RANCANG BANGUN *GAME* PENGENALAN ILMUWAN MUSLIM MENGGUNAKAN ALGORITMA *BEE COLONY* SEBAGAI PEMBANGKIT PERILAKU PENCARIAN NPC TERHADAP TARGET

SKRIPSI

Oleh :
ZAUHAR AFIFUDDIN
NIM. 12650067

JURUSAN TEKNIK INFORMATIKA
FAKULTAS SAINS DAN TEKNOLOGI
UNIVERSITAS ISLAM NEGERIMAULANA MALIK IBRAHIM
MALANG
2019

HALAMAN PENGAJUAN

RANCANG BANGUN *GAME* PENGENALAN ILMUWAN MUSLIM MENGGUNAKAN ALGORITMA *BEE COLONY* SEBAGAI PEMBANGKIT PERILAKU PENCARIAN NPC TERHADAP TARGET

SKRIPSI

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

Oleh :
ZAUHAR AFIFUDDIN
NIM. 12650067

JURUSAN TEKNIK INFORMATIKA
FAKULTAS SAINS DAN TEKNOLOGI
UNIVERSITAS ISLAM NEGERI MAULANA MALIK IBRAHIM MALANG
2019

HALAMAN PERSETUJUAN

RANCANG BANGUN *GAME* PENGENALAN ILMUWAN MUSLIM MENGGUNAKAN ALGORITMA *BEE COLONY* SEBAGAI PEMBANGKIT PERILAKU PENCARIAN NPC TERHADAP TARGET

SKRIPSI

Oleh:
ZAUHAR AFIFUDDIN
NIM. 12650067

Telah Diperiksa dan Disetujui untuk Diuji :
Tanggal : 21 Mei 2018

Pembimbing I	Pembimbing II
Dr. M. Amin Hariyadi, M.T	Yunifa Miftachul Arif, M.T
NIP. 19670118 200501 1 001	NIP. 19830616 201101 1 004

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

Dr. Cahyo Crysdian
NIP. 19740424 200901 1 008

HALAMAN PENGESAHAN

RANCANG BANGUN *GAME* PENGENALAN ILMUWAN MUSLIM MENGGUNAKAN ALGORITMA *BEE COLONY* SEBAGAI PEMBANGKIT PERILAKU PENCARIAN NPC TERHADAP TARGET

SKRIPSI

Oleh :
ZAUHAR AFIFUDDIN
NIM. 12650067

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

Susunan Dewan Penguji		Tanda Tangan
1. Penguji Utama	: Fatchurrohman, M.Kom NIP. 19700731 200501 1 002	( )
2. Ketua Penguji	: Roro Inda Melani, M.T., M.Sc NIP. 19780925 200501 2 008	( )
3. Sekretaris Penguji	: Dr. M. Amin Hariyadi, M.T NIP. 19670118 200501 1 001	( )
4. Anggota Penguji	: Yunifa Miftachul Arif, M.T NIP. 19830616 201101 1 004	( )

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

Dr. Cahyo Crysdian
NIP. 19740424 200901 1 008

HALAMAN PERNYATAAN
ORISINALITAS PENELITIAN

Nama : Zauhar Afifuddin
NIM : 12650067
Jurusan : Teknik Informatika
Fakultas : Sains dan Teknologi
Judul Skripsi : Rancang Bangun *Game* Pengenalan Ilmuwan Muslim Menggunakan Algoritma *Bee Colony* Sebagai Pembangkit Perilaku Pencarian NPC Terhadap Target.

Menyatakan dengan sebenarnya bahwa skripsi yang saya tulis ini benar-benar merupakan hasil karya saya sendiri, bukan merupakan pengambil alihan data, tulisan, atau pikiran orang lain yang saya akui sebagai hasil tulisan atau pikiran saya sendiri, kecuali dengan mencantumkan sumber cuplikan pada daftar pustaka. Apabila dikemudian hari terbukti atau dapat dibuktikan skripsi ini hasil jiplakan, maka saya bersedia menerima sanksi atas perbuatan tersebut.

Malang, 2019
Yang membuat pernyataan,

Zauhar Afifuddin
NIM. 12650067

HALAMAN MOTTO

Lebih Baik Mencoba dan Gagal

Daripada Tidak Pernah Sama Sekali.

HALAMAN PERSEMBAHAN

Bismillahirrohmanirrohim, kupersembahkan sebuah
karya sederhanaku ini untuk orang-orang yang
paling kusayangi, kubanggakan, dan selalu
memberikan energi semangat untukku..

Seluruh keluarga besarku
Khususnya Bapak dan Ibu tercinta
A. Latif dan Muslimah
yang selalu ikhlas mendoakan putra-putrinya
yang selalu mengarahkan kami dalam kebaikan
yang dengan sabar membimbing kami.
Semoga Allah SWT melindungi dan menjaga mereka dalam
naungannya...
Aamiin

KATA PENGANTAR

بِسْمِ اللَّهِ الرَّحْمَٰنِ الرَّحِيمِ

Segala puji bagi Allah SWT yang Maha Pengasih lagi Maha Penyayang atas Rahmat dan Hidayah-Nya sehingga penulis dapat menyelesaikan penyusunan skripsi ini. Sholawat serta Salam tetap tercurahkan kepada junjungan kita, kekasih Allah, Nabi Muhammad SAW, sang pemberi syafaat kelak di hari akhir, beserta seluruh keluarga, sahabat, dan para pengikutnya.

Penelitian skripsi yang berjudul **"Rancang Bangun *Game* Pengenalan Ilmuwan Muslim Menggunakan Algoritma *Bee Colony* Sebagai Pembangkit Perilaku Pencarian NPC Terhadap Target"** ini ditulis untuk memnuhi salah satu syarat guna memperoleh gelar Sarja Strata Satu (S1) Fakultas Sains dan Teknologi Universitas Maulana Malik Ibrahim Malang. Karya penelitian skripsi ini tidak akan pernah ada tanpa bantuan baik moral maupun spiritual dari berbagai pihak yang telah terlibat. Untuk itu dengan segala kerendahan hati, penulis mengucapkan rasa terimakasih yang sebesar-besarnya kepada:

1. Prof. Dr. Abdul Haris, M.Ag, selaku rektor Universitas Islam Negeri Maulana Malik Ibrahim Malang.
2. Bapak Dr. M. Amin Hariyadi, M.T , selaku Dosen Pembimbing I yang telah bersedia meluangkan waktu, tenaga dan pikiran untuk memberikan bimbingan, berbagai pengalaman, arahan, nasihat, motivasi dan pengarahan dalam pembangunan program hingga penyusunan skripsi ini.
3. Bapak Yunifa Miftachul Arif, M.T, selaku dosen pembimbing 2 yang selalu memberi masukan, serta pengarahan dalam penyusunan laporan skripsi ini.
4. Bapak Dr. Muhammad Faisal, S.Kom., M.T, selaku dosen wali yang juga selalu memberi pengarahan terkait akademik selama masa study.
5. Dr. Cahyo Crysdian selaku ketua jurusan Teknik Informatika yang mendukung dan mengarahkan skripsi ini.
6. Segenap civitas akademika Fakultas Saintek, Universitas Islam Negeri Maulana Malik Ibrahim Malang terutama seluruh dosen, terimakasih atas segala ilmu dan bimbingannya.

7. Bapak, Ibu dan seluruh keluarga yang selalu memberikan doa, kasih sayang, semangat, dukungan moril, serta motivasi sampai saat ini, terimakasih banyak.
8. Teman-teman angkatan 2012, yang berjuang bersama-sama untuk meraih mimpi, terimakasih atas kenang-kenangan indah yang dirajut bersama.
9. Semua pihak yang tidak dapat penulis sebutkan satu-persatu atas bantuan, masukan, dukungan serta motivasi kepada penulis.

Harapan penulis semoga semua amal kebaikan dan jasa-jasa dari semua pihak yang telah membantu hingga skripsi ini selesai diterima oleh Allah SWT, serta mendapatkan balasan yang lebih baik dan berlipat ganda.

Penulis juga menyadari bahwa skripsi ini masih jauh dari kesempurnaan yang disebabkan keterbatasan Harapan penulis, semoga karya ini bermanfaat dan menambah ilmu pengetahuan bagi kita semua, Aamiin.

Malang, 2019

Penulis

Zauhar Afifuddin

DAFTAR ISI

DAFTAR GAMBAR

DAFTAR TABEL

ABSTRAK

Afifuddin, Zauhar. 2018. **Rancang Bangun *Game* Pengenalan Ilmuwan Muslim Menggunakan Algoritma *Bee Colony* Sebagai Pembangkit Perilaku Pencarian Npc Terhadap Target**. Skripsi. Jurusan Teknik Informatika Fakultas Sains dan Teknologi Universitas Islam Negeri Maulana Malik Ibrahim Malang.

Pembimbing : (I) Dr. Ir. M. Amin Hariyadi, M.T, (II) Yunifa Miftachul Arif, M.T.

Kata Kunci : *Game*, *Non-Player Character* (NPC), *Pathfinding*, *Bee Colony*.

Game merupakan media hiburan yang paling banyak diminati. Penelitian ini membahas tentang bagaimana pergerakan *Non-Player Character* (NPC) dalam sebuah *game* dapat bergerak menuju Player. Pergerakan tersebut berjalan secara *autonomous* tanpa campur tangan *user*. NPC dapat bergerak menuju *player* dengan cara memilih posisi baru secara acak. Pergerakan itu meniru pergerakan lebah ketika mencari sumber makanan. Oleh karena itu peneliti menggunakan algoritma *bee colony* untuk diterapkan kepada NPC sebagai algoritma *pathfinding* atau pencarian jalur terpendek menuju player (target). Pemilihan jalur terpendek dilakukan dengan membandingkan *fitness* poisisi sebelum dan posisi yang telah dicari secara acak. Algoritma *Bee Colony* diterapkan pada 2 kelompok *agent* dengan masing-masing kelompok berjumlah 3 *agent*. Berdasarkan pengujian algoritma pada kelompok *agent* pertama, keakuratan menunjukan sebesar 97% pada *agent* 1, dan 2. Sedangkan *agent* 3 menunjukan ke akuratan sebesar 98%. NPC (*agent*) dapat memilih posisi mana yang mendekati *player* hingga mencapai posisi yang sangat dekat dengan *player*.

ABSTRACT

Afifuddin, Zauhar. 2018. **Game Design of Introduction to Muslim Scientists Using *Bee Colony* Algorithm to Generate Search Behavior of NPC against Targets**. Thesis. Department of Informatics Engineering Faculty of Science and Technology UIN Maulana Malik Ibrahim Malang.

Advisor : (I) Dr. Ir. M. Amin Hariyadi, M.T, (II) Yunifa Miftachul Arif, M.T.

Keyword : Game, *Non-Player Character* (NPC), *Pathfinding*, *Bee Colony*.

A game is the most well-known entertainment media. This study discusses how the movement of *Non-Player Character* (NPC) in a game can move towards the Player. The movement runs autonomously without user intervention as well as NPC can move towards the player by randomly selecting new positions. The movement imitates the movement of bees while looking for a food source. Therefore the researchers started to use the *Bee Colony* algorithm to be applied to the NPC as a *pathfinding* algorithm or search for the shortest path to the player (target). The selection of the shortest path is conducted by comparing the fitness position before and the position that has been searched randomly. The *Bee Colony* algorithm is applied to 2 groups of agent with each group contains 3 agents. Thus, based on the results of this algorithm trials in first group, reveal the accuracy of 97% in agent 1 and 2, whereas, agent 3 shows the accuracy of 98%. The NPC (agent) is able to opt which position approaches the player until it reaches the closest position to the player.

ملخص

زوهر عفيف الدين. **تصميم لعبة مقدمة من العلماء المسلمين باستخدام خوارزمية مستعمرة النحل لمنهض سلوك البحث عن شخصية غير لاعب ضد الأهداف**. البحث. قسم صناعة المعلوماتية بكلية العلوم والتكنولوجيا بالجامعة الإسلامية الحكومية في مولانا مالك إبراهيم مالانج.

مشرف البحث : د. محمد امين هريادي, يونيتا مفتاح العريف

الكلمة الرئيسية : اللعبة ، الشخصية غير لاعب ، مستعمرة النحل.

اللعبة هي وسائل التسلية الأكثر الراغب فيها. هذا البحث يتحدث عن كيفية حركة شخصية غير لاعب في احدى لعبة و يمكن ان تصير اللاعب. الحركة تعمل بطريقة مستقلة دون تدخل المستخدم. الشخصية غير لاعب تتحرك نحو اللاعب بطريق اختيار الموقع الجديد دون النظام. هذه الحركة كحركة النحل عند البحث عن مصادر الطعام. لذا استخدام الباحث خوارزمية مستعمرة النحل لتطبيقها على شخصية غير لاعب مثل خوارزمية الأستطلاع أو البحث عن أقصر خطط للاعب (الهدف). يتم اختيار أقصر خطط بقيام مقارنة بين الموقف من قبله و الموقف الذي بُحثت دون النظام. تطبيق خوارزمية مستعمرة النحل على ٥ وكلاء. من نتائج هذه التجربة الخوارزمية تظهر الدقة ٩٧٪ في الوكيل ١ و ٢ اما الوكيل ٣ تظهر الدقة ٩٨٪. تستطيع شخصية غير لاعب (الوكيل) اختيار الموقف الذي يقترب من اللاعب حتى تصل إلى أقرب الموضع مع اللاعب.

BAB I

PENDAHULUAN

1.1. Latar Belakang

Islam merupakan agama yang sangat memperhatikan segala aspek kehidupan. Segalanya telah diatur sesuai dengan perintah Allah SWT. Hal demikian diatur agar kita senantiasa menjalani kehidupan dengan teratur, baik dan bermanfaat. Aspek yang cukup diperhatikan dalam Islam adalah ilmu. Sebagaimana telah disabdakan oleh Rasulullah SAW dalam haditsnya :

طَلَبُ الْعِلْمِ فَرِيضَةٌ عَلَى كُلِّ مُسْلِمٍ

"*Menuntut ilmu wajib atas setiap muslim*". (HR. Ibnu Majah)

Ilmu juga berkaitan dengan perkembangan teknologi. Hingga saat ini ilmu pengetahuan dan teknologi telah berkembang pesat. Perkembangan IPTEK tersebut didominasi kuat oleh peradaban barat. Pendiri *New Foundation Of Peace* menyatakan bahwa hal tersebut merupakan "sejarah yang hilang". Sejarawan Jack Goody lebih jauh lagi menyebutnya sebagai "pencurian sejarah". Seakan-akan ingatan atas suatu peradaban dan sumbangsih ilmuwan muslim terhadap khazanah pengetahuan telah dihapus dari kesadaran manusia. Terutama di dunia islam, keberhasilan para ilmuwan muslim telah dilupakan atau setidaknya diabaikan (Ehsaan Masood, 2009).

Sebagai agama yang sangat memperhatikan ilmu pengetahuan Islam membuktikan dengan melahirkan tokoh-tokoh ilmuwan muslim yang bahkan penemuannya menjadi rujukan ilmu pengetahuan dunia. Oleh karena itu sebagai generasi muslim sudah seharusnya kita mengenal kembali ilmuwan-ilmuwan tersebut yang bersumbangsih dalam kemajuan ilmu pengetahuan hingga saat ini.

Ibnu Al-Haytham, orang barat mengenalnya dengan nama *Alhazen*. Dia menemukan kamera pertama setelah melihat cahaya yang masuk melalui lubang jendela. *Al-Jazari* bisa dibilang sebagai salah satu pionir penemuan robot. Sebagai kepala insinyur di Istana Artuklu, Turki, pada abad ke-11 dan 12, Al-Jazari menciptakan berbagai alat otomatis yang sangat maju untuk jamannya (Syaifuddin Al Indunisi, 2013). *Al-Battani* sebagai ahli Astronomi dan bapak Trigonometri (*Izzah Annisa, 2018*). *Ibnu Sina* yang memiliki nama lengkap *Abu 'Ali al-Husayn bin 'Abdullah bin Sina*, beliau dikenal sebagai bapak Kedokteran. Selain itu beliau juga adalah seorang fisikawan juga filosof. *Jabir bin Hayyan* atau Ibnu Hayyan merupakan bapak Kimia modern. Orang-orang barat mengenalnya dengan nama Geber (Salman Iskandar, 2007). *Muhammad Al-Karaji* dianggap sebagai orang yang merintis terciptanya mesin air. Beliau adalah seorang ahli matematikan sekaligus ahli mesin. *Abbas Ibn Firnas* pernah membuat alat untuk terbang dengan mengenakan jubah yang diikatkan pada rangka kayu. Dia melompat dari menara Masjid Agung Kordoba, Spanyol pada 852 M. Meskipun usahanya tidak berhasil, namu alatnya mampu memperlambat tubuhnya jatuh ke bawah. Itulah usaha manusia dalam menciptakan alat yang bisa terbang ke angkasa. (Ridwan Abqary, 2010).

Di atas merupakan contoh beberapa ilmuwan muslim yang dapat diutarakan oleh peneliti. Salah satu perkembangan iptek di era sekarang adalah *game*. *Game* merupakan media hiburan di semua kalangan umur, terutama anak-anak. Menurut Tom Watson, salah satu Menteri Sekretaris Kabinet Inggris, bahwa anak-anak akan mendapatkan pelajaran berharga dari video *game* ketimbang menonton televisi. Dengan bermain *video game* anak-anak dapat belajar melatih pikiran, konsetrasi, menjawab tantangan, dan beradaptasi terhadap perubahan sekitar mereka (Derisna Aditya, 2014). Hal tersebut menjadikan *game* bukan hanya sebagai media hiburan melainkan menjadi salah satu terobosan baru dalam dunia pendidikan. Dengan demikian peneliti bermaksud untuk merancang sebuah *game* yang mampu mengenalkan kembali ilmuwan-ilmuwan muslim terdahulu yang merupakan cikal bakal dari perkembangan ilmu pengetahuan saat ini.

Mengacu pada penelitian sebelumnya yakni penelitian Heru S. Djamaludin (2017), menerapkan algoritma *Artificial Bee Colony* dan algoritma *Boids* pada simulasi thawaf sebagai metode dalam perilaku NPC agar bergerak fleksibel sesuai dengan keadaan lingkungan, menjaga agar tidak saling bertabrakan antar jamaah dan pencarian target sehingga NPC bisa mengelilingi ka'bah dengan sempurna.

Dalam penelitian ini, peneliti bermaksud untuk menerapkan algoritma *Artificial Bee Colony* sebagai metode pencarian jalur terpendek NPC menuju ke target (*player*). Pencarian jalur terpendek dilakukan tanpa memperhitungkan hambatan (*obstacle*) yang ada di dalam *game*. Peneliti juga akan menghitung keakuratan metode ABC yang diterapkan pada NPC.

1.2. Rumusan Masalah

Adapun rumusan masalahnya dari penelitian ini yaitu : Seberapa akurat penerapan algoritma *Bee Colony* terhadap NPC untuk pencarian jalur terpendek menuju target?

