SIMULASI PERGERAKAN AGEN PENGUNJUNG PADA WISATA PULAU PARI MENGGUNAKAN ALGORITMA *ARTIFICIAL BEE COLONY*

SKRIPSI

oleh :
MOH. HUSNUS SYAWAB
NIM. 16650055

JURUSAN TEKNIK INFORMATIKA
FAKULTAS SAINS DAN TEKNOLOGI
UNIVERSITAS ISLAM NEGERI MAULANA MALIK IBRAHIM
MALANG
2020

SIMULASI PERGERAKAN AGEN PENGUNJUNG PADA WISATA PULAU PARI MENGGUNAKAN ALGORITMA *ARTIFICIAL BEE COLONY*

SKRIPSI

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

oleh :
MOH. HUSNUS SYAWAB
NIM. 16650055

JURUSAN TEKNIK INFORMATIKA
FAKULTAS SAINS DAN TEKNOLOGI
UNIVERSITAS ISLAM NEGERI MAULANA MALIK IBRAHIM MALANG
2020

LEMBAR PERSETUJUAN

SIMULASI PERGERAKAN AGEN PENGUNJUNG PADA WISATA PULAU PARI MENGGUNAKAN ALGORITMA *ARTIFICIAL BEE COLONY*

SKRIPSI

MOH. HUSNUS SYAWAB
NIM. 16650055

Telah Diperiksa dan Disetujui untuk Diuji
Tanggal 24 Desember 2020:

Dosen Pembimbing I	Dosen Pembimbing II
Hani Nurhayati, M. T	Yunifa Miftachul Arif, M. T
NIP. 19780625 200801 2 006	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

LEMBAR PENGESAHAN

SIMULASI PERGERAKAN AGEN PENGUNJUNG PADA WISATA PULAU PARI MENGGUNAKAN ALGORITMA *ARTIFICIAL BEE COLONY*

SKRIPSI

Oleh:

MOH. HUSNUS SYAWAB
NIM. 16650055

Telah Dipertahankan Di depan Dewan Penguji
dan Dinyatakan Diterima Sebagai Salah Satu Persyaratan
untuk Memperoleh Gelar Sarjana Komputer (S.Kom)
Pada Tanggal 24 Desember 2020

Susunan Penguji			**Tandatangan**
1. Penguji Utama	:	Fatchurrohman, M.Kom NIP. 19700731 200501 1 002	( )
2. Ketua Penguji	:	Fresy Nugroho, M. T NIP. 19710722 201101 1 001	( )
3. Sekretaris Penguji	:	Hani Nurhayati, M.T NIP. 19780625 200801 2 006	( )
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

PERNYATAAN KEASLIAN TULISAN

Saya yang bertanda tangan di bawah ini:

Nama : Moh. Husnus Syawab

NIM : 16650055

Fakultas/jurusan : Sains dan Teknologi/Teknik Informatika

Judul Skripsi : Simulasi Pergerakan Agen Pengunjung Pada Wisata Pulau Pari Menggunakan Algoritma *Artificial Bee Colony*

Menyatakan dengan sebenarnya bahwa skripsi yang saya tulis ini benar-benar merupakan hasil karya sendiri, bukan merupakan pengambilalihan 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 di kemudian hari terbukti atau dapat dibuktikan skripsi ini hasil jiplakan, maka saya bersedia menerima sanksi atas perbuatan tersebut.

Malang, 24 Desember 2020

Yang membuat pernyataan,

METERAI TEMPEL 78861AHF076235915 6000 ENAM RIBURUPIAH

Moh. Husnus Syawab

NIM. 16650055

MOTTO

“Teruslah berbuat kebaikan.”

HALAMAN PERSEMBAHAN

اَلْحَمْدُ لِلّٰهِ رَبِّ الْعَالَمِينَ

Puji syukur kepada Allah SWT penulis persembahkan hasil karya ini kepada:

Ayah, almarhum Dimjati, yang telah berjasa mendidik dan membimbing hingga saya bisa sampai pada titik pencapaian sekarang ini.

Ibu dan kakak, Umi Sulistiany, M. Aruman Hasmi, yang selalu memanjatkan do'a dan memberikan motivasi untuk segera menyelesaikan skripsi ini.

Bapak Dr. Cahyo Crysdian, selaku Ketua jurusan Teknik Informatika, Fakultas Sains dan Teknologi UIN Maliki Malang.

Dosen pembimbing, Ibu Hani Nurhayati, M. T dan Bapak Yunifa Miftachul Arif, M. T, yang telah memberikan bimbingan dan saran sehingga penelitian ini bisa berjalan dengan lancar.

Dosen penguji, Bapak Fatchurrohman, M.Kom & Bapak Fresy Nugroho, M. T, yang tidak banyak memberikan revisi yang sulit

Seluruh dosen Teknik Informatika Universitas Islam Negeri Maulana Malik Ibrahim Malang, yang telah dengan ikhlas membagi ilmu dan berbagai pengalamannya.

Mbak Citra Fidya Atmalia, S.H, yang telah membantu dalam urusan administrasi.

Support system yang telah banyak membantu dalam menyelesaikan skripsi ini.

Sahabat-sahabat saya yang begitu banyak dan tidak bisa disebutkan namanya satu per satu, saudara Andromeda Teknik Informatika 2016, keluarga besar Teknik Informatika UIN Maulana Malik Ibrahim Malang, yang tiada hentinya memberikan semangat untuk menyelesaikan skripsi ini.

KATA PENGANTAR

بِسْمِ اللهِ الرَّحْمنِ الرَّحِيمِ

Puji syukur penulis panjatkan kehadirat Tuhan yang Maha Esa yang telah memberikan berkat dan rahmat-Nya sehingga penulis dapat menyelesaikan skripsi yang berjudul "Simulasi Pergerakan Agen Pengunjung Pada Wisata Pulau Pari Menggunakan Algoritma *Artificial Bee Colony*". Tujuan dari penyusunan skripsi ini guna memenuhi salah satu syarat untuk bisa menempuh ujian sarjana computer pada Fakultas Sains dan Teknologi Program Studi Teknik Informatika di Universitas Islam Negeri Maulana Malik Ibrahim Malang. Dalam penyusunan skripsi penulis banyak mendapatkan bantuan ataupun masukan dari berbagai pihak. Oleh sebab itu, penulis mengungkapkan rasa terima kasih kepada:

1. Bapak Prof. Dr. Abdul Haris, M.Ag., selaku Rektor Universitas Islam Negeri (UIN) Maulana Malik Ibrahim Malang.
2. Ibu Dr. Sri Harini, M.Si, selaku Dekan Fakultas Sains dan Teknologi.
3. Bapak Dr. Cahyo Crysdian, selaku Ketua jurusan Teknik Informatika, Fakultas Sains dan Teknologi UIN Maliki Malang.
4. Ibu Hani Nurhayati, M. T, selaku Wali Dosen yang telah memberikan saran dan masukan hingga saya bisa menyelesaikan perkuliahan dengan baik.
5. Ibu Hani Nurhayati, M. T, selaku pembimbing I, yang telah memberikan bimbingan dan masukan sehingga penulis dapat menyelesaikan penelitian ini.
6. Bapak Yunifa Miftachul Arif, M. T, selaku pembimbing II, yang telah memberikan bimbingan dan masukan sehingga penulis dapat menyelesaikan penelitian ini.
7. Mbak Citra Fidya Atmalia, S.H dan para staff laboran, yang telah membantu dalam urusan administrasi.
8. Orang tua dan keluarga penulis yang telah memanjatkan do'a dan memberikan suntikan motivasi.
9. Sahabat-sahabat tercinta, saudara Andromeda TI 16, kakak tingkat, dan adik tingkat Teknik Informatika UIN Maulana Malik Ibrahim Malang.
10. Semua pihak yang terlibat dalam penyusunan skripsi.

Penulis menyadari bahwa skripsi ini masih terdapat banyak kekurangan, namun penulis berharap skripsi ini dapat bermanfaat khususnya bagi penulis secara pribadi.

Malang, 24 Desember 2020

Penulis

DAFTAR ISI

DAFTAR GAMBAR

DAFTAR TABEL

ABSTRAK

Syawab, Moh. Husnus. 2020. ***Simulasi Pergerakan Agen Pengunjung Pada Wisata Pulau Pari Menggunakan Algoritma Artificial Bee Colony.*** Skripsi. Jurusan Teknik Informatika Fakultas Sains Dan Teknologi Universitas Islam Negeri Maulana Malik Ibrahim Malang. Pembimbing : (I) Hani Nurhayati, M. T. (II) Yunifa Miftachul Arif, M. T.

Kata Kunci : Simulasi, Pulau Pari, *Artificial Bee Colony*.

Pulau pari merupakan salah satu tempat wisata yang terletak di Indonesia, tepatnya di Kepulauan Seribu, Jakarta. Terdapat berbagai macam wisata pada pulau tersebut diantaranya yaitu pantai bintang, pantai kresek, dan pantai pasir perawan. Menurut Kepala Suku Dinas Pariwisata dan Kebudayaan Kepulaaun seribu, Cucu Ahmad Kurnia mengungkapkan jumlah wisatawan meningkat hingga 18% dibandingkan tahun lalu. Karena banyaknya pengunjung di pulau pari, maka dibutuhkan suatu simulasi untuk merencanakan skenario marketing dalam pemasaran kedepannya. Penelitian ini membuat simulasi *game*, khususnya *crowd simulation*. Dalam menunjang *crowd simulation* yang akan dibuat, maka dibutuhkan sebuah AI yang digunakan untuk mengatur *Non-Playable Character* (NPC). Algoritma yang akan digunakan yaitu *Artificial Bee Colony*(ABC) karena dinilai algoritma ABC dapat menangani pergerakan pengunjung yang kompleks. Pengujian yang dilakukan dengan menghitung jumlah pengunjung dan estimasi waktu yang dibutuhkan untuk menyelesaikan semua wisata. Uji coba dilakukan dengan 3 NPC dan 4 *target* yang akan dikunjungi. NPC akan menuju *player* atau *tour guide* terlebih dahulu, kemudian NPC dan *player* akan menuju *target*, dalam hal ini tempat wisata dan titik *finish*. Hasil pengujian dalam segi jarak menunjukkan bahwa dalam semua percobaan, semua NPC berhasil dalam mencapai 5 tujuan yang ditentukan, Dalam segi waktu, hasilnya dari 12 percobaan, pada percobaan keempat, kedelapan dan kedua belas, dengan masing-masing berbeda jumlah NPC menunjukkan uji coba tercepat. Dari hasl tersebut didapatkan rute pada percbaan keempat,kedelapan dan kedua belas merupakan rute yang paling efisien dalam berwisata mengelilingi pulau pari.

