,166), Gs=44, Hs=104, Hg=44, Fs=74
UnityEngine.Debug:Log(Object)
[5] BESTNODE YANG DIPINDAH KE CLOSED ADALAH NODE(170,166)
UnityEngine.Debug:Log(Object)
Node(171,165), Parent=Node(170,166), Gs=54, Hs=94, Hg=54, Fs=74
UnityEngine.Debug:Log(Object)
Node(171,166), Parent=Node(170,166), Gs=50, Hs=90, Hg=50, Fs=70
UnityEngine.Debug:Log(Object)
Node(171,167), Parent=Node(170,166), Gs=54, Hs=94, Hg=54, Fs=74
UnityEngine.Debug:Log(Object)
[6] BESTNODE YANG DIPINDAH KE CLOSED ADALAH NODE(171,166)
UnityEngine.Debug:Log(Object)
Node(172,165), Parent=Node(171,166), Gs=64, Hs=84, Hg=64, Fs=74
UnityEngine.Debug:Log(Object)
Node(172,166), Parent=Node(171,166), Gs=60, Hs=80, Hg=60, Fs=70
UnityEngine.Debug:Log(Object)
Node(172,167), Parent=Node(171,166), Gs=64, Hs=84, Hg=64, Fs=74
UnityEngine.Debug:Log(Object)
[7] BESTNODE YANG DIPINDAH KE CLOSED ADALAH NODE(172,166)
UnityEngine.Debug:Log(Object)
Node(173,165), Parent=Node(172,166), Gs=74, Hs=74, Hg=74, Fs=74
UnityEngine.Debug:Log(Object)
Node(173,166), Parent=Node(172,166), Gs=70, Hs=70, Hg=70, Fs=70
UnityEngine.Debug:Log(Object)
Node(173,167), Parent=Node(172,166), Gs=74, Hs=74, Hg=74, Fs=74
UnityEngine.Debug:Log(Object)