1.3. Batasan Masalah

Untuk memfokuskan penelitian ini, terdapat beberapa batasan masalah yaitu sebagai berikut :

a. Posisi target dinamis.
b. NPC bergerak pada bidang 2 dimensi.
c. *Game* ini ditujukan kepada siswa SD kelas V.

1.4. Tujuan Penelitian

Tujuan dari penelitan ini adalah membuktikan ke akuratan algoritma *Bee Colony* yang diterapkan ke dalam NPC yang bergerak menuju target.

1.5. Manfaat Penelitian

Manfaat pembuatan *game* ini adalah untuk mengenali kembali tokoh-tokoh ilmuwan muslim yang merupakan cikal bakal dari kemajuan ilmu pengetahuan dan teknologi di masa sekarang.

BAB II

TINJAUAN PUSTAKA

2.1 Penelitian Terkait

Heru Santoso Djamaludin (2016) merancang sebuah simulasi Thawaf dengan menerapkan algoritma *Boids* dan *Artificial Bee Colony* terhadap NPC jemaah agar berperilaku dinamis yakni mempunyai perilaku yang fleksibel sesuai dengan keadaan lingkungan. Dari pengujian telah menunjukan hasil sebesar 100% terhadap pengujian 150 jemaah dalam keadaan bergerombol untuk mengelilingi Ka'bah (Thawaf). Sedangkan hasil untuk rata-rata waktu yang diperlukan dalam melakukan satu putaran simulasi akan meningkat dengan semakin meningkatnya jumlah jemaah. Dengan jumlah awal 15 jemaah kemudian ditingkatkan menjadi 150 jemaah, maka waktu yang diperlukan mengalami peningkatan sebesar 32.09%.

Wilujeng Jatiningsih, Eko Mulyanto Yuniarno, dan Mochamad Hariadi (2014), mengimplementasikan algoritma *Bee Colony* tergadap NPC agar dapat menyerang target dengan pola *swarm*. Terdapat empat fungsi umum yang digunakan untuk mencari jumlah siklus yang dibutuhkan lebah untuk menuju titik temu. Empat fungsi tersebut adalah : ukuran populasi = 125, jangkauan parameter mulai dari -100 hingga +100, jumlah siklus dibatasi 50 kali, batas pengabaian = 10 dan posisi sumber makanan pada (0,0). Hasil yang diperoleh menunjukkan

lebah menuju titik temu pada siklus 0-25 dan hanya membutuhkan waktu kurang dari 30 menit serta hanya menggunakan 30 iterasi.

Pada penelitian Rakhmad Fajar Nugroho dan Mewati Ayub (2013), mereka berhasil membuat sistem penjadwalan pelajaran untuk SMP dengan menerapkan algoritma *Artificial Bee Colony* sehingga mampu mengurangi jumlah bentrokan antara jadwal kegiatan guru dan periode waktu dalam mengajar. Algoritma ABC sangat dipengaruhi oleh fungsi *random* yang digunakan, terutama untuk penerapan eksploitasi dengan metode *random*, sehingga hasil yang didapatkan dari beberapa pengujian tidak akan sama persis walaupun parameter yang diinput tidak berbeda.

Penelitian yang dilakukan oleh Muhammad Zidny Alam (2013), membahas bagaimana menerapkan algoritma *Bee Colony* untuk menyelesaikan *Capacitated Vehicle Routing Problem* (CVRP) agar didapatkan hasil terbaik yaitu biaya total perjalanan yang minimum. Pada penelitian ini metode optimasi ABC merupakan metaheuristik yang dapat menjadi salah satu alternatif dan mampu diterapkan untuk menyelesaikan permasalahan CVRP dengan baik. Percobaan yang dilakukan untuk menyelesaikan data permasalahan untuk pelanggan kurang dari 100, algoritma ABC mampu menghasilkan solusi yang lebih baik dari solusi algoritma *Branch* and *Cut*. Sedangkan untuk data permasalahan yang memiliki pelanggan lebih dari 100 algortima ABC mampu menyelesaikan namun tidak lebih baik dari algoritma pembanding.

Rendra Firma Pratama, Purwanto, dan Mohammad Yasin, meneliti tentang bagaimana menerapkan algoritma ABC untuk menyelesaikan permasalahan *Travelling Salesman Problem*. ABC sebagai alat bantu perhitungan sangat membantu para *sales* untuk menentukan rute perjalanan yang lebih minimum

dalam setiap permasalahan TSP yang ditemui. Program *Artificial Bee Colony* ini mampu menyelesaikan masalah *Travellin Salesman Problem* dengan jumlah konsumen yang besar.

2.2 Landasan Teori

2.2.1 *Game*

Game adalah sesuatu yang dimainkan dengan aturan tertentu sehingga ada yang menang dan ada yang kalah, biasanya dalam konteks tidak serius dengan tujuan *refreshing* (Anggara, 2008). *Game* atau permainan menggunakan interaksi dengan antarmuka pengguna melalui gambar yang dihasilkan oleh piranti *video*.

Dalam buku Cerdas Dengan *Game*, Poggenpohl mengemukakan bahwa ”*video game* adalah alat efektif untuk belajar karena *game* mampu menawarkan lingkungan hipotesis untuk siswa, di mana mereka dapat mengeskplorasi berbagai keputusan alternatif tanpa resiko kegagalan. Pemikiran dan tindakan digabungkan menjadi perilaku yang bertujuan mencapai suatu tujuan. *Video game* mengajari kita cara menyusun strategi, mempertimbangkan alterenatif dan berpikir fleksibel”.

Terdapat empat elemen dasar yang meliputi semua *game* (Schell, 2008):

Mechanics

Mechanics adalah prosedur yang berlaku dalam suatu *game*. *Mechanics* mendeskripsikan tujuan dari sebuah *game*, apa yang harus dilakukan pemain untuk mencapai tujuan dari *game* tersebut, apa yang tidak boleh pemain lakukan dan apa yang terjadi bila dilakukan. Elemen *mechanics* dalam suatu *game* terdiri dari enam kategori yaitu :

a. *World*

Setiap *game* memiliki latar tempat atau dunia yang berbeda-beda. Dunia ini adalah sebuah *"magic circle"* dalam sebuah *gameplay* yang mendefinisikan bagaimana dunia tersebut hadir dan bagaimana dunia tersebut terhubung satu dan lainnya.

b. *Object and Attribute*

Object adalah elemen yang dapat dilihat dan dimanipulasi dalam suatu *game*. Atribut adalah informasi yang menjelaskan suatu objek.

c. *Action*

Action adalah segala tindakan yang dapat dilakukan dan di kendalikan oleh pemain dalam suatu *game.*

d. *Rule*

Peraturan adalah batasan terhadap hal-hal yang ada dalam suatu *game*. Peraturan tersebut menentukan apa yang pemain dapat

lakukan dan yang tidak dapat dilakukan, menentukan dunia, objek, konsekuensi dan tujuan utama dari *game* tersebut.

e. ***Ability***

Setiap *game* mengharuskan pemain melatih kemampuan tertentu, semakin tinggi tingkat kesulitan sebuah *game*, maka semakin tinggi pula kemampuan yang harus dimiliki.

f. *Opportunity*

Kesempatan dalam *game* menitik beratkan pada interaksi antara lima elemen sebelumnya. Kesempatan merupakan bagian penting, karena kesempatan dalam sebuah *game* mempengaruhi apakah *game* tersebut menyenangkan untuk dimainkan atau tidak.

Story

Story menjelaskan apa yang terjadi dalam *game*. Sangat jarang sekali ditemui *game* yang tidak memiliki *story*, karena *story* membuat *game* menjadi lebih hidup dan lebih bermakna untuk dimainkan. Beberapa *game* bahkan bisa terkenal di karenakan cerita yang bagus dan membuat perasaan *player* yang memainkannya campur aduk walaupun bisa dikatakan *game* tersebut memiliki grafik dan *gameplay* yang memadai.

Aesthetic

Aesthetic atau estetika menjelaskan bagaimana tampilan, suara dan rasa dalam *game*. Estetika merupakan aspek penting dalam sebuah desain *game*, karena menyangkut hubungan dengan pengalaman dalam bermain.

Technology

Teknologi menyangkut pada material yang digunakan dalam bermain sebuah *game*. Teknologi menghubungkan antara aspek mekanik, cerita dan estetika.

2.2.2 Ilmuwan Muslim

Sekitar abad ke-10 orang-orang Yunani kuno menganggap mata kita memancarkan sinar. Mereka berpikir ilutah yang memungkinkan manusia untuk melihat. Ini pemikiran yang salah. Orang pertama yang menyadari bahwa sebenarnya cahaya yang memasuki mata adalah Ibnu Al-Haytham, orang barat mengenalnya dengan nama Alhazen. Dia menemukan kamera pertama setelah melihat cahaya yang masuk melalui lubang jendela. Semakin kecil lubang gambar, semakin baik gambarnya. Ia terus meneliti, dan membuat kamera *Obskura* (Kata *Obskura* atau *Obscura* berasal dari kata Arab *Qamara* yang berarti kamar gelap atau kamar pribadi). Kamera *Obskura* merupakan cikal bakal kamera *modern* dengan baik Al Haytham adalah orang pertama yang berhasil memindahkan gambaran alam sekitar ke atas sebuah layar dalam ruangan. Dalam bukunya *Al Manazir* (Buku Ilmu Optik) Al Haytham menulis tentang kaca pembesar, yaitu sebuah lensa cembung yang memperbesar gambar. Dia telah berkesperimen dengan 27 jenis lensa yang berbeda. Buku Al Haytham diterjemahkan ke dalam bahasa latin dan dimanfaatkan oleh para ilmuwan barat seperti : Roger Bacon, Leonardo Da Vinci, Yohanes Kepler dan Isaac Newton. Buku Al Haytham menjadi dasar ditemukannya

kamera, kacamata, teleskop, dan mikroskop (Syaifuddin Al Indunisi, 2013).

Al-Jazari merupakan salah satu pionir penemuan robot. Sebagai kepala insinyur di Istana Artuklu, Turki, pada abad ke-11 dan 12, Al-Jazari menciptakan berbagai alat otomatis yang sangat maju untuk zamannya. Selain itu, beliau adalah salah satu penemu pertama yang menggabungkan teknologi dan estetika, karena banyak penemuannya yang tidak hanya melampaui masanya, namun juga indah, dengan ornamen-ornamen yang lekat dengan kehidupan orang-orang Turki pada masa itu, seperti ular, gajah, burung merak, serta patung-patung orang berjubah dan surban. Rangkaian penemuannya beragam, mulai dari robot pelayan yang bisa menghidangkan minuman secara otomatis, wastafel otomatis berbentuk merak, hingga sekelompok robot musisi yang dapat menghibur tamu-tamu kerajaan. Namun penemuannya yang paling terkenal adalah jam gajah. Lahir dengan nama Badi' al-Zaman Abu-'l-'Izz Ibn Isma'il Ibn al-Razzaz al-Jazari, Al-Jazari mendapatkan nama panggilan tersebut dari tanah kelahirannya, Al-Jazirah, sebuah daerah yang terletak antara Tigris dan Eufrat, yang lebih terkenal dengan nama Mesopotamia. Al-Jazari mengikuti jejak ayahnya untuk mengabdi kepada Sultan Dinasti Artuqid selama beberapa dekade di Diyar-Bakir yang kini berada di wilayah Turki. Pada 1206, beliau menyelesaikan sebuah buku tentang tehnik mesin berjudul *Al-Jami' Bayn Al-'Ilm Wa-'L-'Amal Al-Nafi' Fi Sinat'at Al-Hiyal* yang berarti "Buku Pengetahuan tentang Penemuan-Penemuan Geometris

yang Jenius" berisi kumpulan teori dan praktek mekanisme. George Sarton, penulis *Introduction to the History of Science* di tahun 1927 menyebut buku tersebut sebagai "Bahan rujukan yang paling terperinci dari jenisnya dan dapat disebut sebagai klimaks dalam pencapaian para ilmuwan muslim di bidangnya" (Cholis Akbar, 2016).

2.2.3 *Non-Player Character* (NPC)

NPC atau *agent otonomus* adalah sebuah sistem komputer yang hidup di dalam lingkungan buatan. Dia beraksi sesuai dengan *agent* yang ditanamkan padanya sehingga dia bisa tahu harus melakukan apa jika menerima suatu rangsangan. (Franklin dan Graesser, 1997).

Non Player Character kemudian ditulis sebagai NPC disebut juga *autonomous character* atau *autonomous agent* yaitu karakter yang terdapat dalam *game* atau *virtual reality* dimana memiliki kemampuan untuk melakukan gerakan secara otomatis atau tidak dikontrol secara *real time* oleh pemain. Secara garis besar NPC dapat diartikan sebagai karakter dalam *game* yang dikendalikan oleh komputer dan tidak dapat dikendalikan oleh pemain, untuk itu pengendalian NPC umumnya menggunakan kecerdasan buatan. Dengan kecerdasan buatan yang diberikan maka NPC dapat melakukan pekerjaan seperti yang dapat dilakukan oleh manusia dan membuat *game* lebih menarik serta variatif. Penelitian tentang kecerdasan buatan pada NPC dalam *game*, hingga saat ini masih terus dikembangkan. Kecerdasan buatan tersebut dikembangkan untuk merancang perilaku NPC. (JimHyuk, 2005)

Menurut Yunifa Miftachul Arif (2010) pada penelitiannya menjelaskan bahwa NPC adalah *agent otonomus* yang digunakan sebagai media interaktif dalam sebuah *game* atau *virtual reality*, di mana *agent* tersebut memiliki kemampuan untuk mengimprovisasi tindakannya sendiri serta dapat mewakili tokoh permainan.

NPC harus mampu beraksi sesuai dengan sensor yang dia terima. Artinya *agent* tidak hanya berada dan menjadi bagian dari sebuah lingkungan buatan, tetapi menjadi pasangan dari lingkungan buatan itu. Arsitektur dan mekanisme *agent* harus terhubung dengan lingkungannya sehingga dapat merasakan setiap tujuan yang dirancang untuknya dan beraksi untuk memenuhi tujuan itu. (Maturana 1975, Maturana dan Varela 1980, Varela 1991).

2.2.4 Kecerdasan *Bee Colony*

Kecerdasan *Bee Colony* atau yang biasa disebut *Artificial Bee Colony* (ABC) merupakan satu dari sekian algoritma optimasi yang diadaptasikan dari konsep *Swarm Intelligence* (SI). Algoritma ini pertama kali diusulkan pada tahun 2005 oleh seorang *Computer Engineering* di *Erciyes University*-Turki yang bernama Dervis Karaboga. Seperti diimplikasikan pada namanya, algoritma ini memodelkan kecerdasan kolektif lebah madu dalam mencari sumber makanan. setelah mendapatkannya lebah madu menyebarkan informasi kepada lebah lain di sarang, dan mencoba memaksimalkan jumlah nektar yang akan diserahkan

kesarang mereka. Ada 3 esensial pada dalam mencari sumber makanan pada lebah (Seeley TD, 1995) yaitu :

1. *Food Sources* (sumber makanan) : nilai atau kualitas dari suatu sumber makanan ditentukan oleh jaraknya dengan sarang lebah, banyaknya jumlah makanan, dan kemudahan dalam mengambil makanan tersebut.
2. *Employed Foragers* : merupakan lebah-lebah yang bertugas menyimpan informasi dari sumber makanan yang ditemukan.
3. *Unemployed Foragers* : merupakan lebah-lebah yang bertugas mencari sumber-sumber makanan yang dieksploitasi. Terdapat 2 jenis yaitu :
   - *Scouts*
   - *Onlookers*

Dua perilaku utama lebah madu yang digunakan dalam kecerdasan *bee colony* adalah mencari sumber makanan yang mengandung banyak nektar mengabaikan sumber makanan yang mengandung sedikit nektar. Pada kecerdasan ini, posisi sumber makanan mewakili solusi yang mungkin bisa menyelesaikan masalah dan jumlah nektar yang terkandung dalam sumber makanan mewakili kualitas (*fitnesss*) dari solusi-solusi tersebut.

Dalam algoritma ABC, setiap iterasi terdiri dari 3 langkah, yaitu :

1. Mengirim *employed* ke lokasi *foodsource* kemudian *employed* akan menghitung jumlah nektar yang terkandung dalam *foodsource* itu.
2. *Onlooker* menyeleksi *foodsource* berdasarkan informasi yang diperoleh dari *employed* dan memastikan *foodsource* mana yang akan dipilih.
3. Memunculkan *scout* kemudian mengirimnya ke kandidat *foodsource* baru yang dipilih secara acak.

Pada tahap inisialisasi, *employed bee* memilih secara acak posisi *foodosurce* dan menentukan jumlah nektarnya. Kemudian pada tahap pertama, *employed bee* menuju ke sarang dan berbagi informasi kandungan nektar yang ada pada *foodosurce* dengan *onlooker* yang menunggu di dalam *dance floor*. Setelah berbagi informasi, *employed bee* akan pergi menuju ke *foodosurce* siklus sebelumnya kemudian memilh *foodosurce* baru berdasarkan informasi yang diterima. Pada tahap ketiga, *onlooker* memilih *foodosurce* berdasarkan informasi jumlah nectar yang dipresentasikan di *dance floor*.

Dalam memodelkan perilaku lebah saat mencari sumber makanan menjadi algoritma ABC, posisi *foodsource* dianggap sebagai solusi yang mungkin diambil (*possible solution*) untuk suatu masalah optimasi dan jumlah *nectar* pada setiap *foodsource* menunjukkan kualitas (fitnesss) dari solusi itu. Jumlah *employed* atau *onlooker* sama dengan jumlah solusi dalam populasi.

```
Initialization Phase
REPEAT
  Employed Bees Phase
  Onlooker Bees Phase
  Scout Bees Phase
  Memorize the best solution achieved so far
UNTIL(Cycle=Maximum Cycle Number or a Maximum CPU time)
```

Gambar 2.1 Langkah Algoritma *Bee Colony* (Dervis Karaboga, 2005)

2.2.4.1 Inisialisasi

Pada tahap awal, ABC akan membuat inisialisasi populasi awal *P(G=0)* yang didistribusikan secara acak untuk *SN* solusi (posisi *foodsource*), di mana *SN* mewakili jumlah dari populasi. Setiap posisi *foodsource xi (i = 1, 2, ..., SN)* berada pada *D* vektor dimensi. Di mana *D* adalah parameter optimisasi. Setelah inisialisasi, setiap tahap dalam algoritma ABC diiterasi sebanyak *C* kali (C = 1,2, …, *Cmax*).

2.2.4.2 Fase *Employed*

Artificial employed dan *onlooker* membuat modifikasi posisi *foodsource* untuk menentukan *foodsource* baru. Di alam, lebah asli akan mengumpulkan kemudian membandingkan informasi visual di sekitarnya sebelum menentukan *foodsource* baru yang akan dituju. Pada algoritma ABC penentuan posisi

foodsource baru juga ditentukan berdasarkan pembandingan informasi yang dikumpulkan oleh lebah. Hanya saja, di dalam pemodelan algoritma ABC, lebah tidak membandingkan informasi visual saat menentukan *foodsource* baru. Lebah secara acak memilih posisi *foodsource* baru dan membuat posisi *foodsource* baru dengan menggunakan persamaan (2.1).

$$X'_{ij} = X_{ij} + \emptyset_{ij} \times R \qquad (2.1)$$

Di mana X'_{ij} adalah posisi baru dan X_{ij} posisi awal lebah. Sedangkan $\emptyset_{ij}$ adalah nilai *random* antara [-1,1] dan R adalah batas ruang lingkup.