ABSTRACT

Syawab, Moh. Husnus. 2020. ***Simulation of the Movement of Visitor Agents on Pari Island Tourism Using Artificial Bee Colony Algorithms.*** Essay. Department of Informatics Engineering, Faculty of Science and Technology, Islamic State University of Maulana Malik Ibrahim of Malang. Supervisor: (I) Hani Nurhayati, M. T. (II) Yunifa Miftachul Arif, M. T.

Keyword : Simulation, Pari Island, Artificial Bee Colony.

Pari Island is one of the tourist attractions located in Indonesia, to be precise in the Thousand Islands, Jakarta. There are various kinds of tours on the island including star beach, crackle beach, and virgin sand beach. According to the Head of the Thousand Islands Tourism and Culture Office, Cucu Ahmad Kurnia said the number of tourists increased by 18% compared to last year. Due to the large number of visitors on Pari Island, a simulation is needed to plan marketing scenarios for future marketing. This study makes game simulations, especially crowd simulation. To support the crowd simulation that will be created, an AI is needed to control the Non-Playable Character (NPC). The algorithm that will be used is Artificial Bee Colony (ABC) because it is assessed that the ABC algorithm can handle complex visitor movements. Tests are carried out by calculating the number of visitors and the estimated time it will take to complete all tours. The trial was conducted with 3 NPCs and 4 targets to be visited. The NPC will go to the player or tour guide first, then the NPC and player will go to the target, in this case the tourist spot and the finish point. The test results in terms of distance show that in all experiments, all NPCs succeeded in achieving the 5 specified objectives.In terms of time, the results were from 12 trials, in the fourth, eighth and twelfth trials, with each different number of NPCs showing the fastest trial . From the results, it was found that the routes in the fourth, eighth and twelfth experiments were the most efficient routes for traveling around Pari Island.

الملخص

الثواب, محمد حسن .٢٠٢٠. محاكاة لحركة وكلاء الزوار في سياحة جزيرة باري بخوارزمية "مستعمرة النحل الاصطناعية". أطروحة. قسم علوم و الطقنية, كلية العلوم و الهندسة، الجامعة الأسلامية الحكومية مولانا مالك إبراهيم مالانج .المشرف:) ١(هاني نور حياتي الماجستير ٢() يونيفا مفتاح العارف الماجستير.

الكلمات الرئيسية: محاكاة ، جزيرة باري ، مستعمرة نحل اصطناعية.

جزيرة باري هي واحدة من مناطق الجذب السياحي الموجودة في إندونيسيا ، على وجه الدقة في جزر الألف ، جاكرتا. هناك أنواع مختلفة من الجولات في الجزيرة بما في ذلك شاطئ النجوم وشاطئ كراكل وشاطئ الرمال البكر. وفقًا لرئيس مكتب السياحة والثقافة في جزر الألف ، قال كوكو أحمد كورنيا إن عدد السياح ارتفع بنسبة 18٪ مقارنة بالعام الماضي. نظرًا للعدد الكبير من الزوار في جزيرة باري ، يلزم إجراء محاكاة لتخطيط سيناريوهات التسويق للتسويق في المستقبل. تقوم هذه الدراسة بمحاكاة الألعاب ، وخاصة محاكاة الجماهير. لدعم محاكاة الجماهير التي سيتم إنشاؤها ، هناك حاجة إلى ذكاء اصطناعي للتحكم في). الخوارزمية التي سيتم استخدامها هي NPCالشخصية غير القابلة للعب (يمكنها ABC) لأنه تم تقييم أن خوارزمية ABCمستعمرة النحل الاصطناعية (التعامل مع حركات الزائر المعقدة. يتم إجراء الاختبارات عن طريق حساب عدد الزوار والوقت المقدر لاستكمال جميع الجولات. أجريت التجربة مع 3 إلى NPCشخصيات غير قابلة للعب و 4 أهداف يجب زيارتها. سيذهب واللاعب إلى الهدف ، في NPCاللاعب أو المرشد السياحي أولاً ، ثم سيذهب هذه الحالة المكان السياحي ونقطة النهاية. تظهر نتائج الاختبار من حيث المسافة أنه في جميع التجارب ، نجحت جميع الشخصيات غير القابلة للعب في تحقيق

الأهداف الخمسة المحددة. . من النتائج ، وجد أن الطرق في التجارب الرابعة والثامنة والثانية عشرة كانت أكثر الطرق كفاءة للسفر حول جزيرة باري.

BAB I

PENDAHULUAN

1.1 Latar Belakang

Pulau Pari adalah salah satu tempat wisata yang terletak di Indonesia, tepatnya di Kepulauan Seribu, Jakarta. Pulau ini terletak di tengah gugusan pulau yang berderet dari arah selatan sampai utara perairan di Jakarta. Dalam Pulau Pari, terdapat berbagai macam wisata, diantaranya yaitu pantai Bintang, pantai Kresek, dan pantai Pasir Perawan.

Menurut Kepala Dinas Pariwisata dan Kebudayaan Kepulaaun seribu, Cucu Ahmad Kurnia mengungkapkan jumlah wisatawan meningkat hingga 18% dibandingkan tahun lalu. Jumlah pengunjung dari tanggal 1 – 8 Juni 2019 tercatat mencapai 57.700 orang dimana 56.981 orang merupakan wisatawan nusantara dan 719 orang merupakan wisata mancanegara.

Dengan banyaknya pengunjung di Pulau Pari, utamanya pada waktu liburan, maka dibutuhkanlah suatu gambaran atau simulasi untuk merencanakan skenario *marketing* untuk pemasaran kedepannya. simulasi yang digunakan dapat digunakan dengan berbagai cara, salah satunya yaitu *game*. *Game* menjadi *trending* dalam menjalankan sebuah simulasi dinilai lebih atraktif dan lebih menarik dalam hal *user interface*.

Sesuai dalam Al-Qur'an surat Al-Isra' ayat 7, yang berbunyi:

...إِنْ أَحْسَنْتُمْ أَحْسَنْتُمْ لِأَنْفُسِكُمْ وَإِنْ أَسَأْتُمْ فَلَهَا

Artinya:

"Jika kamu berbuat baik (berarti) kamu berbuat baik bagi dirimu sendiri dan jika kamu berbuat jahat maka kejahatan itu bagi dirimu sendiri." (QS. Al-Isra'/ 17 : 7)

Maksud dari ayat tersebut yakni kita sebagai umat manusia seharusnya saling berbuat baik *(hablum minannas*). Apabila kita mempermudah pekerjaan orang lain, maka segala bentuk pekerjaan kita akan dimudahkan oleh Allah *Subhanahu Wa Ta'ala*. Dalam hal ini, peneliti memberikan kemudahan kepada petugas Pulau Pari dalam mengatur wisatawan. Selain itu, wisatawan pun juga ikut terbantu sewaktu mengunjungi wisata-wisata yang ada di Pulau Pari.

Oleh sebab itu, maka penulis memiliki ide untuk membuat simulasi *game* untuk Pulau Pari, khususnya *crowd simulation*. Dalam menunjang *crowd simulation* yang akan dibuat, maka dibutuhkan sebuah *Artificial Intelligence*(AI) yang digunakan untuk mengatur *Non-Playable Character* (NPC). Algoritma yang akan digunakan yaitu *Artificial Bee Colony* (ABC) karena dinilai algoritma ABC dapat menangani pergerakan pengunjung yang kompleks.

1.2 Pernyataan Masalah

Berdasarkan latar belakang diatas, penelitian ini membahas bagaimana mengatur pergerakan *Non-Playable Character* (NPC) pengunjung wisata Pulau Pari menggunakan algoritma *Artificial Bee Colony* (ABC) dalam mengatasi banyaknya pengunjung yang ada dan waktu yang dibutuhkan dalam menyelesaikan semua wisata.

1.3 Tujuan Penelitian

Tujuan Penelitian ini adalah untuk mengetahui pergerakan *Non-Playable Character* (NPC) pengunjung wisata Pulau Pari menggunakan algoritma *Artificial Bee Colony* (ABC) dalam mengatasi banyaknya pengunjung yang ada dan waktu yang dibutuhkan dalam menyelesaikan semua wisata.

1.4 Manfaat Penelitian

Penelitian ini menghasilkan sebuah pembuatan simulasi dalam bentuk *game* untuk mengetahui pergerakan NPC agen pengunjung menggunakan algoritma *Artificial Bee Colony* dalam mengatasi banyaknya pengunjung dan waktu menyelesaikan wisata.

1.5 Batasan Masalah

Batasan masalah pada penelitian ini adalah sebagai berikut :

1. Data yang digunakan merupakan data pengunjung wisata Pulau Pari.
2. Simulasi yang akan dibangun menggunakan *platform unity desktop*.

1.6 Sistematika Penulisan

Penelitian ini tersusun dalam laporan yang terbagi dalam beberapa bab sebagai berikut :

BAB I PENDAHULUAN : Berisi tentang latar belakang, rumusan masalah, tujuan penelitian, manfaat penelitian, dan batasan masalah.

BAB II STUDI PUSTAKA : Berisi tentang *literature review* dan teori tentang simulasi beserta metode yang akan digunakan dalam penelitian.

BAB III DESAIN DAN IMPLEMENTASI: Berisi tentang desain pembangunan simulasi beserta implementasi algoritma *Artificial Bee Colony*.

BAB IV UJI COBA DAN PEMBAHASAN : Berisi tentang hasil penelitian yang telah dilakukan.

BAB V PENUTUP : Berisi tentang kesimpulan dan saran dari penelitian ini sebagai masukan untuk memperbaiki kekurangan dari penelitian dan dapat dikembangkan kembali pada penelitian selanjutnya.

BAB II
STUDI PUSTAKA

2.1 Simulasi

Simulasi menurut Simun & Emshoff (1970) merupakan model sistem yang komponennya diuraikan dengan proses aritmatika dan logika untuk memperkirakan sifat-sifat dinamis sistem yang dijalankan oleh komputer. Menurut Udin Syaefudin Sa'ud (2005), simulasi adalah sebuah model yang terdapat satu unit variabel dengan menampilkan karakteristik utama dari sistem yang sebenarnya.

Menurut Pusat Bahasa Depdiknas (2005), simulasi adalah metode pelatihan yang memperagakan sesuatu dalam bentuk artifisial yang mirip dengan keadaan yang sesungguhnya.

Simulasi yaitu proses suatu aplikasi dalam membangung model dari sistem nyata atau usulan, melakukan percobaan dengan model tersebut untuk mempelajari kinerja sistem, menunjukkan perilaku sistem, atau untuk membangun sistem yang baru sesuai dengan kinerja yang diharapkan (Khosnevis, 1994).