Gambar 4.12 Uji Coba Pencarian Maju MBDA* pada *Console Unity3D*


Start Node: 180, 166
UnityEngine.Debug:Log(Object)
Target Node: 166, 166
UnityEngine.Debug:Log(Object)
[1] BESTNODE YANG DIPINDAH KE CLOSED ADALAH NODE(180,166)
UnityEngine.Debug:Log(Object)
Node(179,165), Parent=Node(180,166), Gg=14, Hg=134, Hs=14, Fg=74
UnityEngine.Debug:Log(Object)
Node(179,166), Parent=Node(180,166), Gg=10, Hg=130, Hs=10, Fg=70
UnityEngine.Debug:Log(Object)
Node(179,167), Parent=Node(180,166), Gg=14, Hg=134, Hs=14, Fg=74
UnityEngine.Debug:Log(Object)
Node(180,165), Parent=Node(180,166), Gg=10, Hg=144, Hs=10, Fg=77
UnityEngine.Debug:Log(Object)
Node(180,167), Parent=Node(180,166), Gg=10, Hg=144, Hs=10, Fg=77
UnityEngine.Debug:Log(Object)
Node(181,165), Parent=Node(180,166), Gg=14, Hg=154, Hs=14, Fg=84
UnityEngine.Debug:Log(Object)
Node(181,166), Parent=Node(180,166), Gg=10, Hg=150, Hs=10, Fg=80
UnityEngine.Debug:Log(Object)
Node(181,167), Parent=Node(180,166), Gg=14, Hg=154, Hs=14, Fg=84
UnityEngine.Debug:Log(Object)
[2] BESTNODE YANG DIPINDAH KE CLOSED ADALAH NODE(179,166)
UnityEngine.Debug:Log(Object)
Node(178,165), Parent=Node(179,166), Gg=24, Hg=124, Hs=24, Fg=74
UnityEngine.Debug:Log(Object)
Node(178,166), Parent=Node(179,166), Gg=20, Hg=120, Hs=20, Fg=70
UnityEngine.Debug:Log(Object)
Node(178,167), Parent=Node(179,166), Gg=24, Hg=124, Hs=24, Fg=74
UnityEngine.Debug:Log(Object)
[3] BESTNODE YANG DIPINDAH KE CLOSED ADALAH NODE(178,166)
UnityEngine.Debug:Log(Object)
Node(177,165), Parent=Node(178,166), Gg=34, Hg=114, Hs=34, Fg=74
UnityEngine.Debug:Log(Object)
Node(177,166), Parent=Node(178,166), Gg=30, Hg=110, Hs=30, Fg=70
UnityEngine.Debug:Log(Object)
Node(177,166), Parent=Node(178,166), Gg=30, Hg=110, Hs=30, Fg=70
UnityEngine.Debug:Log(Object)
Node(177,167), Parent=Node(178,166), Gg=34, Hg=114, Hs=34, Fg=74
UnityEngine.Debug:Log(Object)
[4] BESTNODE YANG DIPINDAH KE CLOSED ADALAH NODE(177,166)
UnityEngine.Debug:Log(Object)
Node(176,165), Parent=Node(177,166), Gg=44, Hg=104, Hs=44, Fg=74
UnityEngine.Debug:Log(Object)
Node(176,166), Parent=Node(177,166), Gg=40, Hg=100, Hs=40, Fg=70
UnityEngine.Debug:Log(Object)
Node(176,167), Parent=Node(177,166), Gg=44, Hg=104, Hs=44, Fg=74
UnityEngine.Debug:Log(Object)
[5] BESTNODE YANG DIPINDAH KE CLOSED ADALAH NODE(176,166)
UnityEngine.Debug:Log(Object)
Node(175,165), Parent=Node(176,166), Gg=54, Hg=94, Hs=54, Fg=74
UnityEngine.Debug:Log(Object)
Node(175,166), Parent=Node(176,166), Gg=50, Hg=90, Hs=50, Fg=70
UnityEngine.Debug:Log(Object)
Node(175,167), Parent=Node(176,166), Gg=54, Hg=94, Hs=54, Fg=74
UnityEngine.Debug:Log(Object)
[6] BESTNODE YANG DIPINDAH KE CLOSED ADALAH NODE(175,166)
UnityEngine.Debug:Log(Object)
Node(174,165), Parent=Node(175,166), Gg=64, Hg=84, Hs=64, Fg=74
UnityEngine.Debug:Log(Object)
Node(174,166), Parent=Node(175,166), Gg=60, Hg=80, Hs=60, Fg=70
UnityEngine.Debug:Log(Object)
Node(174,167), Parent=Node(175,166), Gg=64, Hg=84, Hs=64, Fg=74
UnityEngine.Debug:Log(Object)
[7] BESTNODE YANG DIPINDAH KE CLOSED ADALAH NODE(174,166)
UnityEngine.Debug:Log(Object)
Node(173,165), Parent=Node(174,166), Gg=74, Hg=74, Hs=74, Fg=74
UnityEngine.Debug:Log(Object)
Node(173,166), Parent=Node(174,166), Gg=70, Hg=70, Hs=70, Fg=70
UnityEngine.Debug:Log(Object)
Node(173,167), Parent=Node(174,166), Gg=74, Hg=74, Hs=74, Fg=74
UnityEngine.Debug:Log(Object)



Gambar 4.13 Uji Coba Pencarian Mundur MBDA* pada *Console Unity3D*

No.	Node	Gs	Hs	Hg	Fs
1.	(166, 166)	0	140	0	70
2.	(167, 166)	10	130	10	70
3.	(168, 166)	20	120	20	70
4.	(169, 166)	30	110	30	70
5.	(170, 166)	40	100	40	70
6.	(171, 166)	50	90	50	70
7.	(172, 166)	60	80	60	70
8.	(173, 166)	70	70	70	70
9.	(174, 166)	80	60	80	70
10.	(175, 166)	90	50	90	70
11.	(176, 166)	100	40	100	70
12.	(177, 166)	110	30	110	70
13.	(178, 166)	120	20	120	70
14.	(179, 166)	130	10	130	70
15.	(180, 166)	140	0	140	77

Tabel 4.4 Uji Coba Sampel Start Node (166, 166) dan goal node (179, 166)

Sedangkan untuk pencarian mundur akan diambil sampel start node (179, 166) dan goal node (166, 166).

No.	Node	Gs	Hs	Hg	Fs
1.	(180, 166)	140	0	140	77
2.	(179, 166)	130	10	130	74

3.	(178, 166)	120	20	120	70
4.	(177, 166)	110	30	110	70
5.	(176, 166)	100	40	100	70
6.	(175, 166)	90	50	90	70
7.	(174, 166)	80	60	80	70
8.	(173, 166)	70	70	70	70
9.	(172, 166)	60	80	60	70
10.	(171, 166)	50	90	50	70
11.	(170, 166)	40	100	40	70
12.	(169, 166)	30	110	30	70
13.	(168, 166)	20	120	20	70
14.	(167, 166)	10	130	10	70
.15.	(166, 166)	0	140	0	70

Tabel 4.5 Uji Coba Sampel Start Node (179, 166) dan Goal Node (166, 166)

Pada 2 tabel diatas dapat disimpulkan bahwa pencarian maju dan pencarian mundur antara 2 objek yang sudah terdapat algoritma Modified Bi-directional A* bertemu pada node ke 8 (173, 166).