2.2.4.3 Fase *Onlooker*

Setelah semua proses pencarian *foodsource* baru selesai dilakukan oleh semua *employed*, mereka akan memberikan informasi pada *onlooker*. Pada fase ini akan dihitung nilai *fitness* dari posisi awal dan posisi baru. Jika nilai *fitnesss* pada *foodsource* baru lebih tinggi dari *foodsource* sebelumnya, maka lebah akan menyimpan posisi *foodsource* baru dalam memorinya dan membuang posisi *foodsource* lama. Sebaliknya, lebah akan menyimpan posisi *foodsource* lama jika ternyata fitnesss *foodsource* baru lebih rendah dari *foodsource* lama. Untuk menghitung nilai tersebut digunakan persamaan (2.2) dengan d_{ij} adalah jarak posisi lebah saat ini ke target.

$$d_{ij}{}^2 = (x_j - x_i)^2 + (y_j - y_i)^2 \qquad (2.2)$$

2.2.4.4 Fase *Scout*

Fitnesss setiap posisi *foodsource* baru akan dihitung oleh lebah, dan *foodsource* baru akan diabaikan jika nilai *fitness*-nya lebih rendah dari *foodsource* lama. Lebah akan mencari kemungkinan posisi *foodsource* baru untuk menggantikan *foodsource* yang diabaikan tadi. Jika selama L (*limit*) kali pencarian posisi *foodsource* baru tidak ditemukan, maka *scout* akan mencarikan *foodsource* baru untuk lebah *i*.

$$X'_{ij} = X_i + rand\,[-1,1] \times R, \quad if \;\; L > max \qquad (2.3)$$

2.2.5 Game Tools

2.2.5.1 *Unity 5*

Menurut Will Goldstone (2009), *Unity* membuat proses produksi suatu *game* menjadi lebih mudah. *Unity* menyediakan rangkaian langkah logikal untuk membuat suatu skenario *game*. *Unity* dapat digunakan untuk membuat beragam tipe *game*.

Untuk membuat suatu *game* yang menarik, *Unity* menyediakan berbagai macam fitur dukungan, yaitu :

a. *Assets*

Unity menyediakan blok untuk meletakkan sesuatu seperti gambar, model 3 dimensi dan suara yang akan digunakan dalam sebuah *game*.

b. *Scenes*

Dalam *Unity*, *scene* dipakai sebagai level individual atau area dari konten *game*, misalnya tampilan menu. Dengan membuat banyak *scene* dalam suatu *game*, pembuat *game* bisa melakukan tes pada bagian dari *game* secara terpisah.

c. *Game Object*

Ketika suatu *asset* dimasukkan ke dalam *scene*, maka *asset* tersebut menjadi sebuah *game objects*. Dimana *game objects* tersebut dapat digerakkan, dirubah ukurannya dan rotasinya.

d. *Components*

Components hadir dalam berbagai fungsi, yang digunakan untuk mempengaruhi suatu *game object*. Dengan memberikan *components* pada suatu *game object*, maka *game object* tersebut akan memiliki karakteristik dari komponen tersebut, misalnya membuat *game object* memiliki berat dan terpengaruhi gravitasi.

e. *Scripts*

Scripts merupakan bagian penting dalam produksi suatu *game* dan bisa disebut sebagai faktor kunci. Pada *Unity*, *scripts* ditulis dengan menggunakan *JavaScript*, C# dan *Boo*. Untuk menulis *scripts* di *Unity*, digunakan *script editor* tersendiri yang disediakan oleh *Unity*.

f. *Prefabs*

Prefabs digunakan untuk menyimpan suatu *game object* beserta dengan *component* dan konfigurasi lainnya, sehingga memungkinkan *object* tersebut bisa digunakan kembali tanpa melakukan konfigurasi berulang kali.

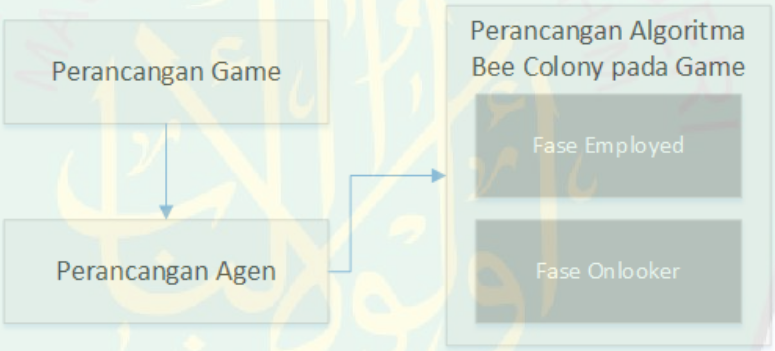
2.2.5.2 Bahasa Pemrograman *C#*

C# merupakan salah satu bahasa pemrograman pilihan yang tersedia di *Unity* 5 yang bermanfaat dalam pembangunan sebuah game. *C#* merupakan bahasa pemrograman berorientasi objek yang dikembangkan oleh *Microsoft* sebagai bagian dari inisiatif kerangka *.NET Framework*. Bahasa pemrograman ini dibuat berbasis bahasa *C++* yang telah dipengaruhi oleh bahasa pemrograman lainnya seperti *Java*, *Visual Basic*, *Delphi* dan lain-lain.

BAB III

DESAIN DAN PERANCANGAN

Agent dalam sebuah *game* membutuhkan *path planning* untuk menentukan jalur menuju targetnya. Penelitian ini ditujukan untuk membuat NPC *otonomus* mencari jalur terpendek menuju *player* dari arah yang berbeda. Pergerakan NPC diterapkan pada ruang 2 dimensi. Tahapan penelitian dilakukan berdasarkan diagram alur dibawah ini.



Gambar 3.1 Diagram Alur Penelitian

3.1 Perancangan *Game*

Game yang dibangun adalah *game First Person Controller* atau *game* yang menggunakan sudut pandang orang pertama. *Game* ini bergenre *action* namun memiliki unsur edukasi. NPC musuh pada *game* ini dibuat untuk menerapkan algoritma *Bee Colony* guna mendapatkan perilaku pencarian terhadap *player*.

3.1.1 Keterangan Umum

Game yang dibangun diberi nama *Guess Who*. *Game* berbasis *dekstop* dan dimainkan secara *single*. *Game* ini merupakan *game* edukasi yang mengenalkan ilmuwan-ilmuwan muslim terdahulu kepada anak-anak.

NPC musuh pada *game* ini adalah boneka beruang. NPC ini akan menghalangi *player* dalam menyelesaikan misi permainan.

3.1.2 Deskripsi Karakter

a. Karakter Utama

Karakter utama (*Player*) merupakan karakter *First Person Shooter* (ditampilkan dengan sudut pandang orang pertama).

b. NPC *Enemy*

NPC *enemy* digambarkan dengan *zombie* yang akan menghalangi *player* untuk menyelasaikan misi. Pada NPC ini algoritma *bee colony* akan diterapkan.

Gambar 3.2 NPC *Enemy (Zombear)*

c. *Item* Huruf

Di bawah ini adalah contoh *item* huruf yang harus dikumpulkan oleh *player* sesuai dengan misi pada *game*. Jika huruf yang dikumpulkan tidak sesuai dengan misi maka nilai kesempatan *player* akan dikurangi 1. Jika kesempatan habis maka *player* kalah atau *game over*.

Gambar 3.3 *Item* Huruf

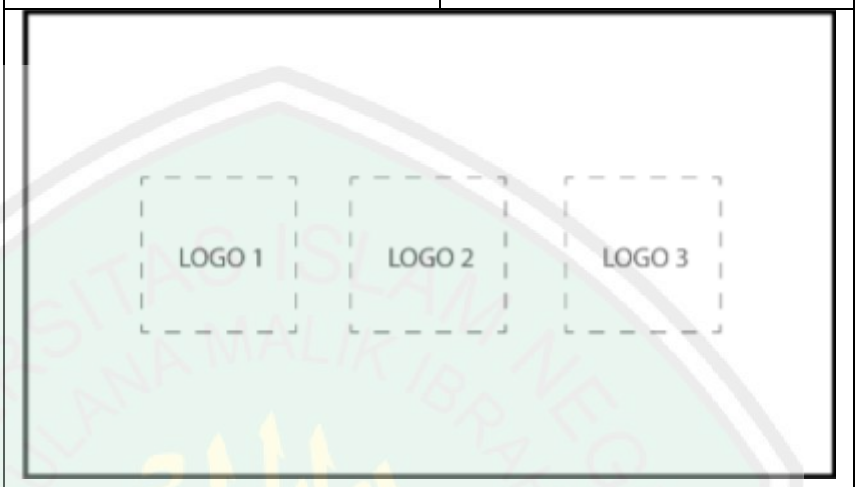
3.1.3 Desain *Interface Game*

Antarmuka atau yang biasa disebut dengan *interface* merupakan bagian penting dalam membangun sebuah aplikasi. *Interface* befungsi untuk menghubungkan *user* dengan sistem yang telah dirancang. Dalam sebuah *game* dibutuhkan *interface* yang menarik dan *user friendly*. Rancangan *interface game* pada penelitian ini adalah sebagai berikut :

a. *Splash Screen*

Bagian ini akan menampilkan *splash screen* dengan *background* yang terdiri logo *Unity* 3D, logo UIN Malang dan logo *developer*.

Tabel 3.1 Tampilan *Splash Screen*

Project : *Switch On*	*Date* : 17 Juli 2017
Scene : *Splash Screen*	*ScreenID* : 1
	

b. *Main Menu*

Berikut adalah tampilan menu utama dari *game Guess Who*, terdapat beberapa tombol disini. *Start* akan memulai permainan, *Option* akan membawa *user* ke menu pengaturan, *Credit* yang berisi nama *developer* dan tahun serta tujuan pembuatan, dan tombol *Quit* yang akan membawa *player* keluar dari halaman *game*.

Tabel 3.2 Tampilan *Main Menu*

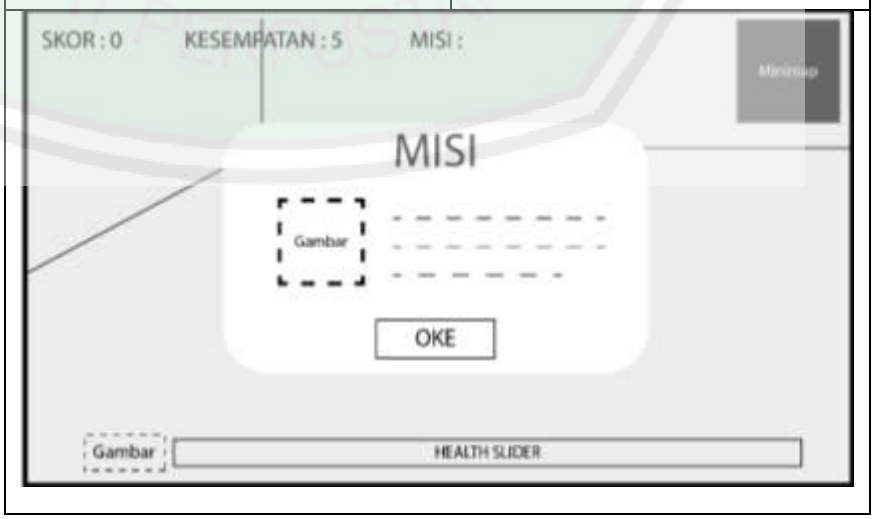
Project : *Switch On*	*Date* : 17 Juli 2017
Scene : *Main Menu*	*ScreenID* : 2

c. *Gameplay*

Merupakan tampilan saat *game* dimulai, *player* akan ditampilkan dengan *mode First Person Shooter* atau sudut pandang orang pertama. Ketika p*layer* memasuki area permainan, *gameplay* akan menampilkan informasi tentang misi yang harus diselesaikan.

Tabel 3. 3 Tampilan *Gameplay*

Project : *Switch On*	*Date* : 17 Juli 2017
Scene : *Gameplay*	*ScreenID* : 3

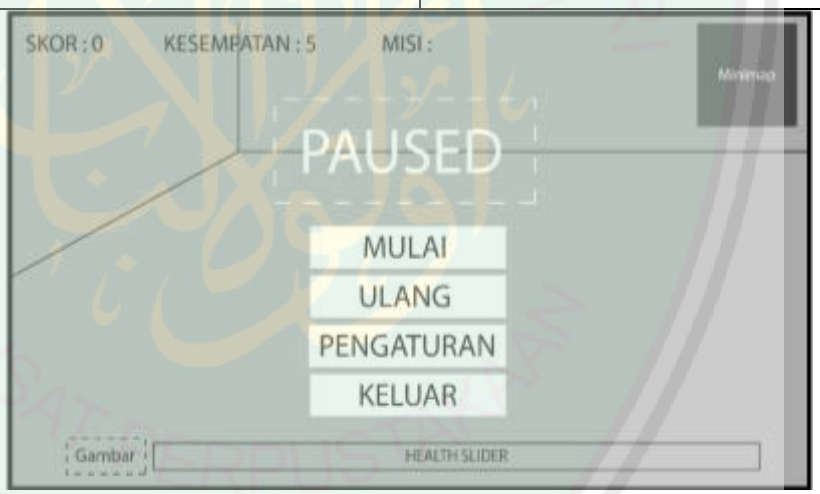


d. *Pause Menu*

Saat *player* sedang bermain dan menekan tombol *Esc* pada *keyboard*, maka *game* akan membawa *player* ke halaman ini, dimana *player* dapat mengubah *setting game*, memulai ulang *game* atau keluar dari *game*. Ketika jendela *pause menu* muncul, permainan akan berhenti sejenak sampai *player* kembali menekan tombol *Esc* atau tombol mulai untuk melanjutkan permainan.

Tabel 3.4 Tampilan *Pause Menu*

Project : *Switch On*	*Date* : 17 Juli 2017
Scene : *Gameplay*	*ScreenID* : 3.1

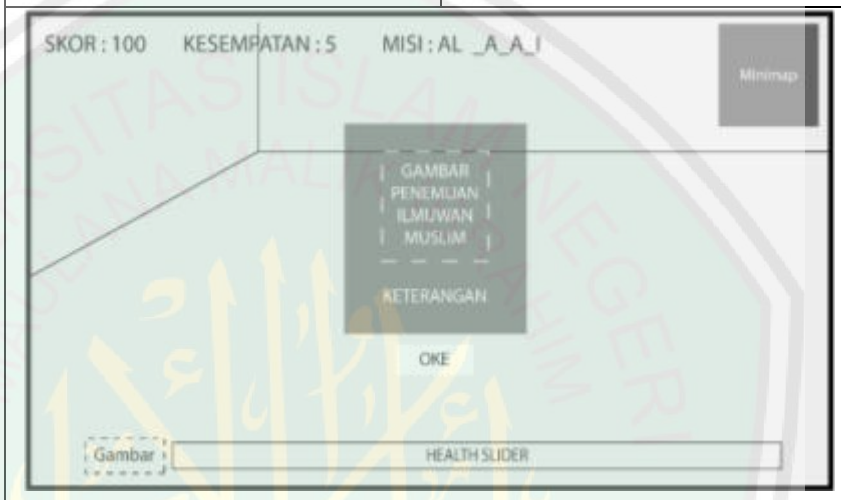


e. *Item* Misi

Player harus mengumpulkan beberapa *item* huruf untuk menyelesaikan misi. Huruf-huruf tersebut merupakan *puzzle* untuk melengkapi huruf-huruf yang kosong pada *tab* misi di bagian atas

gameplay. Setiap *item* huruf memiliki informasi terkait tokoh yang terdapat pada misi.

Tabel 3.5 Tampilan Informasi pada *Item* Huruf

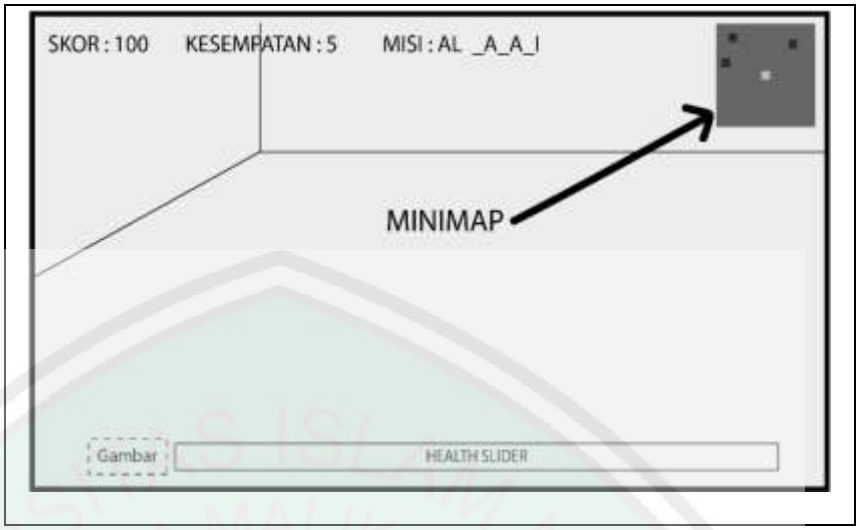
Project : *Switch On*	*Date* : 17 Juli 2017
Scene : *Gameplay*	*ScreenID* : 3.3
	

f. *Minimap*

Minimap berfungsi sebagai petunjuk untuk mengetahui letak *puzzle* huruf yang harus dikumpulkan. Namun *minimap* tidak mendeteksi keberadaan musuh yang ada pada *gameplay*. *Minimap* terletak di pojok kanan atas *gameplay*.

Tabel 3.6 Tampilan *Minimap*

Project : *Switch On*	*Date* : 17 Juli 2017
Scene : *Gameplay*	*ScreenID* : 3.4



g. *Win Game*

Seperti yang dijelaskan pada bagian sebelumnya, *player* akan memenangkan permainan jika semua *puzzle* huruf terkumpulkan sesuai dengan misi. Terdapat 3 tombol pada tampilan *Win* yaitu, tombol Ulang untuk memulai kembali permainan pada level yang sama, kemudian ada tombol Lanjut untuk melanjutkan permainan pada *level* berikutnya dan tombol *Keluar* untuk keluar dari permainan.

Tabel 3.7 Tampilan *Win Game*

Project : *Switch On*	*Date* : 17 Juli 2017
Scene : *Gameplay*	*ScreenID* : 3.5

h. *Game Over*

Player akan kalah jika poin kesempatan dan *Health point* sama dengan nol. Maka tampilan ketika player kalah seperti pada **tabel 3.8**. *Game* akan secara otomatis di-*restart* setelah lima detik jika berada pada keadaan ini.