2.2 *Artificial Bee Colony*(ABC)

Algoritma *Artificial Bee Colony* (ABC) merupakan algoritma dalam mengoptimalkan suatu permasalahan pada *meta-heuristic* (Karaboga, 2005). ABC ditemukan oleh David Karaboga yang menemukan ide dari perilaku cerdas koloni lebah madu ketika mencari makanan. Pada koloni lebah madu, terdapat 3 lebah dalam pembagian tugas diantaranya yaitu *Employed Bee* (lebah pekerja), *Onlooker Bee* (lebah pengintai), *Scout Bee* (lebah pengamat).

Menurut Karaboga, tahapan algoritma ABC dalam menyelesaikan permasalahan optimasi yakni sebagai berikut :

Initialization Phase

REPEAT

Employed Bees Phase

Onlooker Bees Phase

Scout Bees Phase

Memorize The Best Solution achieved as far

UNTIL (*Cycle = Maximum Cycle Number or a Maximum CPU time*)

2.2.1 Scouting Phase

Dalam mencari sumber makanan, untuk mencari dan menemukan sumber makanan, koloni lebah akan mengirimkan *scout bee* (lebah pengamat) secara acak dari posisi awal ke posisi selanjutnya. Secara matematis, pola yang dilakukan *scout bee* dapat dijabarkan seperti pada persamaan (1).

$$X'_i = X_i + rand[-1\ 1] \quad (1)$$

Keterangan :

X'_i = Posisi lebah selanjutnya dalam sumbu X

X_i = Posisi lebah saat ini dalam sumbu X

Pada persamaan (1), persamaan tersebut bisa dikembangkan apabila pencarian dibatasi ruang lingkup dengan *radius* (R), sehingga akan menjadi seperti persamaan (2).

$$X'_i = X_i + rand[-1\ 1]XR \quad (2)$$

Ketika *scout bee* melakukan pencarian sumber makanan, tiap lebah bisa meminimalkan tabrakan dengan objek lain ataupun dengan lebah sendiri. Jika sumber makanan yang akan dituju terhalang oleh adanya *obstacle*, maka *scout bee* yang melakukan pencarian harus mencari posisi yang baru.

Secara matematis, pencarian posisi baru yaitu dengan melakukan perhitungan ulang dari posisi awal dan lebah atau agen dikatakan bertabrakan apabila jarak posisi agen dengan halangan lebih kecil dari jumlah *radius* lebah (r_{bee}) dan *radius* halangan (r_{obs}) objek tersebut. Oleh karena itu, jarak minimal lebah atau agen pada halangan atau lebah lainnya (d_{ij}) harus sama atau lebih besar dari jumlah kedua *radius* objek tersebut ($r_{bee} + r_{obs}$).

$$d_{ij} - (r_{bee} + r_{obs}) \quad (3)$$

$$f_{collision} = d_{ij} - (r_{bee} + r_{obs}) > 0 \quad (4)$$

Untuk mencari manakah yang lebih rendah, persamaan (4) harus dibandingkan dengan nilai 0 sehingga menghasilkan nilai C.

$$C = \min_0 (d_{ij} - (r_{bee} + r_{obs}) \quad (5)$$

Persamaan (5) tersebut menjadi sebuah persyaratan dasar dalam pemilihan posisi bagi persaman (2). Apabila nilai C kurang dari 0, maka lebah atau agen akan mengulang proses pencarian posisi baru.

2.2.2 Onlooker Phase

Lebah Pengintai (*onlooker bee*) yang sudah mengetahui posisi sumber makanan baru dari *scout bee*, kembali ke sarang kemudian menginformasikan data lokasi sumber makanan baru kepada lebah pengamat di sarang. Digunakan persamaan *euclidian* untuk mengukur jarak lebah ke *target*.

$$d^2 = (X_j - X_i)^2 + (y_j - y_i)^2$$
$$d = \sqrt{(X_j - X_i)^2 + (y_j - y_i)^2} \quad (6)$$

2.2.3 Employed Phase

Saat lebah pengintai telah menginformasikan data yang didapatkan kepada lebah pengamat, lebah tersebut akan menganalisis data dan mengolah data tersebut kemudian diteruskan kepada lebah pekerja sebagai tumpuan untuk mengambil makanan yang terdapat pada *target*. Tiap lebah pekerja kembali ke sarang dan membawa makanan, maka lebah pekerja akan memberikan informasi ke lebah pengamat bahwasanya makanan yang tersajikan pada *target* telah berkurang.

2.3 *Tools* Pembuat *Game*

2.3.1 Blender

Blender merupakan aplikasi yang digunakan untuk membuat perancangan suatu multimedia khususnya 3 dimensi, diantaranya seperti *assets* sebuah *game*,

animasi 3D, *visual effect* dan model cetak 3D. Blender merupakan *software open source*, sehingga aplikasi ini dapat digunakan untuk beberapa *Operation System* di antaranya Mac, Windows, dan Linux (Ardhianto, et.al, 2012).

2.3.2 Unity 3D

Unity 3D merupakan suatu aplikasi untuk membuat *game*, simulasi, dan bangunan arsitektur. Dalam unity 3D, terdapat banyak fitur yang menunjang pembuatan suatu *game* menjadi menarik, diantaranya seperti bahasa pemrograman yang dapat digunakan antara lain Boo, javascript, dan C#. Selain itu unity 3D adalah game *multi platform*, sehingga dapat digunakan untuk PC, Mac, Android, *browser*, dan lainnya. Fitur lainnya yang terdapat pada unity 3D seperti *sky box, rendering, lighting, sound effect, physic game.* (Sudarmilah, E., dkk, 2013).

2.3.3 Bahasa Pemrograman C#

Bahasa C# adalah bahasa yang dapat digunakan di aplikasi Unity 3D yang biasanya digunakan untuk *scripting* dalam pembuatan suatu *game*. C# adalah sebuah bahasa pemrograman berorientasi objek yang mempunyai akses secara khusus untuk *platform* .NET sebagai bahasa yang memungkinkan *programmer* untuk bermigrasi ke .NET (Deitel, 2010).

Dalam Kutipan Yulius Eka Agung Saputra (2014), Bahasa C# adalah bahasa pemrograman *object oriented* dan memilki *class library* yang sangat lengkap yang berisi *prebuilt component*, sehingga memudahkan *programmer* untuk *develop program* lebih cepat. Menurut Erico Darmawan (2014), C# tidak hanya dapat digunakan pada operasi Windows, namun aplikasi C# dapat digunakan dalam

berbagai macam Sistem Operasi menggunakan .NET *Framework* dan Mono *Framework*.

C# (*C Sharp*) merupakan bahasa pemrograman yang baru diciptakan oleh Microsoft yang dikembangkan dibawah kepemimpinan Anders Hejlsberg. Bahasa C# telah distandarisasi secara internasional oleh ECMA. Seperti bahasa pemrograman lainnya, C# bisa digunakan dalam pembuatan berbagai macam jenis aplikasi, baik berbentuk windows, web, atau android.

2.4 Penelitian Terkait

Penelitian mengenai algoritma ABC telah dilakukan sebelumnya yaitu oleh Heru Santoso Djamaluddin (2016) dengan judul Pergerakan NPC Menggunakan Algoritma Boids Dan Artificial Bee Colony Pada Simulasi Mengelilingi Ka'bah (Thawaf). Penelitian tersebut menjelaskan tentang pembuatan simulasi mengelilingi Ka'bah (Thawaf) dengan algoritma *Boids* dan *Artificial Bee Colony*. Hasil penelitian ini yaitu dengan menerapkan metode *Boids* dan *Artificial Bee Colony*, NPC jama'ah yang mengelilingi ka'bah memiliki perilaku yang fleksibel sesuai dengan keadaan pada lingkungan,

Penelitian selanjutnya berjudul Implementasi Algoritma Artificial Bee Colony Untuk Membangkitkan Perilaku NPC Pada Game Survival Horror "Left Alone" Sebagai Media Pengenalan Rumah Cut Nyak Dhien oleh Innamul hasan pada tahun 2017 yang membahas suatu perancangan perilaku pencarian NPC pada *game* dan memberikan penjelasan pembangunan *game* edukasi bergenre *survival*

horror. Algoritma *Aritifical Bee Colony* digunakan sebagai pembangkit perilaku pencarian NPC. Hasil dari penelitiannya yaitu algoritma tersebut dapat dijalankan sebagai pembangkit perilaku pencarian pada NPC seperti dalam pencarian jalur terpendek ke lokasi *player*, selain itu juga berhasil menempuh jarak maksimal dengan nilai jarak akhir dari NPC ke *player*.

Penelitian berikutnya berbentuk tesis oleh Wilujeng jatiningsih pada tahun 2015 berjudul Pergerakan Serangan Kelompok NPC Berbasis Agen Otonom Menggunakan Algoritma Artificial Bee Colony tentang pembuatan simulasi pergerakan kelompok NPC yang dapat mendeteksi datangnya musuh dan melakukan penyerangan secara berkelompok dan *random* dari yang berbeda dengan algoritma ABC, karena memiliki *self-organization* yang baik juga mempunyai pembagian tugas yang detail. Hasil dari penelitian ini berhasil menunjukkan bahwa dengan ABC dapat membuat simulasi pergerakan sekelompok agen yang dapat mengetahui posisi *target* kemudian mendekati dalam rute atau formasi acak dengan tujuan berkerumun di sekitar *target*.

Penelitian selanjutnya dilakukan oleh Dinita Rahmalia dan Teguh Herlambang berjudul Optimasi Masalah Transportasi Distribusi Semen Menggunakan Algoritma Artificial Bee Colony untuk mengatasi masalah pendistribusian semen dari pabrik ke gudang tujuan dengan jumlah muatan maksimum yang dapat diangkut supaya biaya transportasi minimum dengan metode *heuristic* yaitu algoritma *Artificial Bee Colony*. Dari hasil yang diperoleh dengan algoritma ABC, diperoleh pendekatan solusi yang optimum dalam hal ini

meminimumkan biaya distribusi dengan memenuhi kendala jumlah persediaan dan jumlah permintaan unit semen.

Penelitian A Study On Bee Algorithm and A* Algorithm for Pathfinding in Games dilakukan oleh Aimi Najwa Sabri pada tahun 2018 dimana berfokus membandingkan algoritma tradisional, algoritma A* (*A Star*) dengan pencarian optimisasi algoritma *Bee* dalam melakukan *pathfinding* yang paling optimal pada sebuah *game.**s***

Penelitian yang berjudul Improved Artificial Bee Colony Algorithm for Vehicle Routing Problem With Time Window dilakukan oleh Baozhen Yao, Qianqian Yan, Mengjie Zhang dan Yuong yang pada tahun 2017 untuk menyelidiki masalah kombinatorial yang kompleks yang dikenal sebagai *Vehicle Routing Problem with Time Windows* (VRPTW) menggunakan *Improved Artificial Bee Colony* (IABC).Hasilnya menunjukkan bahwa IABC tingkat keefektifannya dapat digunakan sebagai tolak ukur dalam memecahkan VRPTW.