4.2.2 Uji Coba Aplikasi Game

Uji coba pada aplikasi *game* ini dilakukan untuk mengetahui apakah *game* tersebut dapat dimainkan pada personal komputer/PC atau laptop. Berikut adalah hasil dari pengujian yang akan disajikan dalam bentuk table.

No.	Versi OS	Layar	Spesifikasi	RAM	Keterangan
1.	Windows 7	14”	Acer Aspire 4740 Processor Intel (R) Core (TM) 2,13Ghz (4 CPU) VGA Intel HD Graphic Media Accelerator	4 GB	Tampilan menu berjalan dengan baik. Tombol berfungsi dengan baik. Tampilan *game* berjalan dengan baik.
2.	Windows 8	14”	Asus A43SA Processor Intel Core i3-2330M, VGA ATI 6730 (2 GB)	2 GB	Tampilan menu berjalan dengan baik. Tombol berfungsi dengan baik. Tampilan *game* berjalan dengan baik.
3.	Windows 8.1	14”	HP Processor Quadcore 2.1 Ghz VGA Radeon Dual Graphics (4GB)	4 GB	Tampilan menu berjalan dengan baik. Tombol berfungsi dengan baik. Tampilan *game* berjalan dengan baik.
4.	Windows 10	14”	Axioo Neon HNM Prosesor Intel (R) Celeron (R) 1,70	2 GB	Tampilan menu berjalan dengan baik. Tombol berfungsi dengan baik. Tampilan *game* berjalan

			GHz VGA Intel (R) HD Graphics		dengan baik.

Tabel 4.6 Uji Coba Pada Beberapa Laptop/PC

Dari pengujian yang dilakukan sebanyak 4 kali pada berbagai platform windows yang memiliki sistem operasi dan spesifikasi berbeda. Dapat diketahui prosentase pengujian pada tabel di bawah ini.

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 4.7 Presentase Uji Coba Game

Keterangan:

Tabel di atas merupakan tabel yang berisi hasil pengujian *game* terhadap 4 sistem operasi windows desktop yang telah dijelaskan pada Tabel 4.7. Hasil persentase

yang didapatkan dari pengujian adalah **100%** *game* dapat berjalan dengan baik pada *device* tersebut.

4.3 Integrasi dalam Islam

Ilmu agama merupakan hal yang sangat penting bagi setiap Muslim. Dengannya seorang Muslim bisa membedakan mana yang benar dan mana yang salah. Ilmu agama juga akan memberikan tuntunan kepada setiap Muslim kepada jalan yang diridhai oleh Allah dan Rasul-Nya. Ilmu agama yang diamalkan dengan baik tentu akan menjadikan pemiliknya bahagia di dunia dan di akhirat.

Firman-firman Allah Ta'ala dan Sunnah-Sunnah Nabi-Nya shallallahu 'alaihi wa sallam hanya bisa diketahui dengan ilmu. Dan ilmu tersebut tidaklah datang kepada seseorang dengan sendirinya. Seseorang yang ingin memiliki ilmu, maka ia harus pergi mencari dan mempelajarinya dari orang-orang yang berilmu. Dan tentu saja, hal ini membutuhkan pengorbanan dan kesabaran, dan hanya orang-orang yang diberikan petunjuk oleh Allah sajalah yang mau bersabar dan berkorban dalam rangka menuntut ilmu.