Tabel 3.8 Tampilan *Game Over*

Project : *Switch On*	*Date* : 17 Juli 2017
Scene : *Gameplay*	*ScreenID* : 3.6

3.1.4 Skor

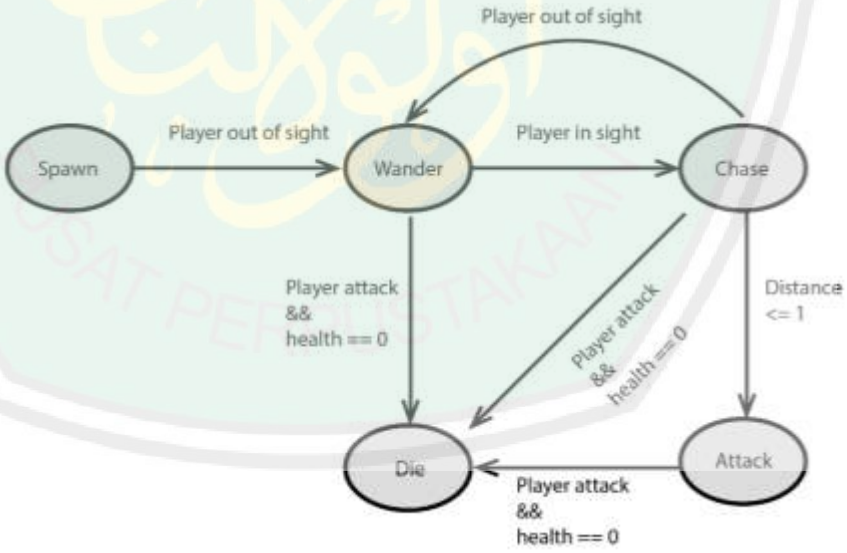
Setiap *item* huruf yang sesuai pada misi mempunyai nilai skor. Masing-masing *item* bernilai 100 poin. Sehingga skor dari permainan ini ditentukan dari *item* huruf yang telah dikumpulkan oleh *player*.

3.1.5 Pelevelan

Pada penelitian ini hanya dirancang 1 *level* saja untuk pengujian Algoritma *Artificial Bee Colony*.

3.1.6 *Finite State Machine*

FSM pada *game* ini digunakan untuk menentukan perilaku NPC. Sedangkan karakter utama mengikuti perintah *user* yang menjalankan *game*. Berikut adalah *finite state machine game* yang akan dirancang :



Gambar 3.4 *Finite State Machine* NPC

Pada *game* ini NPC akan di-*spawn* secara langsung ketika *game* dimainkan. *Wander* adalah proses NPC mengelilingi area permainan untuk

menemukan *player* atau yang dikenal dengan istilah patroli. Proses *wander* dilakukan jika jarak NPC dengan *player* sangat jauh atau diluar dari jarak pandangnya. Sebaliknya jika *player* berada pada jarak pandang NPC, maka NPC akan mengejar *player*. Proses tersebut ditandai dengan proses *chase*.

NPC dapat menyerang *player* jika jarak posisinya <= 1f. Pada saat itu *player* dapat menghindari NPC atau menyerang balik NPC tersebut jika telah mempunyai senjata. Proses *hunt* adalah proses dimana NPC mencari *player* pada posisi terdekatnya. Namun jika NPC terkena serangan balik dari *player* dan *health point* NPC <= 0, maka NPC akan di-*destroy* dari area permainan dan di-*respawn* setelah 10 detik.

3.2 Perancangan *Agent*

Pada *game* ini *agent* menerapkan pola pencarian sumber makanan seperti yang dilakukan oleh lebah. Posisi sumber makanan ditentukan secara acak. Sumber makanan dianggap sebagai posisi baru ketika mendekati target.

Setiap *agent* memiliki properti *speed*, *power*, *sensor*, *efector*, dan *behavior*. Properti tersebut dijelaskan pada tabel dibawah ini.

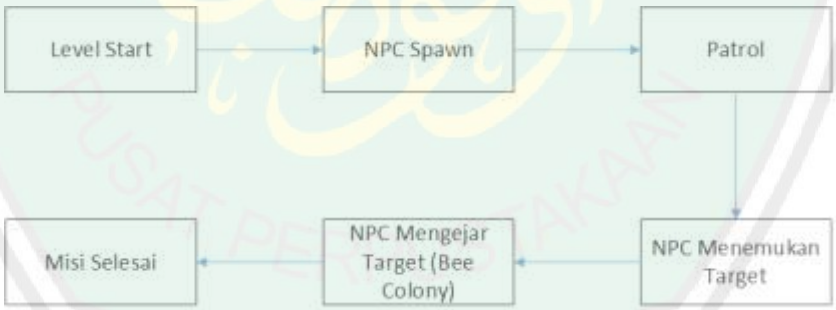
Tabel 3.9 Properti *Agent*

Parameter	*Agent*	Target	Keterangan
Speed	Rendah dan Tinggi	Rendah dan Tinggi	
Power	50	100	

Sensor	*Collider*		Mendeteksi keberadaaan target
Efector	*Speed*		Menentukan kecepatan
Behavior	*Wander*		Menyusuri *environtment* (area permainan)
	Tracking Target		Mencari musuh

3.3 Perancangan Algoritma *Bee Colony* pada *Game*

Algoritma *bee colony* dijadikan sebagai algoritma *pathfinding* yang diterapkan kepada NPC. Pada kasus ini NPC dianalogikan sebagai lebah dan *player* dianalogikan sebagai sumber makanan. Sehingga NPC akan mencari jalan terbaik untuk menemukan lokasi *player* dan mengikutinya. Algoritma akan jalan ketika *player* dalam jarak pandang NPC.



Gambar 3.5 Alur Kerja NPC Pada *Game*.

Iterasi *bee colony* yang diterapkan kepada NPC terdiri dari 2 fase yaitu fase *employed* untuk menemukan informasi posisi sumber makanan yang baru. Kemudian informasi tersebut akan diseleksi pada fase *onlooker* mana yang terbaik untuk dipilih di antara posisi sumber makanan tersebut.

a. Inisialisasi

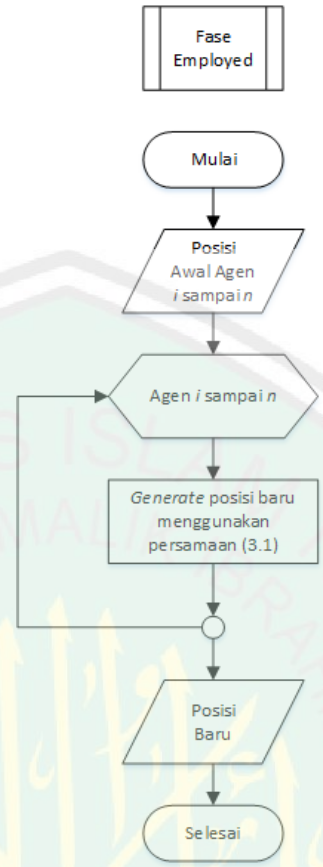
Tahap inisiasi akan ditentukan posisi awal lebah, posisi akhir lebah (target), dan *vector* dimensi (D). Pada penelitian ini populasi lebah akan dianggap sebagai *agent*.

b. Fase *Employed*

Pada fase *Employed*, setiap agent akan mencari posisi barunya secara acak dengan menggunakan rumus berikut ini,

$$P'_{ij} = P_{ij} + \emptyset_{ij} \times R \qquad (3.1)$$

Dimana *Pij* adalah poisisi *agent* ke-i saat ini, *Pij'* merupakan posisi baru lebah ke-*i* dan *R* adalah radius *agent* untuk menemukan posisi baru. Sedangkan Φ merupakan angka acak antara -1 dan 1 (Dervis Karaboga, 2005).



Gambar 3.6 Flowchart Fase *Employed*

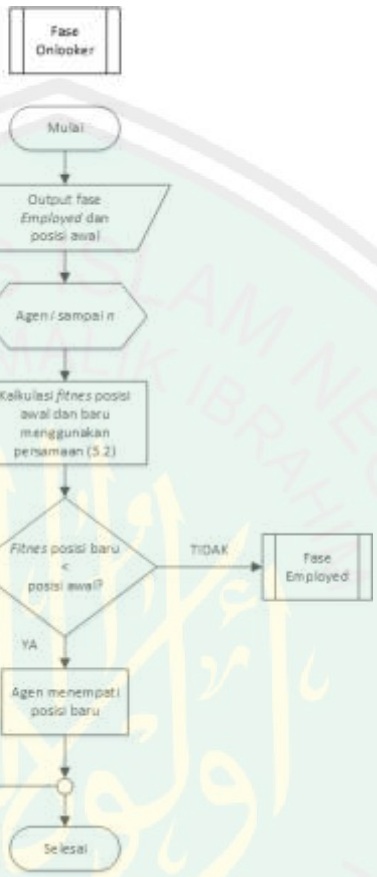
c. Fase *Onlooker*

Setelah ditentukan posisi baru maka akan dihitung nilai *fitness* posisi awal dan posisi baru *agent* yang telah dicari menggunakan rumus berikut.

$$d_{ij}{}^2 = (x_j - x_i)^2 + (z_j - z_i)^2 \qquad (3.2)$$

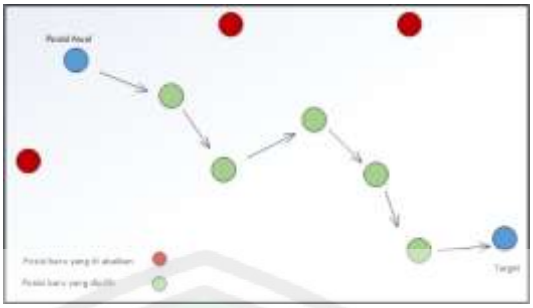
Nilai *fitness* terbaik akan dipilih pada fase ini. Jika nilai *fitness* posisi baru lebih baik dari posisi awal maka *agent* akan menempati posisi baru. Nilai *fitness* terbaik dalam penelitian ini adalah nilai *fitness* yang

lebih kecil sehingga nilai *fitness* tersebut merepresentasikan posisi yang lebih dekat dengan target.



Gambar 3. 7 Flowchart Fase *Onlooker*

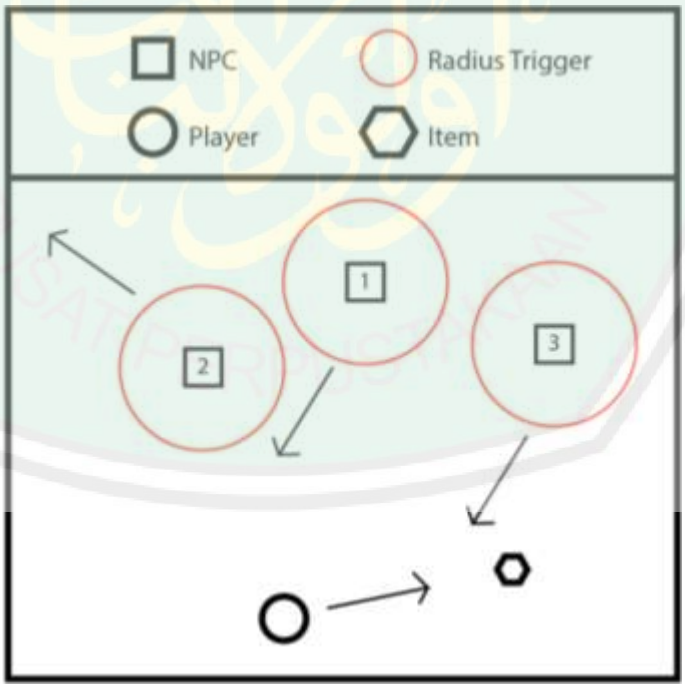
Iterasi di atas dilakukan hingga *agent* mencapai posisi target. Pergerakan *agent* di ilustrasikan seperti gambar di bawah ini.



Gambar 3.8 Ilustrasi Pergerakan *Agent* Menuju Target

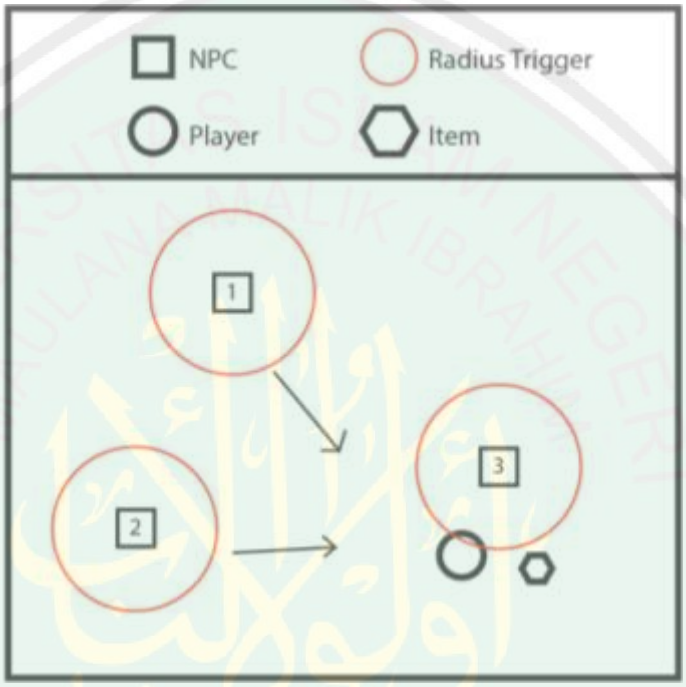
3.4 Skenario Pergerakan NPC

Ketika permainan pertama kali dijalankan, NPC (*agent*) secara otomatis berpatrol di sekitar area permainan untuk menemukan target (*player*). Pada Gambar 3.9 *player* berjalan menuju *item* yang harus dikumpulkan dan NPC 3 juga melakukan patrol ke arah yang sama.



Gambar 3.9 Skenario Patrol

Ketika *player* berada pada radius salah satu NPC, maka NPC otomatis melakukan pengejaran terhadap *player*. Pergerakan NPC tersebut dilakukan menggunakan algoritma *bee colony*, yang mana jika salah satu NPC bergerak menuju *player* maka NPC lainpun akan mengikutinya.



Gambar 3.10 Skenario Deteksi dan Pengejaran

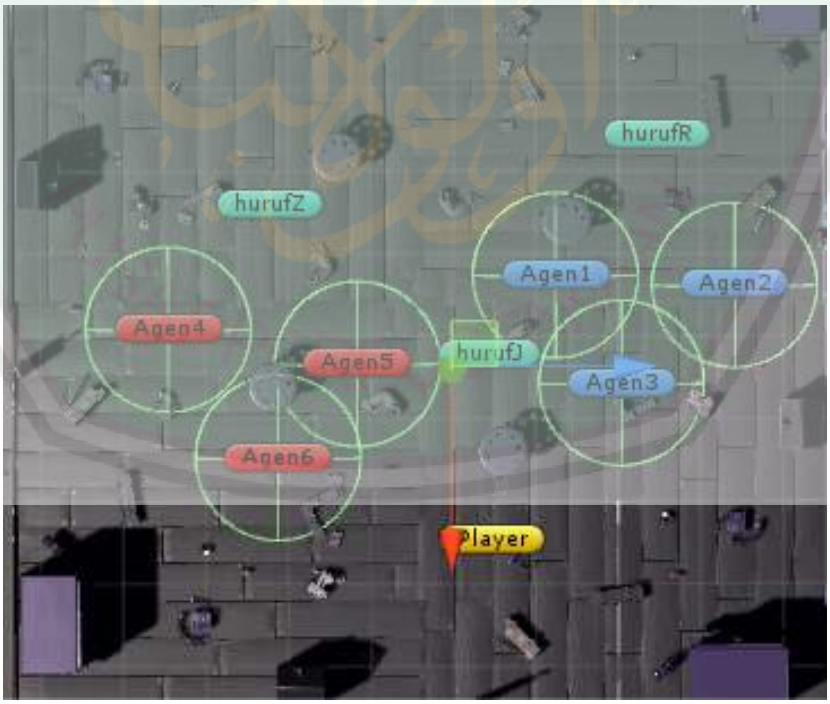
Jika *player* tetap berada pada radius NPC, maka NPC akan terus mengejar hingga mengerumuni *player*, kecuali *player* kembali berada di luar radius NPC dan secara otomatis NPC melakukan patrol kembali.

BAB IV

UJI COBA DAN PEMBAHASAN

4.1 Uji Coba Algoritma *Bee Colony*

Pergerakan sekelompok *agent* yang telah diuraikan pada Bab 3 diimplementasikan pada sebuah *game* komputer menggunakan *Unity 3D*. *Game* ini melibatkan 2 kelompok *agent* dengan masing-masing kelompok berjumlah 3 *agent*, dan satu target. Radius *agent neighbor* (r) = 2, kecepatan *agent* sebesar 2f dan *stopping distance* sebesar 1,8. *Input* untuk setiap percobaan pada skenario berupa posisi awal *agent* dari arah yang berbeda. Sedangkan outputnya rute pergerakan *agent* dari posisi awal ke posisi akhir mendekati target.



Gambar 4.1 Keadaan Awal *Agent* dan Target.

Gambar 4.1 menunjukan posisi awal *agent* dan target serta *item* yang harus diambil untuk menyelesaikan misi. *Agent* berwarna biru merupakan kelompok *agent* pertama sedangkan *agent* berwarna merah adalah kelompok *agent* kedua. Selama perjalanan menuju target, masing-masing *agent* akan di-*deploy* pada posisi yang berbeda-beda seperti yang tertera pada **Gambar 4.1.** Kemudian *agent-agent* tersebut akan mencari posisi terdekat dengan target dari posisi sekitarnya (*neighbor*). Radius posisi *neighbor* yang ditentukan pada posisi ini adalah 2 baik itu pada koordinat x maupun y sehingga *agent* akan sedikit demi sedikit akan mendekati posisi target.

Fitness function untuk pergerakan *agent* adalah menempatkan *agent* pada posisi sedekat mungkin dengan posisi target. Dengan kata lain fungsi tujuan yang hendak dicapai adalah meminimalkan jarak antara tiap *agent* dengan target. Jadi setiap kali iterasi diharapkan posisi *agent* semakin mendekati target atau jarak *agent* ke target semakin kecil. Tabel 4.1 adalah posisi awal *agent* dipercobaan pertama.

Tabel 4.1 Posisi Awal *Agent*

Agent	**X**	**Y**	**Z**
1	-27,4	0,2	-7,6
2	-26,2	0,2	14,1
3	-14,3	0,2	0,4
4	-20,9	0,2	-54,5
5	-16,7	0,2	-31,6

6	-5,3	0,2	-41,2

Pengujian algoritma dilakukan dengan menggunakan satu skenario yaitu meletakan salah satu *item* di antara *agent-agent*. Ketika *game* dimulai maka *agent* akan melakukan *wander* (patrol di sekitar area permainan) dan *player* mulai mencari *item* untuk menyelesaikan misi. Skenario tersebut bertujuan untuk mendeteksi kelompok *agent* mana yang akan mengejar *player*. Jika nilai *fitnesss* (jarak) salah satu kelompok <= 10 maka kelompok itulah yang akan mengejar *player*. Pergerakan kelompok *agent* digambarkan dengan 3 keadaan yaitu keadaan mendeteksi, mengejar, dan mengerumuni.

Berikut adalah keadaan *agent* ketika salah satu kelompok mendeteksi keberaadaan *player*.