BAB III

DESAIN DAN IMPLEMENTASI

3.1 Desain Sistem

Gambar 3.1*Flowchart* desain sistem penelitian

Dalam penelitian ini, NPC yaitu wisatawan dan *tour guide* akan *spawn* (muncul) di titik poin yang sudah ditentukan, kemudian NPC akan mencari *target* untuk dituju. Kemudian NPC menuju *target* menggunakan algoritma *Artificial Bee*

Colony (ABC), sehingga NPC akan sampai di *target*. Kemudian NPC akan mengelilingi Pulau Pari.

.

3.2 Desain *Interface* Game

Game ini akan mensimulasikan tentang aktifitas-aktifitas yang akan dilakukan wisatawan saat berwisata di Pulau Pari. Player akan menggerakan karakter utama yaitu seorang *tour guide*, *tour guide* ini ditugaskan untuk mengarahkan NPC-NPC dalam hal ini yaitu rombongan wisatawan dalam berwisata di Pulau Pari.

Gambar 3.2 Tampilan menu utama

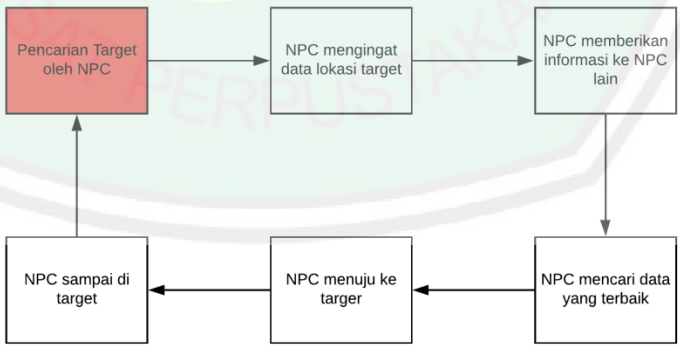
Gambar 3.2 merupakan tampilan menu utama (*main menu*) dalam menu utama. Pada menu tersebut terdapat 3 tombol di antaranya tombol mulai yang digunakan untuk mulai masuk *game*, tombol tentang untuk menjelaskan *game*, dan tombol keluar untuk menutup dan keluar dari *game*.

Gambar 3.3 Tampilan *gameplay*

Gambar 3.3 adalah tampilan *gameplay* saat sudah bermain setelah menekan tombol mulai.

3.3 Perancangan Algortima ABC *(Artificail Bee Colony)*

Dalam perancangan algoritma *Artificial Bee Colony*, Hal ini ditujukan dan di implementasikan pada NPC dalam hal ini yaitu pengunjung. NPC akan mencari jalan yang tepat untuk menuju ke tujuan yaitu *target* (*tour guide*), sehingga NPC dapat menemukan *target* dan mengikuti *target* kemanapun bergerak.

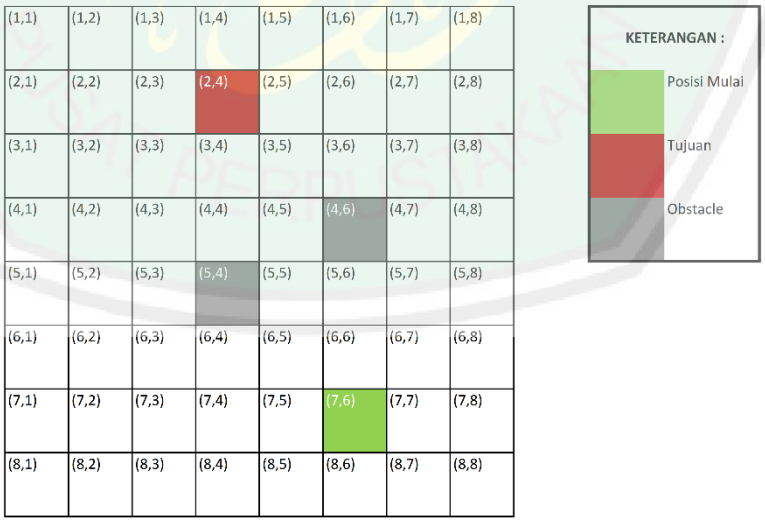


Gambar 3.4 Diagram kerja algoritma ABC

Pada awalnya, NPC akan mencari *target* yang dituju serta mengingat data lokasi *target*, Hal ini merupakan *scouting phase* dan NPC yang mencari *target* dan mengingat data dalam algoritma *Artificial Bee Colony* (ABC) merupakan *Scouting Bee* atau lebah pengamat.

Setelah mendapatkan data *target*, NPC tersebut memberikan informasi ke NPC lain atau NPC B. NPC B ini merupakan *onlooker bee* atau lebah pengintai, NPC B ini akan memberikan info kepada NPC C, hal ini disebut *onlooker phase*. NPC C akan yang telah mempunyai info dari NPC menghitung jarak menuju *target* dari data yang ada dan mengabarkan NPC lainnya untuk menuju *target*. NPC C merupakan *employeed bee* dan proses yang dilakukan NPC C disebut *employeed phase*. Jika sudah menuju *target*, *scouting bee* akan mencari *target* baru dan berulang terus sampai *target* terakhir yang dituju.

3.4 Simulasi Algoritma *Artificial Bee Colony*



Gambar 3.5 Kondisi awal simulasi ABC

Dalam rancangan simulasi ini, tujuannya yaitu mencari jalan dari "Posisi Mulai" dimana NPC muncul menuju ke "tujuan", dimana merupakan posisi tujuan/*Target* dan diantara posisi mulai dan tujuan akan ada *obstacle* atau halangan. Berikut ini merupakan tahap-tahap perhitungan mencari jalur:

1. Menentukan Posisi Mulai NPC, Tujuan, dan *Obstacle.*
2. Perkiraan posisi NPC selanjutnya setelah dari posisi mulai.
3. Pengecekan halangan atau *obstacle* pada posisi NPC selanjutnya, Jika Nilai C kurang dari 0, maka posisi tabrakan dengan *obstacle*, jika lebih dari sama dengan 0, maka tidak bertabrakan
4. Mencari nilai optimal dengan menghitung posisi baru NPC ke *target* dengan posisi mulai NPC ke *Target*. Apabila posisi baru NPC lebih kecil dari posisi mulai, maka NPC beralih ke posisi besar. Apabila lebih besar, maka dilakukan perhitungan ulang posisi dari posisi mulai. Jika tujuan (d) tidak bernilai 0, maka pencarian dilanjutkan lagi seperti langkah 1, jika bernilai 0, maka iterasi berhenti

Iterasi 1 :

- Langkah 1 :

  Posisi mulai NPC = (7,6)

  Tujuan = (2,4)

  Obstacle = (4,6) , (5,4)

- Langkah 2 :

$X_i = 7, X_j = 6$

$X'_i = X_i + rand[-1\ 1]XR$

X'_i = 7 + (-1) * 2

= 5

$X'_j = X_j + rand[-1\ 1]XR$

X'_j = 6 + 0 * 2

= 6

X'_{ij} = (5,6) (Posisi Baru)

- Langkah 3 :

Untuk *obstacle* (5,4)

$X_{ij} = (5,6), y_{ij} = (5,4), r_{bee} = 0{,}5, r_{obs} = 0{,}5$

$d_{ij} = \sqrt{(X_j - X_i)^2 + (y_j - y_i)^2}$

$d_{ij} = \sqrt{(6-5)^2 + (4-5)^2}$

= 1,414214

$C = \min_0 (d_{ij} - (r_{bee} + r_{obs})$

$C = \min_0 (1{,}414214 - (0{,}5 + 0{,}5)$

$C = \min_0 (0{,}414214)$

Untuk *obstacle* (4,6)

$X_{ij} = (5,6), y_{ij} = (4,6), r_{bee} = 0{,}5, r_{obs} = 0{,}5$

$d_{ij} = \sqrt{(X_j - X_i)^2 + (y_j - y_i)^2}$

$d_{ij} = \sqrt{(6-5)^2 + (4-6)^2}$

= 2,236068

$C = \min_0 (d_{ij} - (r_{bee} + r_{obs})$

$C = \min_0 (2{,}236068 - (0{,}5 + 0{,}5)$

$C = \min_0 (1{,}236068)$

Posisi baru tidak bertabrakan dengan *obstacle* karena nilai C tidak ada yang kurang dari 0.

- Langkah 4 :

Posisi Mulai NPC ke *target*

$X_{ij} = (7,6), y_{ij} = (2,4)$

d $= \sqrt{(y_j - x_j)^2 + (y_i - x_i)^2}$

d $= \sqrt{(2-7)^2 + (4-6)^2}$

d $= \sqrt{25 + 4}$

d = 5,385165

Posisi baru NPC ke *target*

$X_{ij} = (5,6), y_{ij} = (2,4)$

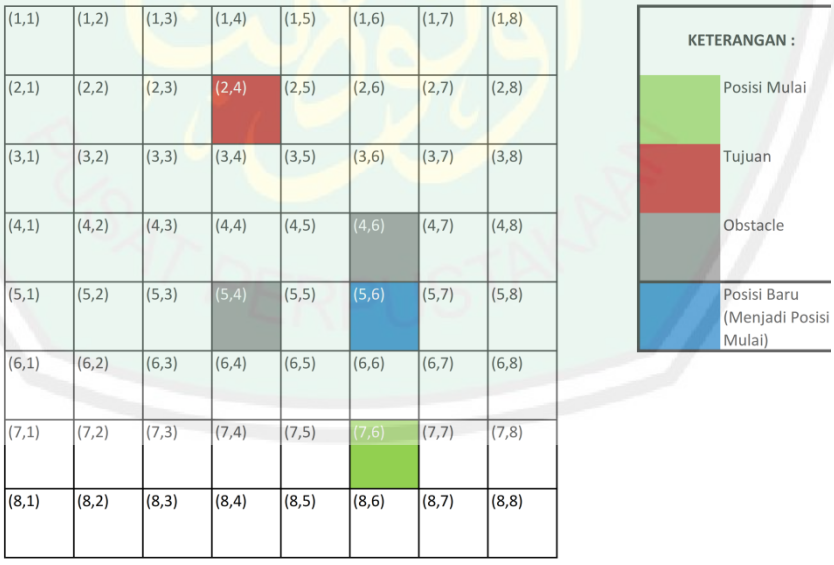
d $= \sqrt{(y_j - x_j)^2 + (y_i - x_i)^2}$

d $= \sqrt{(2 - 5)^2 + (4 - 6)^2}$

d $= \sqrt{9 + 4}$

d $= 3{,}605551$

Posisi baru dijadikan posisi mulai sekarang karena nilai dari posisi baru lebih kecil dari posisi mulai, dan karena d tidak sama dengan 0, maka pencarian/iterasi dilanjutkan.