Sebagai orang islam kita diwajibkan untuk menuntut ilmu, khususnya ilmu tentang agama islam. Imam Syafii rahimahullah berkata, "Bila kamu tidak tahan lelahnya belajar maka kamu akan menanggung perihnya kebodohan". Dalam surat Al-Mujadilah ayat 11:

يَٰٓأَيُّهَا ٱلَّذِينَ ءَامَنُوٓاْ إِذَا قِيلَ لَكُمۡ تَفَسَّحُواْ فِي ٱلۡمَجَٰلِسِ فَٱفۡسَحُواْ يَفۡسَحِ

ٱللَّهُ لَكُمۡۖ وَإِذَا قِيلَ ٱنشُزُواْ فَٱنشُزُواْ يَرۡفَعِ ٱللَّهُ ٱلَّذِينَ ءَامَنُواْ مِنكُمۡ وَٱلَّذِينَ أُوتُواْ

ٱلۡعِلۡمَ دَرَجَٰتٖۚ وَٱللَّهُ بِمَا تَعۡمَلُونَ خَبِيرٞ ١١

Artinya: *Hai orang-orang beriman apabila dikatakan kepadamu: "Berlapang-lapanglah dalam majlis", maka lapangkanlah niscaya Allah akan memberi kelapangan untukmu. Dan apabila dikatakan: "Berdirilah kamu", maka berdirilah, niscaya Allah akan meninggikan orang-orang yang beriman di antaramu dan orang-orang yang diberi ilmu pengetahuan beberapa derajat. Dan Allah Maha Mengetahui apa yang kamu kerjakan.*

Dalam ayat tersebut sudah jelas bahwa Allah mengangkat derajat orang orang yang berilmu. Sungguh besar derajat orang yang berilmu di mata Allah, karena dengan ilmu tersebut kita sebagai umat islam dapat memahami agama yang haq dan yang batil dengan baik. Ilmu mempunyai beberapa keutamaan, diantaranya adalah dimudahkan oleh Allah menuju surga.

Dari Abu Hurairah radhiyallahu ‘anhu, Rasulullah shallallahu ‘alaihi wa sallam bersabda:

وَمَنْ سَلَكَ طَرِيقًا يَلْتَمِسُ فِيهِ عِلْمًا سَهَّلَ اللهُ لَهُ بِهِ طَرِيقًا إِلَى الْجَنَّةِ

Artinya: *"Dan barangsiapa yang menempuh suatu jalan dalam rangka menuntut ilmu, Allah akan memudahkan baginya jalan menuju surga." (At-Targhib wa At-Tarhib no. 84.)*

Seperti yang kita ketahui bersama bahwa sumber ilmu adalah Al-qur'an dan hadits. Kedua sumber itu adalah tertulis dengan bahasa arab. Oleh karena itu wajib hukumnya bagi kita untuk mempelajari bahasa arab karena dengan mempelajarinya kita bisa memahami Al-qur'an dan hadits.

Bahasa Arab merupakan bahasa yang berasal dan berkembang di kawasan Timur Tengah. Disinilah bahasa arab digunakan dalam berbagai aspek kehidupan. Bukan hanya dikawasan Timur Tengah tetapi bahasa Arab sudah menjadi salah satu dari bahasa internasional. Salah satu alasan urgensi dari mempelajari bahasa Arab adalah karena bahasa arab sebagai simbol agama dan pemersatu umat.

Selain ayat tersebut, dalam ayat surat Asy-Syuara ayat 193 sampai 195 juga dijelaskan pentingnya mempelajari bahasa arab.

نَزَلَ بِهِ ٱلرُّوحُ ٱلۡأَمِينُ ١٩٣ عَلَىٰ قَلۡبِكَ لِتَكُونَ مِنَ ٱلۡمُنذِرِينَ ١٩٤ بِلِسَانٍ عَرَبِيّٖ

مُّبِينٖ ١٩٥

Artinya: *"dia dibawa turun oleh Ar-Ruh Al-Amin (Jibril). ke dalam hatimu (Muhammad) agar kamu menjadi salah seorang di antara orang-orang yang memberi peringatan dengan bahasa Arab yang jelas."*

Tidak perlu diragukan lagi, memang sepantasnya seorang muslim mencintai bahasa Arab dan berusaha menguasainya. Allah telah menjadikan bahasa Arab sebagai bahasa Al-Qur'an karena bahasa Arab adalah bahasa yang terbaik yang pernah ada. Dalam ilmu bahasa arab ada beberapa cabang ilmu, diantaranya *Nahwu, Shorof, Balaghoh, Mantiq, Bayan,* dll. Dalam konteks ini kita mempelajari tentang ilmu *shorof.* Ilmu *shorof* adalah ilmu yang mempelajari kaidah-kaidah perubahan kata, dimana dengan berubahnya kata menjadi perubahan makna kata. Biasanya perubahan kata juga biasa disebut dengan *Tashrif*. Secara garis besar, *shorof* adalah ilmu yang mempelajari tentang *Tashrif*.