Gambar 4.2 Keadaan *Agent* Mendeteksi *Player*

Pada gambar 4.2, *agent* 1 mendeteksi keadaan *player* ketika *player* mencoba mengambil *item* huruf J karena jaraknya yang bernilai <=10, sedangkan jarak *agent* 5 >10. Koordinat *player* adalah (-15.9, -17.5). Berikut adalah jarak *agent* 1 dan *agent* 5 terhadap *player*.

Tabel 4.2 Koordinat dan Jarak *Agent*

Agent	**Koordinat (x,z)**	**Jarak**
1	(-22.7, -9.8)	9.943
5	(-26.6, -10.1)	12,962

Setelah *agent* 1 mendeteksi keberadaan *player*, maka *agent* akan mengejar atau berusaha mendekati *player*. *Agent-agent* yang mengejar *player* adalah *agent* 1, 2, dan 3 karena merupakan anggota kelompok 1. Sedangkan *agent* 4, 5, 6 tetap melakukan patrol di sekitar area permainan hingga jaraknya dengan *player* <= 10. Di bawah ini adalah keadaan *agent* 1, 2, 3 ketika mengejar *player*.

Gambar 4.3 Keadaan *Agent* Mengejar *Player*

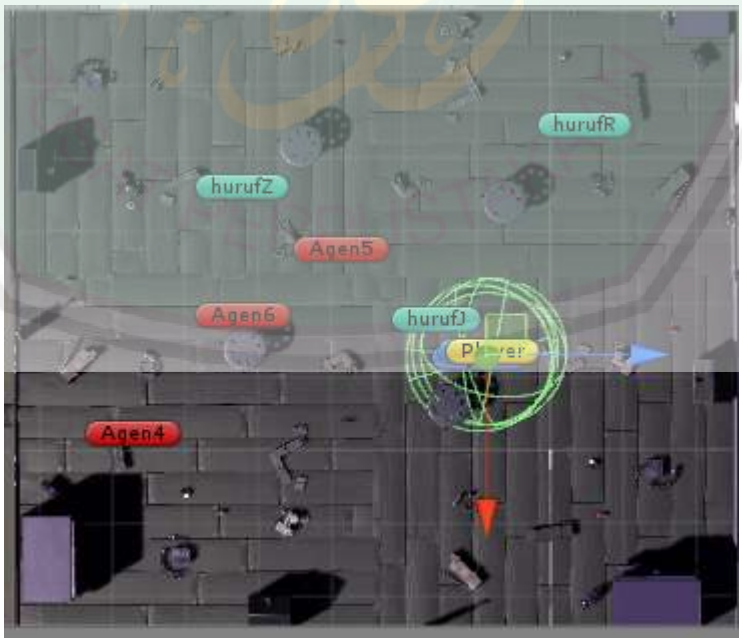
Ketika keadaan mengejar itulah algoritma *bee colony*, *agent* berusaha mendekati posisi *player* dengan cara mencari posisi baru secara random di sekitarnya dengan radius *neighbor* = 2 (fase *employed*). Setelah didapatkan posisi baru, maka dihitung nilai *fitnesss* posisi baru dan posisi *agent* saat ini. Setelah diketahui nilai masing-masing *fitnesss*-nya maka kedua nilai tersebut dibandingkan. Apakah *fitnesss* posisi baru < posisi saat ini. Jika lebih kecil maka posisi baru akan ditempati oleh *agent*. Begitupun sebaliknya jika lebih besar maka *agent* akan tetap menempati posisinya saat ini (fase *Onlooker*). Di bawah ini peneliti akan menunjukan data salah satu *agent* saat mendekati *player* mulai dari iterasi ke-1 sampai ke-5. Data tersebut meliputi posisi *player*, posisi awal *agent* (posisi saat ini), posisi baru dan masing-masing *fitness* kedua posisi.

Tabel 4.3 Data *Agent* 1 Pada Iterasi 1-5

Iterasi	**Posisi Player (x,z)**	**Posisi Awal *Agent* (x,z)**	**Posisi Baru *Agent* (x,z)**	**Fitness Awal**	**Fitness Baru**

1	(-15.1, -17.4)	(-27.4, -7.6)	(-27.5, -8.4)	15.695	15.317
2	(-15.5, -17.5)	(-27.5, -8.4)	(-27.9, -7.5)	15.017	15.870
3	(-15.9, -17.5)	(-27.5, -8.4)	(-28.3, -8.4)	14.723	15.355
4	(-15.9, -17.5)	(-27.5, -8.4)	(-26.6, -10.1)	14.723	12.962
5	(-15.9, -17.5)	(-26.6, -10.1)	(-27.9, -8.2)	12.962	15.192

Pada iterasi ke-1, *player* terletak pada posisi (-15.1, -17.4) dan *agent* 1 terletak pada posisi (-27.4, -7.6) dengan nilai *fitness* 15.695. Sedangkan posisi baru yang didapat pada iterasi ini adalah (-27.5, -8.4) dengan nilai *fitness* 15.317. Jika dibandingkan, nilai *fitnesss* posisi baru lebih kecil dari *fitness* posisi awal. Oleh karena itu, pada iterasi berikutnya (ke-2) *agent* 1 akan menempati posisi baru. Begitupun selanjutnya hingga *agent* 1, 2, 3 mengerumuni *player* seperti gambar di bawah.



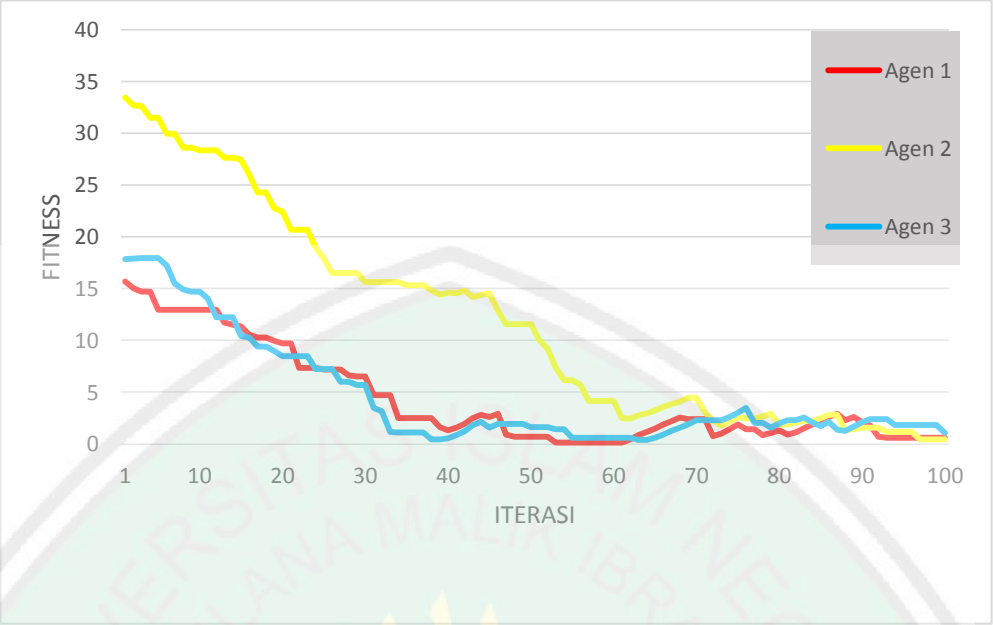
Gambar 4.4 Keadaan *Agent* Mengerumuni *Player*

Gambar 4.4 menunjukan bahwa *agent* telah mencapai *fitnesss* terbaiknya atau mencapai posisi yang sangat dekat dengan *player* dan melebihi nilai *stopping distance*. Tabel di bawah merupakan data *fitness* terbaik masing-masing *agent*.

Tabel 4.4 Fitnesss Terbaik *Agent*

Agent	**Nilai Fitness Terbaik**	**Iterasi ke -**
1	0.916	47
2	0.455	96
3	0.437	37

Iterasi dilakukan sebanyak 100 kali. Dari seluruh iterasi, masing-masing *agent* tidak menemukan *fitness* terbaiknya pada iterasi yang sama. Data keseluruhan *agent* 1, 2, dan 3 dapat dilihat pada Tabel 4.5, Tabel 4.6, dan Tabel 4.7. Data-data tersebut adalah data hasil pengujian *agent* dalam mencari posisi *player*, sehingga pergerakannya digambarkan seperti pada grafik di bawah ini.



Gambar 4. 5 Grafik Hasil Pengujian.

Grafik yang menurun menunjukan *agent* terus menerus mendekati posisi target hingga pada iterasi ke-100. Nilai *fitness* masing-masing *agent* sudah melewati nilai *stopping distance* (1,8) sebelum mencapai iterasi ke-100. Adanya naik turun yang digambarkan pada grafik disebabkan oleh perpindahan posisi *player* (target), sehingga akan dihitung kembali nilai *fitness* dari posisi *agent* saat ini dengan posisi *player* yang baru. Setelah menempuh iterasi tersebut, didapatkan hasil bahwa kesemua *agent* telah berhasil mencapai posisi terdekat dengan *player*.

4.2 Hasil Uji Coba

Proses uji coba dilakukan pada simulasi *Unity 3D* yang melibatkan 5 *agent*. Uji coba dilakukan untuk mengetahui apakah Algoritma *Bee Colony* dapat diterapkan pada *game* yang telah dirancang oleh peneliti. Berikut adalah data hasil uji coba setiap *agent*.

Tabel 4.5 Data Hasil Uji Coba *Agent* 1

Iterasi	Posisi Player (x,z)	Posisi Awal *Agent* (x,z)	Posisi Baru *Agent* (x,z)	Fitness Awal	Fitness Baru
1	(-15.1, -17.4)	(-27.4, -7.6)	(-27.5, -8.4)	15.695	15.317
2	(-15.5, -17.5)	(-27.5, -8.4)	(-27.9, -7.5)	15.017	15.870
3	(-15.9, -17.5)	(-27.5, -8.4)	(-28.3, -8.4)	14.723	15.355
4	(-15.9, -17.5)	(-27.5, -8.4)	(-26.6, -10.1)	14.723	12.962
5	(-15.9, -17.5)	(-26.6, -10.1)	(-27.9, -8.2)	12.962	15.192
6	(-15.9, -17.5)	(-26.6, -10.1)	(-28.1, -11.6)	12.962	13.510
7	(-15.9, -17.5)	(-26.6, -10.1)	(-26.8, -9.9)	12.962	13.254
8	(-15.9, -17.5)	(-26.6, -10.1)	(-28.1, -11.5)	12.962	13.577
9	(-15.9, -17.5)	(-26.6, -10.1)	(-27.3, -10.2)	12.962	13.554
10	(-15.9, -17.5)	(-26.6, -10.1)	(-26.5, -8.6)	12.962	13.860
11	(-15.9, -17.5)	(-26.6, -10.1)	(-26.3, -9.1)	12.962	13.286
12	(-15.9, -17.5)	(-26.6, -10.1)	(-25.3, -10.4)	12.962	11.730
13	(-15.9, -17.5)	(-25.3, -10.4)	(-23.9, -9.1)	11.730	11.522
14	(-15.9, -17.5)	(-23.9, -9.1)	(-24.5, -10.0)	11.522	11.361
15	(-15.9, -17.5)	(-24.5, -10.0)	(-23.9, -10.6)	11.361	10.515
16	(-15.9, -17.5)	(-23.9, -10.6)	(-22.7, -9.8)	10.515	10.261
17	(-15.9, -17.5)	(-22.7, -9.8)	(-22.6, -9.0)	10.261	10.811
18	(-15.9, -17.5)	(-22.7, -9.8)	(-23.6, -9.7)	10.261	10.930
19	(-15.9, -17.5)	(-22.7, -9.8)	(-20.9, -7.8)	9.943	10.470
20	(-16.0, -16.8)	(-22.7, -9.8)	(-24.4, -9.1)	9.693	11.425

21	(-16.0, -16.8)	(-22.7, -9.8)	(-21.0, -11.4)	9.693	7.362
22	(-16.0, -16.8)	(-22.7, -9.8)	(-19.6, -9.4)	7.362	8.239
23	(-16.0, -16.8)	(-22.7, -9.8)	(-21.0, -9.9)	7.362	8.514
24	(-16.0, -16.8)	(-22.7, -9.8)	(-21.9, -12.7)	7.362	7.205
25	(-16.0, -16.8)	(-21.9, -12.7)	(-22.6, -12.6)	7.205	7.849
26	(-16.0, -16.8)	(-21.9, -12.7)	(-23.5, -13.1)	7.205	8.422
27	(-16.0, -16.8)	(-21.9, -12.7)	(-19.9, -11.4)	7.205	6.623
28	(-16.0, -16.8)	(-21.9, -12.7)	(-19.2, -11.1)	6.623	6.521
29	(-16.0, -16.8)	(-19.2, -11.1)	(-20.6, -11.1)	6.521	7.333
30	(-16.0, -16.8)	(-19.2, -11.1)	(-17.7, -12.4)	6.521	4.719
31	(-16.0, -16.8)	(-17.7, -12.4)	(-19.5, -11.0)	4.719	6.751
32	(-16.0, -16.8)	(-17.7, -12.4)	(-18.4, -12.2)	4.719	5.143
33	(-16.0, -16.8)	(-17.7, -12.4)	(-15.8, -14.3)	4.719	2.525
34	(-16.0, -16.8)	(-15.8, -14.3)	(-16.3, -12.4)	2.525	4.360
35	(-16.0, -16.8)	(-16.3, -12.4)	(-17.7, -12.6)	2.525	4.498
36	(-16.0, -16.8)	(-16.3, -12.4)	(-17.6, -13.0)	2.525	4.090
37	(-16.0, -16.8)	(-16.3, -12.4)	(-15.9, -14.1)	2.525	2.678
38	(-16.0, -16.8)	(-16.3, -12.4)	(-17.4, -16.0)	2.525	1.649
39	(-16.0, -16.8)	(-16.3, -12.4)	(-16.3, -17.8)	1.649	1.103
40	(-15.6, -16.7)	(-16.3, -17.8)	(-15.3, -19.2)	1.315	2.510
41	(-15.2, -16.7)	(-16.3, -17.8)	(-17.8, -16.7)	1.601	2.615
42	(-14.8, -16.7)	(-16.3, -17.8)	(-14.7, -19.5)	1.927	2.825

43	(-14.2, -16.6)	(-16.3, -17.8)	(-16.2, -18.7)	2.495	2.954
44	(-13.8, -16.6)	(-16.3, -17.8)	(-15.2, -18.4)	2.790	2.278
45	(-13.4, -16.5)	(-15.2, -18.4)	(-13.6, -19.7)	2.622	3.156
46	(-13.0, -16.5)	(-15.2, -18.4)	(-13.7, -17.1)	2.938	0.916
47	(-13.0, -16.5)	(-15.2, -18.4)	(-13.2, -17.1)	0.916	0.714
48	(-13.0, -16.5)	(-13.2, -17.1)	(-13.3, -17.6)	0.714	1.225
49	(-13.0, -16.5)	(-13.2, -17.1)	(-13.8, -15.8)	0.714	1.091
50	(-13.0, -16.5)	(-13.2, -17.1)	(-14.1, -15.6)	0.714	1.410
51	(-13.0, -16.5)	(-13.2, -17.1)	(-15.2, -17.5)	0.714	2.477
52	(-13.0, -16.5)	(-13.2, -17.1)	(-12.9, -16.6)	0.714	0.144
53	(-13.0, -16.5)	(-12.9, -16.6)	(-10.9, -18.5)	0.144	2.844
54	(-13.0, -16.5)	(-12.9, -16.6)	(-12.4, -18.3)	0.144	1.913
55	(-13.0, -16.5)	(-12.9, -16.6)	(-13.0, -18.0)	0.144	1.490
56	(-13.0, -16.5)	(-12.9, -16.6)	(-12.6, -16.4)	0.144	0.385
57	(-13.0, -16.5)	(-12.9, -16.6)	(-12.8, -17.2)	0.144	0.754
58	(-13.0, -16.5)	(-12.9, -16.6)	(-14.5, -15.8)	0.144	1.637
59	(-13.0, -16.5)	(-12.9, -16.6)	(-14.0, -17.3)	0.144	1.258
60	(-13.0, -16.5)	(-12.9, -16.6)	(-12.4, -15.5)	0.144	1.165
61	(-13.0, -16.5)	(-12.9, -16.6)	(-11.9, -17.0)	0.144	1.202
62	(-13.0, -16.2)	(-12.9, -16.6)	(-12.2, -16.8)	0.452	1.018
63	(-13.0, -15.8)	(-12.9, -16.6)	(-11.4, -18.0)	0.850	2.776
64	(-13.1, -15.4)	(-12.9, -16.6)	(-12.5, -18.5)	1.169	3.071

65	(-13.1, -15.1)	(-12.9, -16.6)	(-13.7, -18.0)	1.489	2.893
66	(-13.1, -14.7)	(-12.9, -16.6)	(-11.8, -18.0)	1.889	3.552
67	(-13.2, -14.4)	(-12.9, -16.6)	(-14.5, -17.1)	2.208	3.035
68	(-13.2, -14.1)	(-12.9, -16.6)	(-12.5, -16.0)	2.528	1.987
69	(-13.2, -13.7)	(-12.5, -16.0)	(-10.6, -16.1)	2.377	3.573
70	(-13.2, -13.7)	(-12.5, -16.0)	(-14.4, -16.2)	2.377	2.780
71	(-13.2, -13.7)	(-12.5, -16.0)	(-12.8, -14.3)	2.377	0.748
72	(-13.2, -13.7)	(-12.8, -14.3)	(-14.2, -15.9)	0.748	2.413
73	(-13.3, -13.4)	(-12.8, -14.3)	(-14.1, -14.3)	1.032	1.250
74	(-13.3, -13.0)	(-12.8, -14.3)	(-14.3, -15.9)	1.409	3.123
75	(-13.3, -12.5)	(-14.3, -15.9)	(-12.9, -13.9)	1.874	1.417
76	(-13.3, -12.5)	(-12.9, -13.9)	(-10.9, -12.9)	1.417	2.448
77	(-13.3, -12.5)	(-12.9, -13.9)	(-14.1, -12.5)	1.417	0.764
78	(-13.4, -12.1)	(-14.1, -12.5)	(-12.4, -13.7)	0.835	1.891
79	(-13.4, -11.7)	(-14.1, -12.5)	(-12.9, -13.3)	1.064	1.644
80	(-13.5, -11.4)	(-14.1, -12.5)	(-14.0, -12.1)	1.308	0.902
81	(-13.5, -11.4)	(-14.0, -12.1)	(-13.9, -12.1)	0.902	0.782
82	(-13.5, -11.0)	(-13.9, -12.1)	(-13.1, -12.2)	1.130	1.300
82	(-13.5, -10.6)	(-13.9, -12.1)	(-13.0, -13.7)	1.504	3.176
84	(-13.3, -10.3)	(-13.9, -12.1)	(-13.4, -12.4)	1.871	2.102
85	(-13.3, -9.9)	(-13.9, -12.1)	(-14.0, -13.3)	2.246	3.542
86	(-13.4, -9.5)	(-13.9, -12.1)	(-15.3, -11.4)	2.626	2.687

87	(-13.4, -9.2)	(-13.9, -12.1)	(-14.7, -10.7)	2.935	2.008
88	(-13.4, -8.8)	(-14.7, -10.7)	(-12.9, -11.6)	2.304	2.905
89	(-13.5, -8.4)	(-14.7, -10.7)	(-13.1, -10.1)	2.628	1.805
90	(-13.5, -8.0)	(-13.1, -10.1)	(-12.7, -9.6)	2.125	1.782
91	(-13.5, -8.0)	(-12.7, -9.6)	(-11.6, -11.4)	1.782	3.851
92	(-13.5, -8.0)	(-12.7, -9.6)	(-13.6, -7.4)	0.725	0.617
93	(-13.5, -8.0)	(-13.6, -7.4)	(-13.0, -7.4)	0.617	0.757
94	(-13.5, -8.0)	(-13.6, -7.4)	(-15.6, -7.2)	0.617	2.246
95	(-13.5, -8.0)	(-13.6, -7.4)	(-13.3, -8.9)	0.617	0.917
96	(-13.5, -8.0)	(-13.6, -7.4)	(-12.3, -6.6)	0.617	1.851
97	(-13.5, -8.0)	(-13.6, -7.4)	(-12.9, -8.6)	0.617	0.821
98	(-13.5, -8.0)	(-13.6, -7.4)	(-14.6, -8.0)	0.617	1.137
99	(-13.5, -8.0)	(-13.6, -7.4)	(-13.9, -6.8)	0.617	1.331
100	(-13.5, -8.0)	(-13.6, -7.4)	(-15.5, -6.2)	0.617	2.702

Data *agent* 1 meliputi data iterasi, posisi *player*, posisi awal *agent*, posisi baru *agent*, *fitness* awal *agent*, dan *fitness* baru *agent*. *Fitness* yang diberi warna hijau adalah *fitness* yang dipilih oleh *agent*. *Fitness* tersebut menunjukan posisi *agent* 1 dekat dengan posisi *player* (target). *Fitness* yang diberi warna hitam adalah *fitness* posisi yang diabaikan oleh *agent* 1. Sedangkan yang berwarna merah adalah *fitness* yang dipilih oleh *agent* 1, namun posisi tersebut adalah posisi yang salah (bukan posisi terdekat dengan target). *Agent* 1 mendapatkan

fitness terbaiknya pada iterasi ke-47 dengan koordinat *agent* 1 : (-15.2, -18.4) dan *player* : (-13.0, -16.5) serta *fitness* : 0,714.