Gambar 3.6 Hasil iterasi 1 simulasi ABC

Iterasi 2 :

- Langkah 1 :

  Posisi mulai NPC = (5,6)

  Tujuan = (2,4)

  Obstacle = (4,6) , (5,4)

- Langkah 2 :

  $X_i = 5, X_j = 6$

  $X'_i = X_i + rand[-1\ 1]XR$

  X'_i = 5 + (-1) * 2

  = 3

  $X'_j = X_j + rand[-1\ 1]XR$

  X'_j = 6 + (-1) * 2

  = 4

  X'_{ij} = (3,4) (Posisi Baru)

- Langkah 3 :

  Untuk *obstacle* (5,4)

  $X_{ij} = (3,4), y_{ij} = (5,4), r_{bee} = 0{,}5, r_{obs} = 0{,}5$

  $d_{ij} = \sqrt{(X_j - X_i)^2 + (y_j - y_i)^2}$

$$d_{ij} = \sqrt{(4-3)^2 + (4-5)^2}$$

$$= 1{,}414214$$

$$C = \min_{0} (d_{ij} - (r_{bee} + r_{obs})$$

$$C = \min_{0} (1{,}414214 - (0{,}5 + 0{,}5)$$

$$C = \min_{0} (0{,}414214)$$

Untuk *obstacle* (4,6)

$$X_{ij} = (3{,}4), y_{ij} = (4{,}6), r_{bee} = 0{,}5, r_{obs} = 0{,}5$$

$$d_{ij} = \sqrt{(X_j - X_i)^2 + (y_j - y_i)^2}$$

$$d_{ij} = \sqrt{(4-3)^2 + (4-6)^2}$$

$$= 2{,}236068$$

$$C = \min_{0} (d_{ij} - (r_{bee} + r_{obs})$$

$$C = \min_{0} (2{,}236068 - (0{,}5 + 0{,}5)$$

$$C = \min_{0} (1{,}236068)$$

Posisi baru tidak bertabrakan dengan *obstacle* karena nilai C tidak ada yang kurang dari 0.

- Langkah 4 :

  Posisi Mulai NPC ke *target*

$X_{ij} = (5,6), y_{ij} = (2,4)$

d $= \sqrt{(y_j - x_j)^2 + (y_i - x_i)^2}$

d $= \sqrt{(2 - 5)^2 + (4 - 6)^2}$

d $= \sqrt{9 + 4}$

d = 3,605551

Posisi baru NPC ke *target*

$X_{ij} = (3,4), y_{ij} = (2,4)$

d $= \sqrt{(y_j - x_j)^2 + (y_i - x_i)^2}$

d $= \sqrt{(2 - 3)^2 + (4 - 4)^2}$

d $= \sqrt{1 + 0}$

d = 1

Posisi baru dijadikan posisi mulai sekarang karena nilai dari posisi baru lebih kecil dari posisi mulai, dan karena d tidak sama dengan 0, maka pencarian/iterasi dilanjutkan.

Gambar 3.7 Hasil iterasi 2 algoritma ABC

Iterasi 3 :

- Langkah 1 :

  Posisi mulai NPC = (3,4)

  Tujuan = (2,4)

  Obstacle = (4,6) , (5,4)

- Langkah 2 :

  $X_i = 3, X_j = 4$

  $X'_i = X_i + rand[-1\ 1]XR$

  X'_i = 3 + (-0,5) * 2

  = 2

$X'_j = X_j + rand[-1\ 1]XR$

X'_j = 4 + (0) * 2

= 4

X'_{ij} = (2,4) (Posisi Baru)

- Langkah 3 :

Untuk *obstacle* (5,4)

$X_{ij} = (2,4), y_{ij} = (5,4), r_{bee} = 0{,}5, r_{obs} = 0{,}5$

$d_{ij} = \sqrt{(X_j - X_i)^2 + (y_j - y_i)^2}$

$d_{ij} = \sqrt{(4-2)^2 + (4-5)^2}$

= 2,236068

$C = \min_0 (d_{ij} - (r_{bee} + r_{obs})$

$C = \min_0 (2{,}236068 - (0{,}5 + 0{,}5)$

$C = \min_0 (1{,}236068)$

Untuk *obstacle* (4,6)

$X_{ij} = (2,4), y_{ij} = (4,6), r_{bee} = 0{,}5, r_{obs} = 0{,}5$

$$d_{ij} = \sqrt{(X_j - X_i)^2 + (y_j - y_i)^2}$$

$$d_{ij} = \sqrt{(4-2)^2 + (4-6)^2}$$

$$= 2{,}828427$$

$$C = \min_{0} (d_{ij} - (r_{bee} + r_{obs})$$

$$C = \min_{0} (2{,}828427 - (0{,}5 + 0{,}5)$$

$$C = \min_{0} (1{,}828427)$$

Posisi baru tidak bertabrakan dengan *obstacle* karena nilai C tidak ada yang kurang dari 0.

- Langkah 4 :

  Posisi Mulai NPC ke *target*

$$X_{ij} = (3,4), y_{ij} = (2,4)$$

$$d = \sqrt{(y_j - x_j)^2 + (y_i - x_i)^2}$$

$$d = \sqrt{(2-3)^2 + (4-4)^2}$$

$$d = \sqrt{1+0}$$

$$d = 1$$

Posisi baru NPC ke *target*

$X_{ij} = (2,4), y_{ij} = (2,4)$

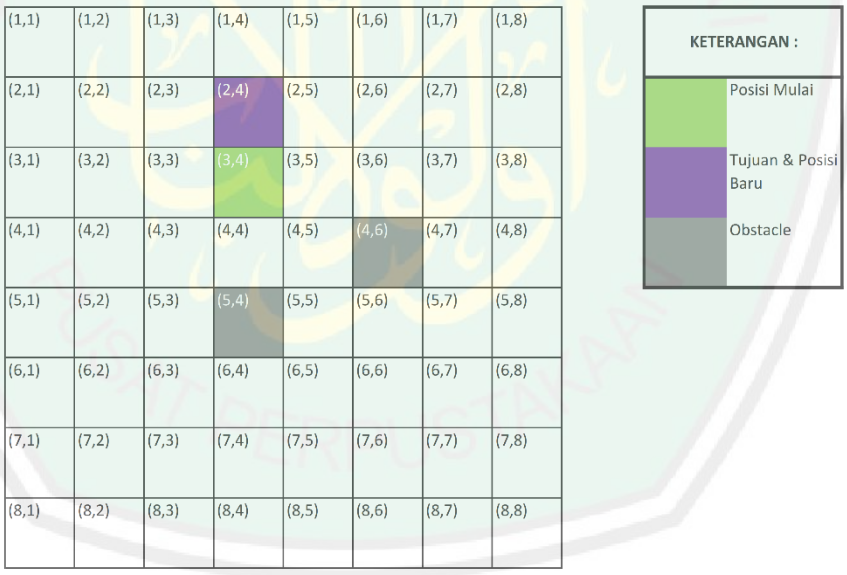
d $= \sqrt{(y_j - x_j)^2 + (y_i - x_i)^2}$

d $= \sqrt{(2 - 2)^2 + (4 - 4)^2}$

d $= \sqrt{0 + 0}$

d $= 0$

Posisi baru dijadikan posisi mulai sekarang karena nilai dari posisi baru lebih kecil dari posisi mulai, dan karena d sama dengan 0, maka pencarian/iterasi berhenti.



Gambar 3.8 Hasil iterasi 3 simulasi ABC

3.5 Rencana Pengujian

Dalam menentukan pengujian yang akan dilakukan memiliki hasil yang baik, beberapa faktor terkait akan digunakan untuk melihat efektifnya metode yang diterapkan pada penelitian ini.

3.5.1 Pengujian Jumlah Pengunjung Yang Tetap Mengikuti *Tour Guide*

Proses pengujian ini dilakukan dengan cara menambahkan jumlah pengunjung saat melakukan wisata dengan *tour guide* setiap bermain *game*. Dalam pengujian ini, jumlah pengunjung akan dihitung untuk mengetahui pengunjung yang mengikuti *tour guide* dalam menyelesaikan wisata akan berhasil semuanya atau ada beberapa yang mungkin menghilang dari pengunjung lain yang mengikuti *tour guide* dalam menyelesaikan wisata yang ada di Pulau Pari.

3.5.2 Pengujian Waktu Yang Dibutuhkan Dalam Menyelesaikan Wisata

Proses pengujian ini dilakukan untuk mengetahui estimasi waktu yang dibutuhkan untuk menyelesaikan semua wisata yang ada di Pulau Pari. Dalam hal ini, waktu akan berhenti ketika pengunjung yang terdepan telah mencapai garis akhir dari wisata yang dilakukan. Semakin cepat waktu yang dibutuhkan dalam menyelesaikan semua wisata, maka semakin berhasil pengujian tersebut.

BAB IV
IMPLEMENTASI DAN PEMBAHASAN

4.1 Implementasi

Tahap implementasi merupakan tahap dimana aplikasi sudah dirancang dan siap untuk diterapkan.. Selain itu dilakukan pengujian pada *game* tersebut apakah game yang sudah dibangun dapat berjalan dengan baik sesuai tujuan.

4.4.1 Kebutuhan Perangkat Keras

Hardware yang digunakan dalam mengimplementasikan aplikasi *game* simulasi:

Tabel 4. 1 Spesifikasi perangkat keras

Perangkat Keras	Spesifikasi
RAM	4096 MB
Processor	Intel Core i5-4200U CPU @ 1.60GHz (4 CPUs), ~2.3GHz
VGA	NVIDIA GeForce GT 720M
Speaker	*On*
Mouse & Keyboard	*On*
Storage	256 GB (SSD), 500 GB (HDD)
Monitor	14.0” HD LED LCD

4.4.2 Kebutuhan Perangkat Lunak

Berikut ini adalah *software* yang digunakan dalam mengimplementasikan aplikasi *game* simulasi.

Tabel 4. 2 Kebutuhan *software*

Perangkat Lunak	Spesifikasi
Desain 3D	Blender
Game Engine	Unity 2018.4.27
Sistem Operasi	Windows 10 Pro
Texture	Blender
Script Editor	Microsoft Visual Studio

4.2 Pengujian Algoritma *Artificial Bee Colony*

Pengujian dilakukan untuk mengetahui proses *Tour Guide/Player* dan Wisatawan/NPC ke *target*/wisata oleh *Artificial Bee Colony* dengan menggunakan 3-7 NPC. NPC akan di *spawn* secara random pada salah satu area. Setiap NPC mempunyai kecepatan dengan nilai 2 dimana digunakan untuk menuju ke *target*. Bahasa pemrograman yang digunakan yaitu C#. Di bawah ini merupakan *source code* dari algoritma *Artificial Bee Colony*.