Dari pemaparan diatas, dapat disimpulkan bahwa permainan ini dapat digunakan pemain untuk mengingat dan mengetahui akan kosa kata bahasa Arab yang sudah terdapat dalam *game* ini.

BAB V

PENUTUP

5.1 Kesimpulan

Berdasarkan hasil dari implementasi dan pengujian terhadap aplikasi *game* yang telah dibuat oleh peneliti, maka dapat ditarik kesimpulan sebagai berikut:

1. Pembuatan *adventure game* berbasis dekstop sebagai *game* perubahan kata dalam bahasa arab dapat menggunakan *Unity 3D* versi 4.6.1 sebagai *game engine* dan software blender sebagai *asset maker*.
2. Algoritma *MBDA** pada *game tashrif* "Si Zaid" ini berhasil diterapkan sebagai pembangkit pencarian jarak terpendek pada *NPC Army* ke kata *tashrif* yang salah. *NPC Army* berhasil menemukan keberadaan dari kata *tashrif* yang salah di arena permainan, kemudian berjalan untuk mengambil kata *tashrif* yang salah dan bisa melewati halangan yang ada di dalam perjalanan.
3. *Game tashrif* "Si Zaid" telah diuji cobakan pada beberapa komputer dengan sistem operasi dan spesifikasi yang berbeda. Pada berbagai *device manager* tersebut *game* ini menunjukkan tingkat keberhasilan 100% pada uji coba sistem, tampilan, dan tombol.

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:

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. Menambah jumlah level permainan serta aturan untuk kenaikan level sehingga permainan menjadi lebih menarik.
2. Menambah ragam *NPC* dengan perilaku yang bervariasi dan disertai animasi yang menarik
3. Permainan ini tidak hanya disajikan dalam *platform desktop* saja, namun juga bisa dikembangkan pada *platform smartphone* agar pemahaman terkait bahasa arab pada khususnya *shorof* akan semakin diminati.
4. Mengingat *genre* dari *game* ini adalah *game adventure* yang diterapkan sebagai *game* pengenalan perubahan kata dalam bahasa arab, diharapkan dalam pengembangan nantinya *game* ini bisa dinikmati oleh siswa-siswa SD/ MI sampai SMA/ MA dan khususnya para santri pondok pesantren yang belajar tentang *shorof*.

DAFTAR PUSTAKA

Rollings, A., & Adams, E. 2003. *Andrews Rolling and Ernest Adams on Game Design 1st Edition*. California: New Riders.

Costikyan, G. (2013). Uncertainty in Games. *MIT Press*, 20.

Dephne Bavelier, C. S. (2009). Increasing Speed of Processing With Action Video Games. *US National Library of Medicine* , 2.

Kevin Night, E. R. (1991). *Artificial Intelligence (Third Edition).* New York: Tata McGraw-Hill Education Pvt. Ltd.

Stuart Russell, P. N. (2010). *Artificial Intelligence: A Modern Approach (3rd Edition).* New Jersey: Prentice Hall.

Arif, Y. M. (2010). Integrasi Hierarchy Finite State Machine dan Logika Fuzzy Untuk Desain Strategi NPC Game. *Jurnal Pasca Sarjana Teknik Elektro ITS Surabaya*, 1.

McFarlane, P. A. 2003. Use of Computer and Video *Game*s in the Classroom. *Journal Digra*, 2-12.

Zainal, Effendi Farid. 2008. Pengantar Ilmu Nahwu dan *Shorof*. Cepu: PP Assalam

Mufid, Ahmad. 2014. Mudahnya Belajar Ilmu *Shorof*. Jakarta : Buku Pintar

Kusumadewi, S. (2003). *Artificial Intelligence.* Yogyakarta : Penerbit Graha Ilmu.

Wibowo, Catur Priyo. 2014. *Game Casual Mobile Benthik / Patil Lele untuk Pembelajaran Ilmu Fiqih Menggunakan Algoritma Modified Bi-Directional A* (MBDA*) dan Algoritma Multiplicative Congruential Random Number Generator (MCRNG)*. Malang:UIN Maulana Malik Ibrahim Malang.

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

Suyanto. 2011. *Artificial Intelligence.* Bandung : Informatika.

Laksono, Hutari. 2010. *Analisis Performansi Algoritma MBDA* pada Penyelesaian Shortest Path dengan Membagi Graf Menjadi Dua Bagian*. *Tugas Akhir*. Bandung: Institut Teknologi Telkom.

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