Tabel 4.6 Data Hasil Uji Coba *Agent* 2

Iterasi	Posisi Player (x,z)	Posisi Awal *Agent* (x,z)	Posisi Baru *Agent* (x,z)	Fitness Awal	Fitness Baru
1	(-15.1, -17.4)	(-26.3, 0.4)	(-24.8, 13.9)	33.414	32.751
2	(-15.5, -17.5)	(-24.8, 13.9)	(-26.6, 15.1)	32.668	34.358
3	(-15.9, -17.5)	(-24.8, 13.9)	(-26.7, 12.1)	32.590	31.491
4	(-15.9, -17.5)	(-26.7, 12.1)	(-27.3, 12.7)	31.491	32.249
5	(-15.9, -17.5)	(-26.7, 12.1)	(-25.0, 11.0)	31.491	29.924
6	(-15.9, -17.5)	(-25.0, 11.0)	(-26.3, 11.8)	29.924	31.093
7	(-15.9, -17.5)	(-25.0, 11.0)	(-23.5, 10.1)	29.924	28.578
8	(-15.9, -17.5)	(-23.5, 10.1)	(-24.5, 11.9)	28.578	30.638
9	(-15.9, -17.5)	(-23.5, 10.1)	(-22.1, 10.1)	28.578	28.326
10	(-15.9, -17.5)	(-22.1, 10.1)	(-22.7, 10.8)	28.326	29.101
11	(-15.9, -17.5)	(-22.1, 10.1)	(-23.1, 11.2)	28.326	29.599
12	(-15.9, -17.5)	(-22.1, 10.1)	(-22.7, 9.3)	28.326	27.605
13	(-15.9, -17.5)	(-22.7, 9.3)	(-21.9, 10.6)	27.605	28.678
14	(-15.9, -17.5)	(-22.7, 9.3)	(-21.1, 9.5)	27.605	27.481
15	(-15.9, -17.5)	(-22.7, 9.3)	(-19.8, 8.2)	27.481	25.951
16	(-15.9, -17.5)	(-19.8, 8.2)	(-20.1, 6.4)	25.951	24.252
17	(-15.9, -17.5)	(-20.1, 6.4)	(-18.8, 8.0)	24.252	25.641
18	(-15.9, -17.5)	(-20.1, 6.4)	(-20.6, 5.2)	24.252	23.156

19	(-15.9, -17.5)	(-20.6, 5.2)	(-19.0, 6.7)	22.759	23.968
20	(-16.0, -16.8)	(-20.6, 5.2)	(-20.8, 3.3)	22.441	20.675
21	(-16.0, -16.8)	(-20.8, 3.3)	(-21.8, 5.3)	20.675	22.810
22	(-16.0, -16.8)	(-20.8, 3.3)	(-19.9, 3.6)	20.675	20.801
23	(-16.0, -16.8)	(-20.8, 3.3)	(-21.3, 1.4)	20.675	18.964
24	(-16.0, -16.8)	(-21.3, 1.4)	(-2 0.6)	18.964	17.903
25	(-16.0, -16.8)	(-2 0.6)	(-19.4, -0.6)	17.903	16.506
26	(-16.0, -16.8)	(-19.4, -0.6)	(-21.0, -0.3)	16.506	17.267
27	(-16.0, -16.8)	(-19.4, -0.6)	(-20.6, -0.1)	16.506	17.311
28	(-16.0, -16.8)	(-19.4, -0.6)	(-18.5, 0.5)	16.506	17.469
29	(-16.0, -16.8)	(-19.4, -0.6)	(-20.0, -1.7)	16.506	15.649
30	(-16.0, -16.8)	(-20.0, -1.7)	(-19.4, -0.1)	15.649	17.025
31	(-16.0, -16.8)	(-20.0, -1.7)	(-18.7, -0.5)	15.649	16.551
32	(-16.0, -16.8)	(-20.0, -1.7)	(-20.3, 0.1)	15.649	17.391
33	(-16.0, -16.8)	(-20.0, -1.7)	(-19.7, -1.2)	15.649	16.029
34	(-16.0, -16.8)	(-20.0, -1.7)	(-21.6, -2.6)	15.649	15.296
35	(-16.0, -16.8)	(-21.6, -2.6)	(-23.1, -2.2)	15.296	16.241
36	(-16.0, -16.8)	(-21.6, -2.6)	(-23.3, -2.8)	15.296	15.802
37	(-16.0, -16.8)	(-21.6, -2.6)	(-22.3, -3.4)	15.296	14.843
38	(-16.0, -16.8)	(-22.3, -3.4)	(-23.8, -4.7)	14.843	14.394
39	(-16.0, -16.8)	(-23.8, -4.7)	(-25.8, -4.6)	14.394	15.617
40	(-15.6, -16.7)	(-23.8, -4.7)	(-25.3, -6.3)	14.582	14.282

41	(-15.2, -16.7)	(-25.3, -6.3)	(-26.3, -5.5)	14.282	15.760
42	(-14.8, -16.7)	(-25.3, -6.3)	(-23.4, -5.8)	14.282	13.850
43	(-14.2, -16.6)	(-23.4, -5.8)	(-25.4, -7.0)	14.209	14.752
44	(-13.8, -16.6)	(-23.4, -5.8)	(-25.0, -7.8)	14.396	14.222
45	(-13.4, -16.5)	(-23.4, -5.8)	(-23.9, -9.7)	14.572	12.555
46	(-13.0, -16.5)	(-23.9, -9.7)	(-22.9, -10.5)	12.870	11.558
47	(-13.0, -16.5)	(-22.9, -10.5)	(-23.9, -11.6)	11.558	11.976
48	(-13.0, -16.5)	(-22.9, -10.5)	(-23.4, -9.8)	11.558	12.361
49	(-13.0, -16.5)	(-22.9, -10.5)	(-24.8, -9.8)	11.558	13.605
50	(-13.0, -16.5)	(-22.9, -10.5)	(-22.0, -12.1)	11.558	10.063
51	(-13.0, -16.5)	(-22.0, -12.1)	(-21.7, -13.6)	10.063	9.183
52	(-13.0, -16.5)	(-21.7, -13.6)	(-19.9, -14.0)	9.183	7.391
53	(-13.0, -16.5)	(-19.9, -14.0)	(-19.0, -15.2)	7.391	6.143
54	(-13.0, -16.5)	(-19.0, -15.2)	(-20.9, -15.5)	6.143	8.033
55	(-13.0, -16.5)	(-19.0, -15.2)	(-17.9, -13.7)	6.143	5.674
56	(-13.0, -16.5)	(-17.9, -13.7)	(-16.7, -14.6)	5.674	4.135
57	(-13.0, -16.5)	(-16.7, -14.6)	(-17.2, -13.2)	4.135	5.313
58	(-13.0, -16.5)	(-16.7, -14.6)	(-17.2, -13.2)	4.135	4.177
59	(-13.0, -16.5)	(-16.7, -14.6)	(-16.9, -15.0)	4.135	4.208
60	(-13.0, -16.5)	(-16.7, -14.6)	(-15.4, -16.5)	4.135	2.487
61	(-13.0, -16.5)	(-16.7, -14.6)	(-14.7, -17.9)	2.487	2.285
62	(-13.0, -16.2)	(-14.7, -17.9)	(-16.3, -17.1)	2.475	3.486

63	(-13.0, -15.8)	(-14.7, -17.9)	(-16.0, -16.3)	2.746	2.985
64	(-13.1, -15.4)	(-14.7, -17.9)	(-14.9, -18.4)	2.984	3.512
65	(-13.1, -15.1)	(-14.7, -17.9)	(-16.0, -17.9)	3.236	3.984
66	(-13.1, -14.7)	(-14.7, -17.9)	(-15.0, -18.2)	3.567	3.934
67	(-13.2, -14.4)	(-14.7, -17.9)	(-15.3, -19.2)	3.841	5.220
68	(-13.2, -14.1)	(-14.7, -17.9)	(-14.6, -18.4)	4.121	4.495
69	(-13.2, -13.7)	(-14.7, -17.9)	(-16.4, -17.9)	4.480	5.230
70	(-13.2, -13.7)	(-14.7, -17.9)	(-13.4, -16.8)	4.480	3.063
71	(-13.2, -13.7)	(-13.4, -16.8)	(-13.4, -15.9)	3.063	2.256
72	(-13.2, -13.7)	(-13.4, -16.8)	(-12.4, -14.9)	2.256	1.506
73	(-13.3, -13.4)	(-12.4, -14.9)	(-11.4, -16.3)	1.792	3.482
74	(-13.3, -13.0)	(-12.4, -14.9)	(-10.6, -16.8)	2.163	4.684
75	(-13.3, -12.5)	(-12.4, -14.9)	(-11.1, -13.4)	2.620	2.467
76	(-13.3, -12.5)	(-11.1, -13.4)	(-10.7, -13.2)	2.467	2.760
77	(-13.3, -12.5)	(-11.1, -13.4)	(-11.1, -15.2)	2.467	3.530
78	(-13.4, -12.1)	(-11.1, -13.4)	(-9.2, -12.8)	2.674	4.278
79	(-13.4, -11.7)	(-11.1, -13.4)	(-11.6, -11.7)	2.922	1.843
80	(-13.5, -11.4)	(-11.6, -11.7)	(-11.6, -12.6)	1.894	2.209
81	(-13.5, -11.4)	(-11.6, -11.7)	(-10.6, -13.4)	1.894	3.478
82	(-13.5, -11.0)	(-11.6, -11.7)	(-11.5, -9.9)	2.028	2.221
82	(-13.5, -10.6)	(-11.6, -11.7)	(-10.5, -11.8)	2.226	3.256
84	(-13.3, -10.3)	(-11.6, -11.7)	(-13.4, -12.9)	2.193	2.634

85	(-13.3, -9.9)	(-11.6, -11.7)	(-13.0, -13.0)	2.487	3.129
86	(-13.4, -9.5)	(-11.6, -11.7)	(-10.8, -10.3)	2.810	2.699
87	(-13.4, -9.2)	(-10.8, -10.3)	(-12.1, -8.8)	2.836	1.315
88	(-13.4, -8.8)	(-12.1, -8.8)	(-10.9, -8.2)	1.310	2.628
89	(-13.5, -8.4)	(-12.1, -8.8)	(-11.6, -9.2)	1.423	2.071
90	(-13.5, -8.0)	(-12.1, -8.8)	(-11.2, -10.1)	1.582	3.014
91	(-13.5, -8.0)	(-12.1, -8.8)	(-11.0, -8.0)	1.582	2.522
92	(-13.5, -8.0)	(-12.1, -8.8)	(-13.1, -6.9)	1.582	1.154
93	(-13.5, -8.0)	(-13.1, -6.9)	(-14.5, -6.6)	1.154	1.770
94	(-13.5, -8.0)	(-13.1, -6.9)	(-13.4, -5.9)	1.154	2.167
95	(-13.5, -8.0)	(-13.1, -6.9)	(-11.2, -8.3)	1.154	2.293
96	(-13.5, -8.0)	(-13.1, -6.9)	(-13.1, -8.3)	1.154	0.455
97	(-13.5, -8.0)	(-13.1, -8.3)	(-12.9, -8.4)	0.455	0.663
98	(-13.5, -8.0)	(-13.1, -8.3)	(-11.5, -6.4)	0.455	2.564
99	(-13.5, -8.0)	(-13.1, -8.3)	(-12.3, -7.9)	0.455	1.204
100	(-13.5, -8.0)	(-13.1, -8.3)	(-13.1, -8.9)	0.455	0.969

Sama halnya dengan data *agent* 1, *fitness* posisi *agent* 2 yang diberi warna hijau adalah *fitness* yang dipilih oleh *agent* yang menunjukan posisi terdekat dengan *player* (target), sedangkan *fitness* yang diberi warna hitam adalah *fitness* posisi yang diabaikan oleh *agent* 2. *Agent* 2 mendapatkan *fitness* terbaiknya pada

iterasi ke-96 dengan koordinat *agent* 2 : (-13.1, -8.3) dan *player* : (-1.3, -8.0) serta *fitness* : 0,455.

Tabel 4. 7 Data Hasil Uji Coba *Agent* 3

Iterasi	Posisi Player (x,z)	Posisi Awal *Agent* (x,z)	Posisi Baru *Agent* (x,z)	Fitness Awal	Fitness Baru
1	(-15.1, -17.4)	(-14.3, 0.4)	(-15.5, 1.3)	17.844	18.775
2	(-15.5, -17.5)	(-14.3, 0.4)	(-12.5, 0.6)	17.901	18.302
3	(-15.9, -17.5)	(-14.3, 0.4)	(-13.9, 1.7)	17.967	19.305
4	(-15.9, -17.5)	(-14.3, 0.4)	(-15.5, 1.7)	17.967	19.174
5	(-15.9, -17.5)	(-14.3, 0.4)	(-15.7, -0.3)	17.967	17.231
6	(-15.9, -17.5)	(-15.7, -0.3)	(-14.9, -2.0)	17.231	15.490
7	(-15.9, -17.5)	(-14.9, -2.0)	(-15.5, -2.6)	15.490	14.918
8	(-15.9, -17.5)	(-15.5, -2.6)	(-13.7, -3.0)	14.918	14.690
9	(-15.9, -17.5)	(-13.7, -3.0)	(-14.7, -1.8)	14.690	15.780
10	(-15.9, -17.5)	(-13.7, -3.0)	(-15.0, -3.5)	14.690	14.033
11	(-15.9, -17.5)	(-15.0, -3.5)	(-15.8, -5.2)	14.033	12.252
12	(-15.9, -17.5)	(-15.8, -5.2)	(-14.7, -3.3)	12.252	14.255
13	(-15.9, -17.5)	(-15.8, -5.2)	(-14.4, -5.4)	12.252	12.226
14	(-15.9, -17.5)	(-15.8, -5.2)	(-15.0, -7.1)	12.226	10.407
15	(-15.9, -17.5)	(-15.0, -7.1)	(-16.8, -7.3)	10.407	10.254
16	(-15.9, -17.5)	(-16.8, -7.3)	(-15.3, -8.1)	10.254	9.380
17	(-15.9, -17.5)	(-15.3, -8.1)	(-16.1, -7.9)	9.380	9.644
18	(-15.9, -17.5)	(-15.3, -8.1)	(-15.1, -6.5)	9.380	11.060

19	(-15.9, -17.5)	(-15.3, -8.1)	(-13.5, -8.7)	8.984	8.772
20	(-16.0, -16.8)	(-13.5, -8.7)	(-15.4, -7.8)	8.475	9.006
21	(-16.0, -16.8)	(-13.5, -8.7)	(-11.5, -7.6)	8.475	10.163
22	(-16.0, -16.8)	(-13.5, -8.7)	(-14.9, -8.0)	8.475	8.860
23	(-16.0, -16.8)	(-13.5, -8.7)	(-15.2, -9.6)	8.475	7.223
24	(-16.0, -16.8)	(-15.2, -9.6)	(-17.0, -7.7)	7.223	9.116
25	(-16.0, -16.8)	(-15.2, -9.6)	(-17.0, -7.7)	7.223	9.123
26	(-16.0, -16.8)	(-15.2, -9.6)	(-14.8, -10.9)	7.223	5.989
27	(-16.0, -16.8)	(-14.8, -10.9)	(-16.7, -10.0)	5.989	6.815
28	(-16.0, -16.8)	(-14.8, -10.9)	(-13.4, -11.7)	5.989	5.675
29	(-16.0, -16.8)	(-13.4, -11.7)	(-13.8, -10.9)	5.675	6.245
30	(-16.0, -16.8)	(-13.4, -11.7)	(-15.0, -13.5)	5.675	3.460
31	(-16.0, -16.8)	(-15.0, -13.5)	(-16.6, -13.7)	3.460	3.157
32	(-16.0, -16.8)	(-16.6, -13.7)	(-15.8, -15.6)	3.157	1.160
33	(-16.0, -16.8)	(-15.8, -15.6)	(-15.0, -17.4)	1.160	1.130
34	(-16.0, -16.8)	(-15.0, -17.4)	(-15.2, -19.3)	1.130	2.647
35	(-16.0, -16.8)	(-15.0, -17.4)	(-14.3, -18.0)	1.130	2.037
36	(-16.0, -16.8)	(-15.0, -17.4)	(-15.2, -17.8)	1.130	1.276
37	(-16.0, -16.8)	(-15.0, -17.4)	(-16.0, -16.3)	1.130	0.437
38	(-16.0, -16.8)	(-16.0, -16.3)	(-14.0, -17.0)	0.437	1.993
39	(-16.0, -16.8)	(-16.0, -16.3)	(-16.3, -16.3)	0.437	0.598
40	(-15.6, -16.7)	(-16.0, -16.3)	(-14.5, -14.9)	0.551	2.115