Tabel 4. 3 Source Code Algoritma Artificial Bee Colony

No	Method dan Fungsi	Keterangan

1	void start ()
	{
	aBegin = transform.position.a;
	bBegin = transform.position.b;
	cBegin = transform.position.c;
	
	jarak = Vector3.Distance(toPlayer.transform.position, transform.position);
	}
		Method untuk menentukan lokasi awal NPC

2	void Update()
	{
	jarak = Vector3.Distance(toPlayer.transform.position, transform.position);
		Method untuk menjalankan ABC dan juga mengupdate perubahan posisi dan rotasi pada NPC

3	if (jarak >= 1.5) {	Pemberian Kondisi Jarak NPC dapat mendekati *player*

4	speed = 100;
	rotationSpeed = 15;
	x*Target* = toPlayer.transform.position.x;
	z*Target* = toPlayer.transform.position.z;
		Pemberian nilai *speed*, *rotation speed* dan lokasi *player*

5	float random_a = Random.Range(-1.0f, 1.0f);
	aNew = aBegin + random_a * 2;
	
	float random_c = Random.Range(-1.0f, 1.0f);
	cNew = cBegin + random_c * 2;
		Merandom nilai a dan c.

6	Vector3 newPosition = new Vector3(xNew, yBegin, zNew);	Menghitung posisi baru NPC yang memungkinkan

7	transform.position = Vctr3.MoveTowards(transform.position, newPosition, Time.deltaTime * speed);
	
	var rot = Quaternion.LookRot (newPosition - transform.position);
	
	transform.rot = Quaternion.Slerp(transform.rot, rot, Time.deltaTime * rotSpeed);
		Memerintah dan mengubah posisi serta rotasi NPC untuk menuju posisi baru

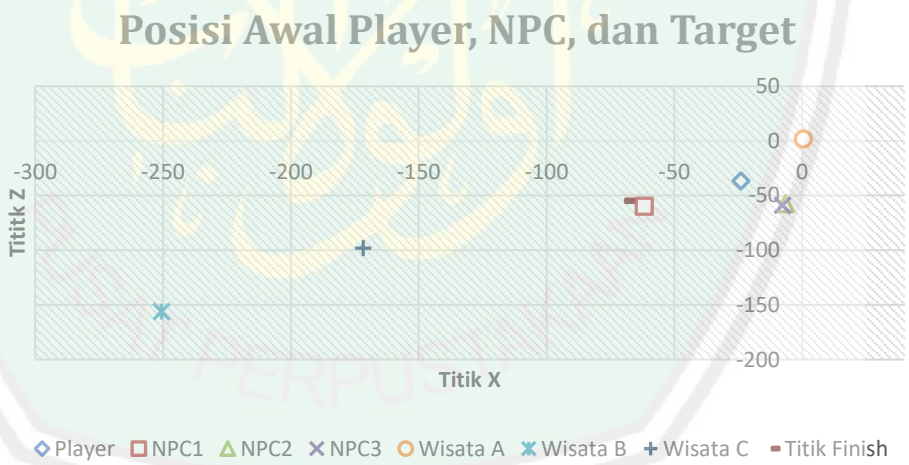
8	X1 = Mathf.Sqrt(((a*Trgt* - aBegin) * (aTrgt - xBegin)) + ((z*Target* - zBegin) * (z*Target* - zBegin)));
	
	X2 = Mathf.Sqrt(((a*Trgt* - aNew) * (a*Trgt* - aNew)) + ((c*Trgt* - cNew) * (c*Trgt* - cNew)));
	
	if (X2 <= X1)
	{
	aBegin = aNew;
	cBegin = cNew;
	X1 = X2;
	}
		Menghitung dan membandingkan jarak posisi NPC lama dan *target* dengan jarak posisi baru NPC dan *target*

9	} else if (jarak <= 1)
	{
	speed = 0;
	rotationSpeed = 0;
	}
	}
		Pemberian kondisi jarak berhenti *player* dan NPC

4.3 Uji Coba

Uji coba yang difokuskan pada subbab ini merupakan uji coba sistem untuk melihat proses yang dihasilkan dari penggunaan *Artificial Bee Colony* dalam mencapai beberapa *target*. Percobaan dilakukan dengan 3 NPC dan 4 *target* yang akan dikunjungi. NPC akan menuju *player* atau *tour guide* terlebih dahulu, kemudian NPC dan *player* akan menuju *target*, dalam hal ini tempat wisata dan titik *finish*.

Untuk *Player* dan *target* yang akan dituju titik koordinatnya tidak di *spawn* secara acak. Koordinat awal *player*, NPC, dan *target*/tempat wisata dalam titik x dan titik z adalah sebagai berikut :



Gambar 4.1 Posisi awal pemain, NPC & *target* sebelum dilakukan uji coba

Posisi pemain :

X = -23.88 , Z = -36.73

Jarak Berhenti dengan NPC = 1.0

Posisi NPC :

X1 = -6.170293, Z1 = -59.87256 (NPC 1)

X2 = -6.551942, Z2 = -57.40668 (NPC 2)

X2 = -7.618847, Z2 = -59.0334 (NPC 3)

Posisi *target* :

XA = 0.48, ZA = 1.68 (Tempat Wisata A)

XB = 0.48, ZB = 1.68 (Tempat Wisata B)

XC = 0.48, ZC = 1.68 (Tempat Wisata C)

XD = 0.48, ZD = 1.68 (Titik *Finish*)

Player berada di titik bentuk belah ketupat (-23.88, -36.73) yang menjadi titik tujuan pertama semua NPC yang ada. Kemudian setelah menuju *player*, *player* dan NPC menuju *target* selanjutnya yaitu tempat Wisata dan titik *Finish*. Algoritma dijalankan pertama kali Ketika NPC ter *spawn* secara acak di *area* yang telah ditentukan. Setiap pergerakan player dan NPC akan dicatat seperti perubahan titik posisi x(x), perubahan titik posisi z(z), dan perubahan jarak(j). Untuk titik posisi y tidak dicatat karena titik posisi y tidak berubah dari awal sampai akhir. Di bawah ini merupakan tabel hasil dari uji coba yang telah dilakukan.

Tabel 4. 4 Hasil iterasi pengujian sistem

Iterasi	*Target*	NPC		
		NPC 1	NPC 2	NPC 3
	(X, Z)	(X1, Z1, J1)	(X2, Z2, J2)	(X3, Z3, J3)
1	(-23.88, -36.73)	(-6.170293, -59.87256, 29.16553)	(-6.551942, -57.40668, 27.00375)	(-7.618847, -59.0334, 27.62757)
2	(-23.88, -36.73)	(-8.126993, -58.14452, 29.17)	(-6.41872, -58.16549, 27.00858)	(-7.714159, -60.53197, 27.63229)
3	(-23.88, -36.73)	(-7.805409, -56.31142, 27.2124)	(-7.447562, -56.9614, 27.67767)	(-8.582841, -58.8637, 28.80183)
4	(-23.88, -36.73)	(-8.147549, -57.12054, 25.53002)	(-8.492817, -55.01067, 26.12916)	(-9.638983, -57.82529, 26.93661)
5	(-23.88, -36.73)	(-5.930689, -55.44033, 25.78684)	(-7.27775, -55.63441, 23.86851)	(-7.707031, -58.75755, 25.48519)
6	(-23.88, -36.73)	(-6.958751, -57.42403, 25.96019)	(-10.32006, -56.85443, 25.19302)	(-9.2146, -57.21812, 27.3579)
7	(-23.88, -36.73)	(-5.841584, -54.95981, 26.76282)	(-10.1203, -53.78247, 24.50387)	(-10.95037, -58.76932, 25.22923)
8	(-23.88, -36.73)	(-7.990431, -55.25322, 26.0553)	(-9.647654, -52.01464, 22.95498)	(-7.279227, -55.75243, 25.46789)
9	(-23.88, -36.73)	(-8.025515, -54.53262, 24.61804)	(-8.33744, -53.6825, 21.74969)	(-7.85935, -56.66812, 25.31164)
10	(-23.88, -36.73)	(-9.798588, -52.56079, 23.87415)	(-9.462574, -52.17021, 23.03553)	(-9.870073, -57.84605, 25.6099)
11	(-23.88, -36.73)	(-9.332769, -54.17958, 22.07841)	(-9.634883, -51.51804, 21.25018)	(-9.617795, -56.02051, 25.40924)

Iterasi	*Target*	NPC		
		NPC 1	NPC 2	NPC 3
	(X, Z)	(X1, Z1, J1)	(X2, Z2, J2)	(X3, Z3, J3)
12	(-23.88, -36.73)	(-10.94538, -53.92466, 22.75492)	(-8.492099, -49.65349, 20.57393)	(-7.946458, -55.04799, 24.02522)
13	(-23.88, -36.73)	(-8.905804, -54.28231, 21.55545)	(-7.819396, -49.71951, 20.14208)	(-8.800321, -54.48846, 24.33207)
14	(-23.88, -36.73)	(-8.669211, -52.87684, 22.78934)	(-7.856501, -48.77883, 20.69655)	(-9.175957, -52.70739, 23.34303)
15	(-23.88, -36.73)	(-11.68003, -53.80312, 22.2534)	(-9.653606, -48.76269, 20.08989)	(-10.65485, -54.62291, 21.78654)
16	(-23.88, -36.73)	(-13.63757, -52.52871, 21.24968)	(-10.76637, -48.21356, 18.6776)	(-10.80836, -53.78576, 22.28762)
17	(-23.88, -36.73)	(-13.40877, -51.44041, 19.83777)	(-10.66521, -46.8888, 17.47903)	(-10.39907, -55.36594, 21.52774)
18	(-23.88, -36.73)	(-15.00952, -51.20757, 16.66051)	(-10.75438, -48.72761, 16.59173)	(-12.33076, -51.95103, 22.00496)
19	(-23.88, -36.73)	(-12.44747, -48.35009, 15.99139)	(-11.61298, -48.04575, 17.53672)	(-10.86411, -52.5522, 21.00188)
20	(-23.88, -36.73)	(-12.38253, -50.90767, 16.20928)	(-13.18462, -47.26856, 16.73926)	(-13.46886, -51.17343, 20.52882)
21	(-23.88, -36.73)	(-13.05967, -48.72429, 17.86631)	(-11.23388, -47.21455, 15.07081)	(-15.31448, -52.7131, 18.87131)
22	(-23.88, -36.73)	(-13.86107, -49.15728, 16.76737)	(-13.52822, -45.79722, 14.97124)	(-14.05951, -51.00559, 16.81114)
23	(-23.88, -36.73)	(-16.19643, -47.91317, 16.01541)	(-14.72175, -46.86605, 13.82211)	(-15.64331, -49.70929, 17.3756)

Iterasi	*Target*	NPC		
		NPC 1	NPC 2	NPC 3
	(X, Z)	(X1, Z1, J1)	(X2, Z2, J2)	(X3, Z3, J3)
24	(-23.88, -36.73)	(-17.42064, -47.37834, 14.4929)	(-13.19089, -45.07988, 13.72188)	(-16.41921, -49.08096, 15.7766)
25	(-23.88, -36.73)	(-19.07907, -47.65561, 12.52148)	(-14.52846, -47.79271, 13.62551)	(-15.50971, -50.64464, 14.48747)
26	(-23.88, -36.73)	(-17.64497, -48.21964, 13.41033)	(-14.6688, -45.03173, 13.23544)	(-15.87675, -48.85878, 14.48746)
27	(-23.88, -36.73)	(-17.86606, -47.93777, 13.13638)	(-14.98341, -43.56857, 12.46763)	(-17.64654, -47.68344, 14.5889)
28	(-23.88, -36.73)	(-17.7655, -47.1457, 12.78509)	(-13.70832, -45.03928, 11.29567)	(-19.29072, -46.12404, 13.08651)
29	(-23.88, -36.73)	(-17.49415, -47.62838, 12.52564)	(-16.77667, -44.43284, 11.51104)	(-19.03917, -44.55714, 12.04606)
30	(-23.88, -36.73)	(-20.82557, -46.3406, 12.69309)	(-14.82562, -44.21943, 10.57987)	(-21.02966, -46.13109, 10.53122)
31	(-23.88, -36.73)	(-20.24669, -46.23669, 11.41284)	(-16.61345, -45.52725, 11.6741)	(-20.16569, -43.66019, 9.975332)
32	(-23.88, -36.73)	(-22.10529, -47.73164, 10.25939)	(-18.59846, -42.90811, 11.49194)	(-20.68589, -44.82301, 8.625911)
33	(-23.88, -36.73)	(-19.69358, -46.52892, 10.8794)	(-17.22432, -41.70835, 9.849098)	(-21.22664, -44.19143, 8.796372)
34	(-23.88, -36.73)	(-19.89189, -47.15911, 9.850491)	(-19.23421, -39.72135, 7.600281)	(-18.69209, -41.287, 7.697765)
35	(-23.88, -36.73)	(-17.12822, -45.59412, 10.9332)	(-19.01232, -38.84735, 5.981734)	(-17.95979, -42.06174, 7.042972)

Iterasi	*Target*	NPC		
		NPC 1	NPC 2	NPC 3
	(X, Z)	(X1, Z1, J1)	(X2, Z2, J2)	(X3, Z3, J3)
36	(-23.88, -36.73)	(-19.8913, -46.94979, 11.01085)	(-18.27743, -40.00789, 5.463923)	(-19.38051, -41.41243, 8.071763)
37	(-23.88, -36.73)	(-19.3373, -46.40454, 11.02512)	(-17.79036, -38.58311, 6.618934)	(-21.02895, -41.19315, 6.621751)
38	(-23.88, -36.73)	(-18.08185, -46.61169, 10.76615)	(-19.39922, -39.21566, 6.495745)	(-19.93547, -41.13485, 5.452074)
39	(-23.88, -36.73)	(-20.29663, -44.39334, 11.5301)	(-20.12706, -39.85751, 5.285155)	(-20.98044, -39.7937, 6.053019)
40	(-23.88, -36.73)	(-20.86803, -43.50066, 9.67679)	(-21.04029, -41.00756, 5.053992)	(-20.24986, -40.5162, 4.412557)
41	(-23.88, -36.73)	(-20.91436, -42.20036, 7.604985)	(-19.37305, -39.81063, 5.295135)	(-20.30878, -40.05183, 5.402798)
42	(-23.88, -36.73)	(-19.43472, -41.32391, 6.355852)	(-20.87999, -39.03318, 5.508068)	(-22.06175, -41.10591, 5.046296)
43	(-23.88, -36.73)	(-22.78324, -43.26165, 6.522388)	(-21.30083, -37.17257, 3.997704)	(-21.19219, -40.48619, 4.886775)
44	(-23.88, -36.73)	(-19.7728, -40.68871, 6.337333)	(-19.32684, -36.51868, 3.014981)	(-19.74262, -41.01878, 4.796903)
45	(-23.88, -36.73)	(-21.01535, -37.69893, 5.37475)	(-23.48608, -37.81593, 2.528798)	(-19.77554, -40.42272, 4.797971)
46	(-23.88, -36.73)	(-20.27582, -37.09023, 3.843828)	(-25.29359, -36.39986, 1.877226)	(-19.34774, -40.11882, 4.523781)
47	(-23.88, -36.73)	(-21.07234, -38.61767, 2.125519)	(-25.29359, -36.39986, 1.877226)	(-22.12665, -38.66661, 3.949573)

Iterasi	*Target*	NPC		
		NPC 1	NPC 2	NPC 3
	(X, Z)	(X1, Z1, J1)	(X2, Z2, J2)	(X3, Z3, J3)
48	(-23.88, -36.73)	(-24.15826, -38.05906, 3.407823)	(-25.29359, -36.39986, 1.877226)	(-22.05433, -37.42511, 2.915759)
49	(-23.88, -36.73)	(-23.74346, -36.15648, 2.226688)	(-25.29359, -36.39986, 1.877226)	(-22.39431, -37.76629, 2.065665)
50	(0.48, 1.68)	(-24.47674, -36.2439, 43.34119)	(-25.63225, -36.09037, 43.91934)	(-23.15784, -39.3616, 43.79675)
51	(0.48, 1.68)	(-24.59856, -35.35445, 43.37863)	(-24.97608, -35.21342, 43.96489)	(-23.67316, -38.49022, 43.8546)
52	(0.48, 1.68)	(-23.45777, -34.29142, 43.06503)	(-26.27498, -37.09498, 43.20634)	(-24.91618, -38.40653, 45.20136)
53	(0.48, 1.68)	(-22.60945, -33.8451, 41.56219)	(-23.25968, -37.07588, 42.85927)	(-23.41381, -34.63946, 44.1133)
54	(0.48, 1.68)	(-22.83998, -34.75328, 40.71762)	(-23.35701, -33.5601, 41.96051)	(-25.82412, -34.70418, 43.78219)
55	(0.48, 1.68)	(-22.96278, -32.97991, 40.90327)	(-22.76976, -32.73962, 41.60171)	(-22.75712, -37.31699, 43.47666)
56	(0.48, 1.68)	(-21.03976, -30.06638, 38.27058)	(-23.50294, -33.82093, 39.4435)	(-22.23138, -32.56237, 40.93244)
57	(0.48, 1.68)	(-20.47132, -31.8698, 36.71384)	(-20.5829, -29.66096, 37.89468)	(-22.58458, -34.42315, 40.15017)
58	(0.48, 1.68)	(-19.94207, -28.64456, 35.9912)	(-19.11977, -29.82252, 37.32401)	(-22.09736, -30.61611, 39.13183)
59	(0.48, 1.68)	(-18.46192, -27.00078, 34.92054)	(-21.09699, -27.5932, 35.81976)	(-21.33114, -31.35148, 39.94939)

Iterasi	*Target*	NPC		
		NPC 1	NPC 2	NPC 3
	(X, Z)	(X1, Z1, J1)	(X2, Z2, J2)	(X3, Z3, J3)
60	(0.48, 1.68)	(-17.77943, -25.72291, 33.09167)	(-18.46192, -27.00078, 34.92054)	(-19.10142, -31.37124, 36.59404)
61	(0.48, 1.68)	(-19.34526, -24.02417, 31.307)	(-17.77943, -25.72291, 33.09167)	(-21.24677, -30.14328, 36.74597)
62	(0.48, 1.68)	(-17.89536, -25.47316, 30.87822)	(-19.34526, -24.02417, 31.307)	(-21.76347, -29.33897, 34.97973)
63	(0.48, 1.68)	(-19.94141, -21.66705, 29.42233)	(-18.12323, -23.34583, 30.3618)	(-18.45574, -24.08273, 32.33321)
64	(0.48, 1.68)	(-19.98003, -22.51014, 28.53485)	(-16.15349, -22.22011, 29.01643)	(-18.51124, -22.58532, 30.06983)
65	(0.48, 1.68)	(-12.98011, -21.82084, 25.68853)	(-14.55412, -23.22889, 27.64783)	(-18.40639, -22.1728, 29.22344)
66	(0.48, 1.68)	(-16.35969, -18.71794, 24.84228)	(-12.3671, -22.50605, 25.41415)	(-16.86891, -23.47166, 28.83878)
67	(0.48, 1.68)	(-18.90089, -15.37562, 24.88469)	(-13.87354, -21.56086, 25.96337)	(-16.99243, -20.1211, 26.77726)
68	(0.48, 1.68)	(-18.92196, -14.39508, 24.57845)	(-13.98165, -23.03792, 25.66306)	(-16.34376, -21.52542, 26.35997)
69	(0.48, 1.68)	(-15.91945, -16.05545, 23.44462)	(-16.63122, -16.15273, 24.37394)	(-12.77098, -19.10819, 24.67041)
70	(0.48, 1.68)	(-17.42455, -13.81233, 22.90905)	(-13.29245, -20.37713, 23.01017)	(-15.91945, -16.05545, 23.44462)
71	(0.48, 1.68)	(-18.77779, -13.01886, 21.98306)	(-10.85534, -16.4036, 22.51867)	(-19.27867, -13.18285, 22.87669)

Iterasi	*Target*	NPC		
		NPC 1	NPC 2	NPC 3
	(X, Z)	(X1, Z1, J1)	(X2, Z2, J2)	(X3, Z3, J3)
72	(0.48, 1.68)	(-14.08915, -15.82272, 19.62381)	(-12.1697, -14.51983, 20.72991)	(-16.93967, -9.90157, 21.79361)
73	(0.48, 1.68)	(-17.04728, -7.920362, 18.76003)	(-11.44321, -10.93836, 19.14601)	(-17.02223, -8.548565, 20.10938)
74	(0.48, 1.68)	(-11.48465, -9.129717, 17.39556)	(-17.04728, -7.920362, 18.76003)	(-14.49089, -14.24096, 19.83787)
75	(0.48, 1.68)	(-7.833448, -6.967704, 14.11964)	(-13.32106, -7.572749, 17.16806)	(-14.26349, -16.58184, 20.56088)
76	(0.48, 1.68)	(-8.124941, -6.04664, 11.97047)	(-14.54811, -9.069256, 15.98107)	(-14.09495, -13.60436, 19.63398)
77	(0.48, 1.68)	(-9.014708, -4.074312, 10.87409)	(-13.34449, -5.957695, 15.80763)	(-14.49089, -14.24096, 19.83787)
78	(0.48, 1.68)	(-10.29238, -2.770374, 10.13706)	(-11.49152, -4.343527, 16.73137)	(-12.95529, -13.79325, 19.28778)
79	(0.48, 1.68)	(-6.18293, -5.210622, 9.330483)	(-9.683897, -4.097023, 12.11018)	(-10.09807, -11.8635, 18.82995)
80	(0.48, 1.68)	(-2.897197, -4.849725, 6.913572)	(-7.024167, -2.335717, 8.008509)	(-9.934486, -9.281906, 14.55675)
81	(0.48, 1.68)	(-2.24034, -5.85023, 4.403164)	(-2.226435, -2.690856, 6.090873)	(-6.004062, -11.21855, 12.22027)
82	(0.48, 1.68)	(-3.661268, -3.359796, 4.490604)	(-1.537814, -4.165343, 5.093138)	(-2.677095, -10.85167, 10.51242)
83	(0.48, 1.68)	(-1.321384, -2.814284, 3.578407)	(-3.94524, -0.5709743, 5.993234)	(-4.208821, -11.52689, 10.62171)