41	(-15.2, -16.7)	(-16.0, -16.3)	(-16.0, -15.6)	0.858	1.392
42	(-14.8, -16.7)	(-16.0, -16.3)	(-17.5, -14.9)	1.220	3.286
43	(-14.2, -16.6)	(-16.0, -16.3)	(-17.8, -17.6)	1.833	3.807
44	(-13.8, -16.6)	(-16.0, -16.3)	(-14.9, -16.4)	2.145	1.088
45	(-13.4, -16.5)	(-14.9, -16.4)	(-16.0, -18.4)	1.559	3.216
46	(-13.0, -16.5)	(-14.9, -16.4)	(-15.2, -15.6)	1.955	2.454
47	(-13.0, -16.5)	(-14.9, -16.4)	(-14.5, -14.7)	1.955	2.353
48	(-13.0, -16.5)	(-14.9, -16.4)	(-15.5, -16.4)	1.955	2.576
49	(-13.0, -16.5)	(-14.9, -16.4)	(-13.3, -18.1)	1.955	1.628
50	(-13.0, -16.5)	(-13.3, -18.1)	(-11.4, -19.5)	1.628	3.384
51	(-13.0, -16.5)	(-13.3, -18.1)	(-14.8, -18.6)	1.628	2.846
52	(-13.0, -16.5)	(-13.3, -18.1)	(-11.5, -16.5)	1.628	1.422
53	(-13.0, -16.5)	(-11.5, -16.5)	(-9.8, -16.7)	1.422	3.179
54	(-13.0, -16.5)	(-11.5, -16.5)	(-12.7, -15.9)	1.422	0.605
55	(-13.0, -16.5)	(-12.7, -15.9)	(-12.4, -16.9)	0.605	0.723
56	(-13.0, -16.5)	(-12.7, -15.9)	(-14.6, -15.1)	0.605	2.097
57	(-13.0, -16.5)	(-12.7, -15.9)	(-10.8, -15.6)	0.605	2.338
58	(-13.0, -16.5)	(-12.7, -15.9)	(-12.8, -14.4)	0.605	2.054
59	(-13.0, -16.5)	(-12.7, -15.9)	(-12.2, -14.0)	0.605	2.568
60	(-13.0, -16.5)	(-12.7, -15.9)	(-12.4, -15.5)	0.605	1.165
61	(-13.0, -16.5)	(-12.7, -15.9)	(-13.1, -17.5)	0.605	0.981
62	(-13.0, -16.2)	(-12.7, -15.9)	(-11.7, -16.7)	0.605	1.298

63	(-13.0, -15.8)	(-12.7, -15.9)	(-11.6, -17.2)	0.370	1.725
64	(-13.1, -15.4)	(-12.7, -15.9)	(-13.2, -17.0)	0.373	1.262
65	(-13.1, -15.1)	(-12.7, -15.9)	(-11.0, -16.5)	0.609	2.344
66	(-13.1, -14.7)	(-12.7, -15.9)	(-14.4, -16.8)	0.898	2.131
67	(-13.2, -14.4)	(-12.7, -15.9)	(-12.2, -17.7)	1.281	3.087
68	(-13.2, -14.1)	(-12.7, -15.9)	(-11.5, -16.6)	1.593	2.754
69	(-13.2, -13.7)	(-12.7, -15.9)	(-13.3, -16.5)	1.908	2.424
70	(-13.2, -13.7)	(-12.7, -15.9)	(-12.6, -16.4)	2.303	2.766
71	(-13.2, -13.7)	(-12.7, -15.9)	(-10.9, -15.6)	2.303	3.028
72	(-13.2, -13.7)	(-12.7, -15.9)	(-12.9, -17.9)	2.303	4.230
73	(-13.3, -13.4)	(-12.7, -15.9)	(-13.8, -17.3)	2.303	3.696
74	(-13.3, -13.0)	(-12.7, -15.9)	(-14.2, -17.0)	2.621	3.714
75	(-13.3, -12.5)	(-12.7, -15.9)	(-11.0, -17.8)	3.018	5.319
76	(-13.3, -12.5)	(-12.7, -15.9)	(-12.2, -14.2)	3.496	2.053
77	(-13.3, -12.5)	(-12.7, -15.9)	(-11.2, -15.9)	3.496	4.000
78	(-13.4, -12.1)	(-12.2, -14.2)	(-13.4, -13.7)	2.053	1.201
79	(-13.4, -11.7)	(-13.4, -13.7)	(-12.8, -14.2)	1.599	2.206
80	(-13.5, -11.4)	(-13.4, -13.7)	(-13.6, -15.4)	1.998	3.729
81	(-13.5, -11.4)	(-13.4, -13.7)	(-15.0, -15.0)	2.317	3.881
82	(-13.5, -11.0)	(-13.4, -13.7)	(-13.0, -13.5)	2.317	2.151
82	(-13.5, -10.6)	(-13.0, -13.5)	(-13.0, -12.6)	2.548	1.710
84	(-13.3, -10.3)	(-13.0, -12.6)	(-13.8, -11.9)	2.104	1.386

85	(-13.3, -9.9)	(-13.8, -11.9)	(-15.1, -10.3)	1.753	1.765
86	(-13.4, -9.5)	(-13.8, -11.9)	(-14.0, -10.7)	2.128	1.056
87	(-13.4, -9.2)	(-14.0, -10.7)	(-13.2, -10.4)	1.365	0.969
88	(-13.4, -8.8)	(-13.2, -10.4)	(-11.3, -11.1)	1.289	2.822
89	(-13.5, -8.4)	(-13.2, -10.4)	(-12.7, -11.7)	1.688	3.068
90	(-13.5, -8.0)	(-13.2, -10.4)	(-11.3, -11.4)	2.088	3.741
91	(-13.5, -8.0)	(-13.2, -10.4)	(-12.9, -10.4)	2.408	2.393
92	(-13.5, -8.0)	(-12.9, -10.4)	(-14.2, -12.3)	2.393	4.287
93	(-13.5, -8.0)	(-12.9, -10.4)	(-13.3, -9.9)	2.393	1.849
94	(-13.5, -8.0)	(-13.3, -9.9)	(-11.7, -11.5)	1.849	3.919
95	(-13.5, -8.0)	(-13.3, -9.9)	(-13.9, -11.1)	1.849	3.099
96	(-13.5, -8.0)	(-13.3, -9.9)	(-13.0, -11.0)	1.849	2.962
97	(-13.5, -8.0)	(-13.3, -9.9)	(-13.6, -11.8)	1.849	3.748
98	(-13.5, -8.0)	(-13.3, -9.9)	(-13.5, -10.7)	1.849	2.657
99	(-13.5, -8.0)	(-13.3, -9.9)	(-14.2, -8.8)	1.849	1.085
100	(-13.5, -8.0)	(-14.2, -8.8)	(-12.8, -8.7)	1.085	0.989

Agent 3 mendapatkan *fitness* terbaiknya pada iterasi ke-37 dengan koordinat *agent* 3 : (-16.0, -16.3) dan *player* : (-16.0, -16.8) serta *fitness* : 0,437.

Setiap *agent* mencapai *fitness* terbaik pada iterasi yang berbeda-beda dari 100 iterasi yang dilakukan. Dari data-data diatas, *agent* dapat memilih posisi yang terdekat dengan posisi *player* (target) dengan cara membandingkan dan memilih

fitness yang lebih kecil di antara kedua nilai *fitness* (*fitness* posisi awal dan *fitness* posisi baru). Dari data tabel di atas dilakukan evaluasi untuk mengetahui keakuratan dari algoritma yang diterapkan kepada NPC, sehingga sesuai dengan tujuan dari penelitian ini. Untuk menghitung akurasi dan *error* dapat digunakan rumus dibawah ini (Taufik Fuadi, 2013) :

$$\text{Akurasi} = \frac{Jumlah\ data\ yang\ sesuai}{jumlah\ seluruh\ data} \times 100\%$$

$$Error = \frac{Jumlah\ data\ yang\ tidak\ sesuai}{jumlah\ seluruh\ data} \times 100\%$$

Jumlah data yang sesuai pada perhitungan ini sama dengan jumlah *fitness* yang dipilih oleh *agent* yang merupakan *fitness* posisi terdekat dengan target (warna hijau), sedangkan jumlah data yang tidak sesuai adalah jumlah *fitness* yang dipilih oleh *agent*, namun bukan *fitness* posisi yang terdekat dengan target (warna merah). Nilai akurasi dan nilai *error* dari masing-masing *agent* adalah sebagai berikut :

- ***Agent* 1**

  Jumlah data yang sesuai = 97

  Jumlah data yang tidak sesuai = 3

  Akurasi = $\frac{97}{100} \times 100\% = 97\%$

  Error = $\frac{3}{100} \times 100\% = 3\ \%$

- ***Agent* 2**

  Jumlah data yang sesuai = 97

Jumlah data yang tidak sesuai = 3

Akurasi = $\frac{97}{100} \times 100\% = 97\ \%$

Error = $\frac{3}{100} \times 100\% = 3\ \%$

- ***Agent* 3**

  Jumlah data yang sesuai = 98

  Jumlah data yang tidak sesuai = 2

  Akurasi = $\frac{98}{100} \times 100\% = 98\ \%$

  Error = $\frac{2}{100} \times 100\% = 2\ \%$

4.3 Spesifikasi Sistem

4.3.1 Spesifikasi Perangkat Keras

Berikut adalah spesifikasi perangkat keras yang digunakan oleh peneliti untuk merancang *game* ini :

Tabel 4.8 Spesifikasi Perangkat Keras

No.	Perangkat Keras	Spesifikasi
1	*Processor*	AMD A6-3420M APU
2	VGA	AMD *Radeon* HD 6520G
3	RAM	DDR3 6GB
4	HDD	500GB
5	*Monitor*	16”
6	*Speaker*	*ON*
7	*Mouse & Keyboard*	*ON*

4.3.2 Spesifikasi Perangkat Lunak

Berikut adalah beberapa perangkat lunak yang digunakan oleh peneliti untuk merancang *game* ini :

Tabel 4.9 Spesifikasi Perangkat Lunak

No.	Perangkat Lunak	Spesifikasi
1	Sistem Operasi	*Windows* 10
2	*Game Engine*	*Unity* 5.5.2f
3	*Script Writer*	*MonoDevelop*
4	Desain 2D	• *Adobe Photoshop* • *Adobe Illustrator*

4.4 Implementasi Algoritma dalam *C#*

Algoritma *Artificial Bee Colony* dalam penelitian ini akan diimplementasikan dengan Bahasa C# dengan *source code* sebagai berikut :

```
//INISIALISASI POSISI
   posPlayerX = target.position.x;
   posPlayerY = target.position.y;
   posPlayerZ = target.position.z;
   posPlayer = new Vector3 (posPlayerX, posPlayerY,
                posPlayerZ);
```

posPlayerX, *posPlayerY*, dan *posPlayerZ* adalah variabel *float* untuk mengambil nilai posisi *player* (target) pada koordinat X, Y, Z. Sedangkan *posPlayer* adalah variabel vektor 3 dimensi untuk menyimpan nilai posisi *player*

yang dihasilkan dari *posPlayerX*, *posPlayerY*, dan *posPlayerZ* pada vektor 3 dimensi. Selanjutnya adalah sintaks untuk mengambil nilai posisi *agent* :

```
posEnemyAwalX = transform.position.x;
posEnemyAwalY = transform.position.y;
posEnemyAwalZ = transform.position.z;
posEnemy = new Vector3 (posEnemyAwalX, posEnemyAwalY,
          posEnemyAwalZ);
```

Sama halnya dengan *player*, *agent* mempunya variabel *float* yang terdiri dari *posEnemyAwalX*, *posEnemyAwalX*, dan *posEnemyAwalX* yang berfungsi untuk mengambil nilai posisi *agent* saat ini pada koordinat X, Y, Z. Sedangkan *posEnemy* adalah variabel vektor 3 dimensi untuk menyimpan nilai posisi *agent* yang dihasilkan dari *posEnemyAwalX*, *posEnemyAwalX*, dan *posEnemyAwalX*. Sintaks diatas adalah sintaks pada fase inisialisai algoritma *bee colony*.

Fase berikutnya adalah fase *Employed*, dimana *agent* akan mencari posisi baru secara acak sebagai posisi yang akan ditempati selanjutnya jika memenuhi syarat pada fase *Onlooker* menggunakan rumus : $\boldsymbol{X'ij = Xij + (rand[-1,1] \times R)}$. Penerapan rumus pada *game* ini adalah sebagai berikut :

```
randomX = Random.Range (-1.0f, 1.0f);
posEnemyBaruX = posEnemyAwalX + (randomX * colliderEnemy);

randomZ = Random.Range (-1.0f, 1.0f);
posEnemyBaruZ = posEnemyAwalZ + (randomZ * colliderEnemy);
posEnemyBaru = new Vector3 (posEnemyBaruX, posEnemyAwalY,
          posEnemyBaruZ);
```

randomX dan *randomZ* adalah variabel *float* untuk menyimpan nilai acak mulai dari -1 sampai dengan 1 untuk koordinat X, Z. Sedangkan *posEnemyBaruX* dan *posEnemyBaruZ* merupakan variabel *float* sebagai rumus untuk mencari posisi baru pada koordinat X, Z. *colliderEnemy* adalah nilai jarak *neighbor* pada masing-masing koordinat posisi *agent*.

Setelah didapatkan posisi baru *agent*, tahap berikutnya adalah menyeleksi apakah posisi baru tersebut adalah posisi yang dekat dengan posisi target dengan cara menghitung *fitness* masing-masing posisi, baik itu posisinya saat ini maupun posisi baru. Tahap ini disebut dengan fase *Onlooker*. Untuk menghitung nilai *fitness* maka digunakan rumus : $\boldsymbol{d = \sqrt{(X_2 - X_1)^2 + (Z_2 - Z_1)^2}}$. Di bawah ini adalah implementasi rumus pada *game* yang dirancang.

```
float Fitness(float Xj, float Xi,float Zj,float Zi)
{
    return(Mathf.Sqrt (((Xj - Xi) * (Xj - Xi)) + ((Zj - Zi)
    * (Zj - Zi))));
}
```

Float Fitness adalah sebuah *method* untuk menghitung nilai *fitness* seperti pada rumus yang terdiri dari *Xj* sebagai variabel posisi target di koordinat X, *Xi* sebagai variabel posisi *agent* di koordinat X, *Zj* sebagai variabel posisi target di koordinat Z, dan *Zi* sebagai variabel posisi *agent* di koordinat Z. Sedangkan *Mathf.Sqrt* adalah fungsi untuk menentukan nilai akar *float* pada bahasa C#.

4.5 Implementasi Aplikasi *Game*

Pada sub bab ini peneliti akan memberikan gambar dari *scene* yang telah dirancang sesuai dengan desain *interface game* yang terdapat pada bab 3.

4.5.1 *Scene Main Menu*

Sebelum menu utama ditampilkan terdapat *splash screen* yang akan muncul beberapa detik sebagai pembuka halaman *game*. *Splash screen* akan menampilkan beberapa gambar logo yang terdiri dari logo *developer*, logo universitas (Universitas Islam Negeri Maulana Malik Ibrahim Malang), dan logo *game engine (Unity)*. Berikut adalah tampilan dari *splash screen* :

a) Logo *Developer*

b) Logo Universitas Islam Negeri Maulana Malik Ibrahim Malang

Gambar 4. 6 Tampilan *Splash Screen*

Scene main menu adalah tampilan kedua dari *game* setelah *splash screen*, *scene* ini menampilkan beberapa tombol pilihan yang terdiri dari

tombol *Mulai* untuk memulai permainan, tombol *Pengaturan* untuk mengatur suara maupun grafik, dan tombol *Keluar* untuk keluar dari permainan.

Gambar 4.7 Tampilan *Menu* Utama

Berikut adalah tampilan ketika *player* menekan tombol *Pengaturan*. *Player* boleh mengubah *setting* sesuai dengan keinginannya.

Gambar 4.8 Tampilan Pengaturan pada *Main Menu*

4.5.2 *Scene Gameplay*

Pada *scene gameplay* terdapat beberapa keadaan ketika *player* memulai permainan. Berikut ini adalah tampilan ketika *player* pertama

kali memasuki *level* permainan. Hal yang pertama yang dilihat oleh *player* dalam *scene* ini adalah informasi tentang tokoh yang menjadi misi dari permainan ini.

Gambar 4.9 Tampilan *Gameplay*

Pada permainan ini terdapat *item* yang harus dikumpulkan oleh *player* yaitu *item* huruf. *Item* huruf merupakan *puzzle* huruf untuk melengkapi huruf-huruf yang kosong pada misi. Setiap *item* bernilai 100 poin. Untuk mempermudah *player* menemukan *item* tersebut, terdapat *minimap* yang terletak di pojok kanan atas untuk mengetahui *item* yang harus dikumpulkan. Berikut adalah tampilan *item* huruf pada permainan.

Gambar 4.10 Tampilan *Item* Huruf

Jika *item* yang dikumpulkan adalah *item* yang sesuai pada *minimap*, maka setiap *item*-nya akan menampilkan informasi tentang karya-karya tokoh tersebut selama masa hidupnya.

Gambar 4.11 Tampilan Informasi tentang Karya Tokoh

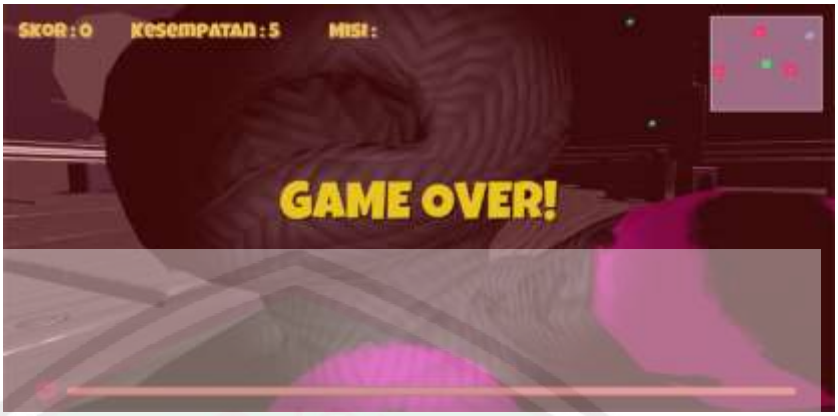
Namun jika *item* yang dikumpulkan tidak sesuai pada *minimap*, maka kesempatan akan dikurangi 1 dan tampilannya akan seperti gambar di bawah ini.

Gambar 4.12 Tampilan Salah Mengambil *Item*

Terdapat musuh yang menghalangi *player* untuk menyelesaikan misi. Ketika musuh menyentuh *player* maka *health point player* akan berkurang. *Player* dapat menghindar agar *health point* tidak berkurang. Berikut adalah tampilan ketika *player* diserang oleh musuh.

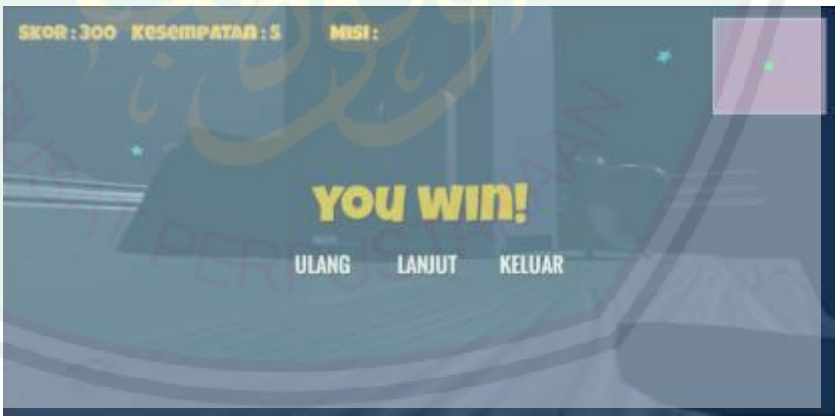
Gambar 4.13 Tampilan Musuh Menyerang *Player*

Jika *healt point player* habis maka secara otomatis akan ditampilkan *panel Game Over*. Selain itu *panel game over* juga akan muncul jika *player* telah kehabisan kesempatan.



Gambar 4.14 Tampilan *Panel Game Over*

Ketika *player* berhasil mengumpulkan semua *item* yang sesuai pada *minimap*, maka akan ditampilkan *panel Win*. Terdapat 3 tombol pada *panel win* yaitu tombol *ulang* untuk mengulang permainan pada *level* yang sama, tombol *lanjut* untuk melanjutkan permainan pada *level* berikutnya dan tombol *keluar* untuk keluar dari halaman permainan.



Gambar 4.15 Tampilan *Panel Win*

Tombol *Escape (ESC)* pada *keyboard* berfungsi untuk menghentikan permainan sejenak atau yang biasa disebut dengan tombol *pause*.

Gambar 4.16 Tampilan *Pause*

Terdapat tambahan pada menu pengaturan ketika kita menekan tombol *pause* yaitu pengaturan grafik untuk memilih kualitas grafik yang terdiri dari *High, Medium, dan Low.* Di bawah ini adalah tampilannya.

Gambar 4.17 Tampilan *Menu* Pengaturan

4.6 Uji Coba *Game*

Uji coba kali ini dilakukan untuk mengetahui apakah *game* dapat dijalankan pada perangkat PC lain dengan baik atau tidak, *game* akan di-*instal* pada beberapa perangkat PC berbeda dan akan dimainkan serta akan diamati

bagaimana kinerja dan kelancaran *game* pada perangkat tersebut. Berikut adalah hasil uji coba pada 4 perangkat PC dengan tipe *system* operasi yang berbeda.

Tabel 4.10 Uji Coba pada *Device* yang Berbeda

No	OS	Layar	Spesifikasi	RAM	Keterangan
1	*Windows* 10	15.6''	*Intel Core* i3 gen4 3.5 Ghz Nvidia 740M 2GB	8 GB	Tampilan menu berjalan dengan baik. Tombol berfungsi dengan baik. Kualitas Grafik *High*.
2	*Windows* 8.1	24''	*Intel Core* i7-6700K 4.0 GHz GeForce GTX 1060 3GB	16 GB	Tampilan menu berjalan dengan baik. Tombol berfungsi dengan baik. Kualitas Grafik *High*.
3	*Windwos* 8	15.6''	*Intel Core* i7-6700 2.6GHz GeForce GTX 950M 4GB	4GB	Tampilan menu berjalan dengan baik. Tombol berfungsi dengan baik. Kualitas Grafik *High*.
4	*Windows* 7	14''	*Intel Core* i5-4210U 2,7 GHz GeForce GT 920M 2GB	4GB	Tampilan menu berjalan dengan baik. Tombol berfungsi dengan baik. Kualitas Grafik *High*.

Dari pengujian yang dilakukan sebanyak 4 kali pada berbagai *platform Windows* yang memiliki sistem operasi dan spesifikasi berbeda. Dapat diketahui presentase pengujian pada **Tabel 4.11** di bawah ini.

Tabel 4. 11 Persentase Pengujian *Game*

No	Jenis Pengujian	Jumlah Yang Dapat Digunakan	Jumlah Seluruh	Persentase = (Jumlah Yang Dapat Digunakan / Jumlah Seluruh) * 100%
1	Sistem	4	4	(4/4) x 100 = 100%
2	Tombol	24	24	(24/24) x 100 = 100%

3	Tampilan	4	4	(4/4) x 100 = 100%

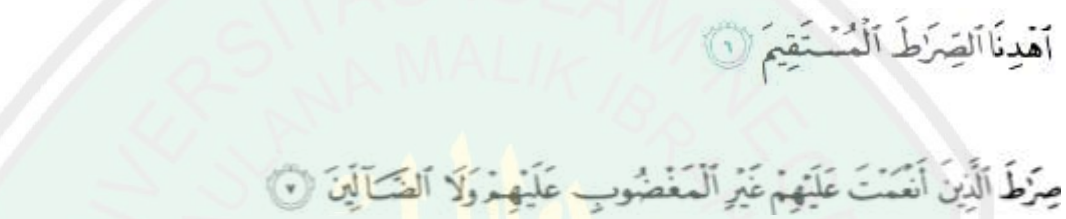
Keterangan :

Tabel 4.12 merupakan tabel yang berisi hasil pengujian *game* terhadap 4 sistem operasi *Windows* desktop yang telah dijelaskan pada **Tabel 4.11**. Hasil persentase yang didapatkan dari pengujian adalah 100% *game* dapat berjalan dengan baik pada *devices* tersebut. *Devices* tersebut bukan menunjukan *specification requirement* dari *game* ini melainkan hasil uji coba saja, untuk membuktikan apakah game ini dapat berjalan pada perangkat PC seperti demikian.

4.7 Integrasi Keislaman

Al-Qur'an merupakan pedoman dan petunjuk paling utama dan fundamental bagi umat Islam. Sebagai pedoman, Al-Qur'an adalah rujukan bagi umat untuk membentuk pribadi, kelompok maupun bangsa untuk diakui mulia oleh Sang Pemilik alam semesta. Dampak kemuliaan hidup manusia tidak hanya akan dirasakan oleh individu-individu tertentu saja, melainkan juga oleh seluruh alam semesta kecuali yang menentangnya. Al-Qur'an mengajarkan tentang cara hidup yang baik yang disajikan dalam bentuk kisah atau sejarah umat-umat terdahulu. Mempelajari kisah atau sejarah tokoh-tokoh pada masa lampau merupakan hal yang sangat penting bagi keberlangsungan umat manusia, karena di dalamnya terdapat kejadian yang memiliki pelajaran yang bisa dipetik oleh umat yang hidup di era sekarang untuk membenahi kekurangan dan kesalahan guna meraih kemuliaan dunia dan akhirat.

Dalam al-Qur'an, surat yang pertama adalah surat Al-Fatihah, yang merupakan bagian pembuka atau pendahuluan. Surat ini merupakan surat yang paling sering dibaca, bahkan surat ini biasanya dihafal terlebih dahulu oleh masyarakat sebelum surat-surat yang lain. Pesan tersirat tentang perintah untuk belajar sejarah sangat kuat terlihat dalam surat Al-Fatihah ini. Hal tersebut tampak pada ayat 6 dan 7 sebagai berikut :


اهْدِنَا الصِّرَاطَ الْمُسْتَقِيمَ ٦

صِرَاطَ الَّذِينَ أَنْعَمْتَ عَلَيْهِمْ غَيْرِ الْمَغْضُوبِ عَلَيْهِمْ وَلَا الضَّالِّينَ ٧

"Tunjukilah kami jalan yang lurus. (yaitu) Jalan orang-orang yang telah Engkau beri nikmat kepada mereka; bukan (jalan) mereka yang dimurkai dan bukan (pula jalan) mereka yang sesat."

Pada ayat yang ketujuh dari surat Al-Fatihah ini perintah tersirat untuk belajar sejarah itu bisa kita dapatkan. Ada tiga kelompok yang disebutkan dalam ayat terakhir ini: (1) Kelompok yang telah diberi nikmat oleh Allah; (2) Kelompok yang dimurkai Allah (3) Kelompok yang sesat. Ketiga kelompok ini adalah generasi yang telah berlalu, generasi di masa lalu yang telah mendapatkan satu dari ketiga hal tersebut.

Imam Ibnu Katsir dalam tafsirnya, menjelaskan bahwa kelompok pertama (yang diberi nikmat oleh Allah) adalah: para Nabi, para *shiddiqin*, para *syuhada'* dan para *shalihin*, semua yang hadir dalam dalam do'a, mereka yang telah meninggal, yang dijelaskan lebih detail dalam Surat An-Nisa: 69-70.

"Dan barangsiapa yang mentaati Allah dan Rasul(Nya), mereka itu akan bersama-sama dengan orang-orang yang dianugerahi nikmat oleh Allah, yaitu: Nabi-nabi, para shiddiiqiin, orang-orang yang mati syahid, dan orang-orang saleh. Dan mereka itulah teman yang sebaik-baiknya. Yang demikian itu adalah karunia dari Allah, dan Allah cukup mengetahui." (Q.S. An-Nisa : 69-70).

Ini adalah pesan tersirat pertama agar umat Islam rajin melihat sejarah hidup mereka. Untuk tahu dan bisa meneladani mereka, agar bisa mengetahui nikmat seperti apakah yang mereka rasakan sepanjang hidupnya, sehingga bisa mengikuti jalan lurus yang pernah mereka tempuh sekaligus bisa merasakan nikmat yang telah mereka rasakan. Hebatnya perjalanan hidup mereka tercatat rapi dalam sejarah.

Selanjutnya ayat lain yang memerintahkan untuk mempelajari sejarah terdapat dalam surah Yusuf ayat 111.

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

"Sesungguhnya pada kisah-kisah mereka itu terdapat pengajaran bagi orang-orang yang mempunyai akal. Al Quran itu bukanlah cerita yang dibuat-buat, akan tetapi membenarkan (kitab-kitab) yang sebelumnya dan menjelaskan segala sesuatu, dan sebagai petunjuk dan rahmat bagi kaum yang beriman."

Menurut Tafsir Ibnu Katsir Allah SWT berfirman bahwa sesungguhnya, dalam kisah para Rasul dan kaum mereka serta bagaimana Allah SWT telah menyelamatkan orang-orang yang beriman dan menghancurkan orang-orang yang kafir : "Terdapat pengajaran bagi orang-orang yang mempunyai akal. Al Quran itu bukanlah cerita yang dibuat-buat." Maksudnya, Al-Qur'an tidak seharusnya didustakan dan dibuat-buat dari selain Allah.

Ayat di atas menunjukkan bahwa Allah SWT memerintahkan untuk mempelajari kisah Nabi dan Rasul dan umat sebelumnya yang terdapat dalam Al-Qur'an karena di dalamnya terdapat pelajaran yang sangat berharga yang dapat diterapkan dalam kehiduapn sehari-hari. Adapun manfaat mempelajari kisah atau sejarah tokoh-tokoh muslim pada masa lampau adalah:

1. Berpartisipasi memelihara peninggalan-peninggalan masa lalu dengan cara mempelajari dan mengambil manfaat dari peninggalan-peninggalan tersebut.
2. Meneladani perilaku yang baik dari tokoh-tokoh terdahulu.
3. Mengambil pelajaran dari berbagai keberhasilan dan kegagalan masa lalu.
4. Merasa bangga dan mencintai kebudayaan Islam yang merupakan buah karya kaum Muslim masa lalu.

BAB V

PENUTUP

5. Kesimpulan

Berdasarkan hasil dari implementasi dan pengujian yang dilakukan peneliti, maka dapat ditarik kesimpulan sebagai berikut :

- Penggunaan Algoritma *Bee Colony* diterapkan untuk mengetahui jarak terpendek NPC menuju ke *player*, yang bertujuan menghalangi *player* dalam menyelesaikan misi yaitu mengambil *item-item* yang terdapat pada *game*. Algoritm *bee colony* berhasil diterapkan ke NPC dengan hasil ke akuratan *agent* 1 dan 2 sebesar 97%, serta *agent* 3 sebesar 98%. Pengujian ke akuratan diperoleh dari data masing-masing *agent* yang dapat dilihat pada tabel 4.5 sampai tabel 4.7.

5.2 Saran

Dalam pembuatan *game* ini peneliti sadar bahwa masih banyak kekurangan yang nantinya sangat perlu untuk dilakukan pengembangan dan sumbangsih terhadap ilmu pengetahuan, di antaranya :

1. Menambah jumlah *level* permainan sehingga permainan menjadi lebih menantang dan menarik.

2. Menambah ragam NPC dan perilaku yang bervariasi seperti tidak bertabarakan antara NPC yang satu dengan NPC lainnya, menghindari hambatan (*Obstacles Avoidance*) dan lain sebagainya.
3. Permainan ini tidak hanya disajikan dalam *platform desktop* saja, namun juga bisa dikembangkan pada *platform smartphone* agar pemahaman terkait ilmuwan-ilmuwan muslim di minati.

DAFTAR PUSTAKA

Abqary, Ridwan. 2010. *101 Info Tentang Ilmuwan Muslim : Menambah Pengetahuan Seputar Ilmuwan Muslim*. Bandung: DAR! Mizan.

Aditya, Derisna. 2014. *Game, Bermanfaat Atau Merugikan?*. https://www.kompasiana.com/derisnaaditya/54f91737a33311f4018b4678/game-bermanfaat-atau-merugikan.html. (di akses pada 22 Februari 2017).

Akbar, Cholis. 2016. *Bapak "Robotika Dunia", Al-Jazari*. https://www.hidayatullah.com/spesial/islamic-discovery/read/2016/02/13/89279/bapak-robotika-dunia-al-jazari.html. (di akses 15 Agustus 2018).

Alam, Muhammad Zidny. 2013. *Optimasi Rute Kendaraan Dengan Kapasitas Mengunakan Modifikasi Algoritma Artificial Bee Colony*. Skripsi. Jakarta : Program Studi Matematika, Universitas Islam Negeri Syarif Hidayatullah.

Al Indunisi, Syaifuddin. 2013. *Ensiklopedia Anak Muslim (Edisi Istimewa) – Temuan yang Mengubah Dunia*. Jakarta: Elex Media Komputindo.

Annisa, Izzah. 2018. *12 Ilmuwan Muslim yang Terkenal di Dunia*. Yogyakarta: Bentang Pustaka.

Creighton, Ryan Henson. 2010. *Unity 3D Game Development by Example*. Birmingham: Packt Publishing Ltd.

Djamaludin Santoso, Heru. 2016. *Pergerakan NPC Menggunakan Algoritma Boids dan Artificial Bee Colony Pada Simulasi Mengelilingi Ka'Bah (Thawaf)*. Skripsi. Malang: Teknik Informatika, Universitas Islam Negeri Maulana Malik Ibrahim.

Franklin, Stan. 1997. *Cybernetics and System*. Jurnal : *Autonomous Agentts as Embodied* AI. 28 : 6 (1997) 499-520.

Goldstone, Will. (2009). Unity Game Development Essentials. Prackt Publishing.

Henry, Samuel. 2010. *Cerdas Dengan Game*. Jakarta: PT. Gramedia Pustaka Utama.

JimHyuk, Hong. 2005. *Evolving Reactive NPCs for the Real-Time Simulation Game*. Seoul : Yonsei University.

Ibnu Katsir, Al-Imam Abu Fida Isma'il. 2004. *Terjemahan Tafsir Ibn Katsir Jus 12*. Jakarta: Sinar Baru Al-Gesindo.

Ingham, James. 1997. *What is an Agentt?*. Centre for Software Maintenance Universityof Durham.

Iqbal Hanafri, Muhammad. 2015. *Game Edukasi Tebak Gambar Bahasa Jawa Menggunakan Adobe Flash CS6 Berbasis Android.* Jurnal : SISFOTEK GLOBAL. ISSN : 2088-1762 Vol. 5 No. 2.

Iskandar, Salman. 2007. *99 Tokoh Muslim Dunia for Kids*. Bandung. DAR! Mizan.

Tafsir Ibnu Katsir. 2004. *Terjemahan Tafsir Ibn Katsir Juz 1*. Halaman 36. Bogor: Pustaka Imam Syafi'i.

Jatiningsih, Wilujeng, dkk. 2014. *Autonomous Agentt Based NPC Swarm Attack Behavior Using Bee Colony Algorithm*. Proceedings of the Seminar on Intelligent Technology and Its Applications (ISITIA), May 22th.

Juniar Ilham, Ahmad. 2015. *Impelementasi Ant Colony Optimization Untuk Pencarian Rute Terpendek Karakter Tank Pada Game Battle Joen.* Skripsi. Malang : Teknik Informatika, Universitas Islam Negeri Maulana Malik Ibrahim.

Karaboga, D. *Artificial Bee Colony Algorithm. Scholarpedia 2010,5,6915*. Availableonline: http://www.scholarpedia.org/article/Artificial_bee_colony_algorithm/. (di akses tanggal 01 April 2017).

Masood, Ehsan. 2009. *Science & Islam (Ilmuwan-Ilmuwan Muslim Pelopor Hebat)*. Jakarta. Gramedia.

Miftachul Arif, Yunifa, dkk. 2012. *Pergantian Senjata NPC pada Game FPS Menggunakan Fuzzy Sugeno.* Vol 1, No 2 (2012).

Nawawi, Imam. 2015. *Gaya Hidup Muslim*. https://www.hidayatullah.com/kajian/gaya-hidup-muslim/read/2015/01/27/37574/tiga-langkah-menumbuhkan-kecintaan-anak-baca-al-quran.html. (di akses pada 22 Februari 2017).

Nugroho, Rakhmad Fajar dan Ayub, Mewati. 2013. *Penerapan Algoritma Artificial Bee Colony dalam Aplikasi Penjadwalan Pelajaran untuk Sekolah Menengah Pertama*. Jurnal Informatika, Vol. 9 No. 1, Juni 2013: 1 – 17.

Robby Nugroho, Priyadiva, dkk. *Pengaturan Perilaku Pasukan Non Player Character Menggunakan Metode Flocking Behavior Berbasis Agentt pada Permainan Real Time Strategy*. Jurnak Teknik Elektro. Di unduh dari https://anzdoc.com/pengaturan-perilaku-pasukan-non-player-character-menggunakan.html

Schell, J. (2008). The Art of Game Design : A Book of Lense. USA: Morgan Kaufmann Publisher.

Swastika, Vannesa Mayrahmah. 2015. *Perkembangan Teknologi Di Indonesia*. http://www.kompasiana.com/vanessams/perkembangan-teknologi-di-indonesia_55547634b67e615e14ba545b. (di akses pada 22 Februari 2017).

Fuad Abidin, Taufik. 2013. *Accuracy Measure, Precision, Recall & F-Measure*. https://www.youtube.com/watch?v=dTCMRtf5YGY (di akses pada 14 Mei 2018).

Zulfadli. 2010. *Penggunaan Education Game Untuk Meningkatkan Hasil Belajar IPA Biologi Konsep Klasifikasi Makhluk Hidup*. Semarang : Jurnal Pendidikan IPA Indonesia.