Iterasi	*Target*	NPC		
		NPC 1	NPC 2	NPC 3
	(X, Z)	(X1, Z1, J1)	(X2, Z2, J2)	(X3, Z3, J3)
84	(0.48, 1.68)	(-1.792254, 1.02176, 1.732773)	(-2.304426, -0.3711522, 4.550655)	(-3.418076, -8.961485, 9.70691)
85	(0.48, 1.68)	(-1.37471, -2.581393, 1.979354)	(-0.8243756, -0.8405877, 3.303512)	(-2.235276, -8.260666, 8.189802)
86	(0.48, 1.68)	(-1.37471, -2.581393, 1.979354)	(2.116832, -3.113486, 2.996699)	(2.693877, -4.636379, 6.357989)
87	(0.48, 1.68)	(-1.37471, -2.581393, 1.979354)	(3.297392, 0.1002746, 2.485503)	(0.8005548, -3.156835, 4.389934)
88	(0.48, 1.68)	(-1.37471, -2.581393, 1.979354)	(2.236996, -1.506292, 1.820525)	(1.194124, 0.07309431, 3.038222)
89	(0.48, 1.68)	(-1.37471, -2.581393, 1.979354)	(2.236996, -1.506292, 1.820525)	(-0.4485383, -1.393082, 2.402779)
90	(0.48, 1.68)	(-1.37471, -2.581393, 1.979354)	(2.236996, -1.506292, 1.820525)	(2.084137, -0.8341649, 1.636526)
91	(-250.59, -156.27)	(-1.180231, -0.5808733, 85.48067)	(-0.640552, -0.8750446, 87.58006)	(0.3606424, -0.08083427, 84.68605)
92	(-250.59, -156.27)	(-4.090386, -2.016849, 84.47878)	(-2.707667, -1.839259, 81.60667)	(-2.257481, -3.248053, 83.71908)
93	(-250.59, -156.27)	(-10.09015, -3.316607, 76.29919)	(-9.059899, -6.054317, 76.17712)	(-9.746513, -4.586303, 76.63969)
94	(-250.59, -156.27)	(-10.83939, -1.281585, 74.49323)	(-11.00313, -10.5115, 71.94949)	(-12.07871, -5.252443, 72.80782)
95	(-250.59, -156.27)	(-13.97554, -0.207817, 72.89445)	(-19.28321, -12.32623, 64.88864)	(-22.49018, -10.28229, 63.79184)

Iterasi	*Target*	NPC		
		NPC 1	NPC 2	NPC 3
	(X, Z)	(X1, Z1, J1)	(X2, Z2, J2)	(X3, Z3, J3)
96	(-250.59, -156.27)	(-27.12504, -11.06192, 57.83164)	(-21.76308, -13.3711, 61.93613)	(-23.90342, -0.8310304, 61.96839)
97	(-250.59, -156.27)	(-202.458, -111.5252, 50.03157)	(-195.5697, -116.6509, 52.42062)	(-205.7084, -112.0163, 49.11032)
98	(-250.59, -156.27)	(-203.093, -126.0243, 40.17559)	(-206.7095, -125.1227, 39.59083)	(-215.7216, -129.5274, 30.38485)
99	(-250.59, -156.27)	(-204.6033, -127.4909, 37.22798)	(-215.8223, -133.821, 24.38449)	(-210.2296, -131.6387, 32.25266)
100	(-250.59, -156.27)	(-232.7847, -141.0344, 21.06488)	(-243.7463, -143.2339, 12.18091)	(-235.9791, -140.9572, 18.78898)
101	(-250.59, -156.27)	(-245.388, -152.5647, 16.29177)	(-249.4586, -165.1608, 5.237226)	(-256.5887, -162.8689, 7.086421)
102	(-250.59, -156.27)	(-250.8504, -157.8047, 9.130751)	(-253.2831, -167.5955, 2.644909)	(-252.9087, -165.3518, 4.659689)
103	(-250.59, -156.27)	(-251.8867, -157.2026, 4.421673)	(-249.5812, -156.9436, 1.683444)	(-254.4418, -154.4404, 3.87796)
104	(-250.59, -156.27)	(-248.6526, -154.9195, 2.052935)	(-249.5812, -156.9436, 1.683444)	(-250.5418, -155.8032, 2.635359)
105	(-171.7, -98.3)	(-248.5577, -156.9732, 28.04259)	(-250.6312, -157.9269, 27.48969)	(-250.2799, -154.8885, 28.58022)
106	(-171.7, -98.3)	(-246.1835, -155.3545, 23.08562)	(-245.4849, -155.7587, 23.82153)	(-248.3487, -155.8717, 26.29145)
107	(-171.7, -98.3)	(-241.9064, -155.9673, 21.09958)	(-243.7062, -153.2723, 22.06342)	(-245.2228, -154.7184, 24.86377)

Iterasi	*Target*	NPC		
		NPC 1	NPC 2	NPC 3
	(X, Z)	(X1, Z1, J1)	(X2, Z2, J2)	(X3, Z3, J3)
108	(-171.7, -98.3)	(-236.5088, -154.9319, 15.75077)	(-241.1328, -150.9669, 19.17442)	(-243.9217, -156.1573, 21.41284)
109	(-171.7, -98.3)	(-234.4421, -153.1271, 15.61147)	(-237.5648, -150.6359, 15.82487)	(-241.4667, -153.9169, 18.92775)
110	(-171.7, -98.3)	(-234.0668, -150.8644, 10.89434)	(-235.5137, -150.7798, 13.83557)	(-236.707, -151.3657, 16.2974)
111	(-171.7, -98.3)	(-231.9804, -148.8689, 9.609456)	(-230.8177, -147.6272, 9.235579)	(-234.281, -149.9782, 11.57407)
112	(-171.7, -98.3)	(-227.0167, -144.4917, 5.184734)	(-230.1499, -148.2992, 6.710022)	(-231.6501, -148.6383, 8.500731)
113	(-171.7, -98.3)	(-226.2366, -143.9405, 3.301403)	(-226.7953, -149.3174, 4.673595)	(-231.2674, -149.7984, 9.200686)
114	(-171.7, -98.3)	(-225.3624, -143.4134, 2.663771)	(-225.9243, -145.2334, 3.969831)	(-230.2405, -149.3195, 8.730116)
115	(-171.7, -98.3)	(-221.638, -146.0488, 1.870506)	(-224.1113, -145.6334, 2.082742)	(-230.805, -147.8913, 6.785544)
116	(-171.7, -98.3)	(-221.638, -146.0488, 1.870506)	(-222.8124, -145.1846, 2.36301)	(-231.0006, -149.3439, 6.947066)
117	(-171.7, -98.3)	(-221.638, -146.0488, 1.870506)	(-221.7788, -144.8559, 1.99403)	(-231.8348, -149.3744, 7.707476)
118	(-171.7, -98.3)	(-221.638, -146.0488, 1.870506)	(-221.7788, -144.8559, 1.99403)	(-227.381, -147.8357, 6.081323)
119	(-171.7, -98.3)	(-221.638, -146.0488, 1.870506)	(-221.7788, -144.8559, 1.99403)	(-225.7314, -146.5522, 4.450882)

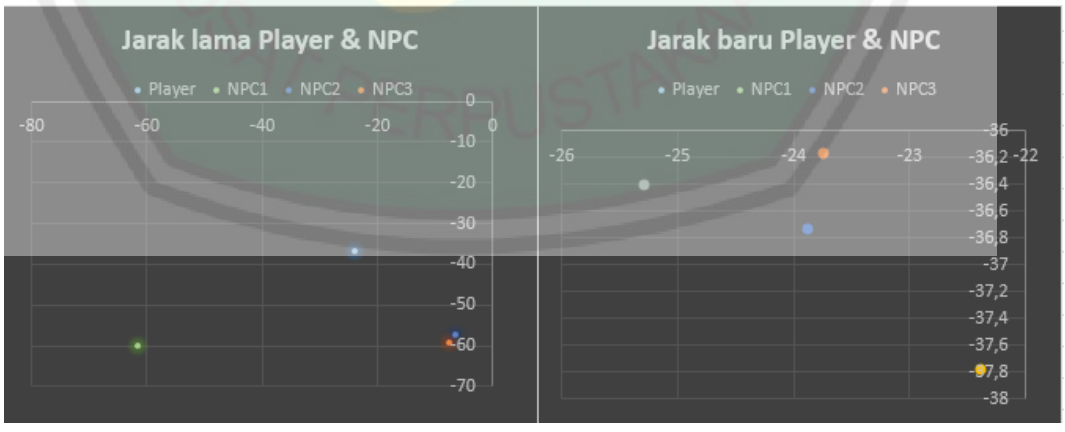
Iterasi	*Target*	NPC		
		NPC 1	NPC 2	NPC 3
	(X, Z)	(X1, Z1, J1)	(X2, Z2, J2)	(X3, Z3, J3)
120	(-171.7, -98.3)	(-221.638, -146.0488, 1.870506)	(-221.7788, -144.8559, 1.99403)	(-223.9618, -146.5694, 2.867225)
121	(-171.7, -98.3)	(-221.638, -146.0488, 1.870506)	(-221.7788, -144.8559, 1.99403)	(-222.9687, -145.8176, 2.047075)
122	(-171.7, -98.3)	(-221.638, -146.0488, 1.870506)	(-221.7788, -144.8559, 1.99403)	(-222.8378, -145.5555, 1.67919)
123	(-171.7, -98.3)	(-221.638, -146.0488, 1.870506)	(-221.7788, -144.8559, 1.99403)	(-224.6943, -147.9097, 1.563601)
124	(-69, -55)	(-171.53, -96.28495, 110.0728)	(-171.7745, -96.23889, 112.5009)	(-171.8569, -97.47874, 110.4756)
125	(-69, -55)	(-155.6334, -96.40958, 97.57481)	(-159.3984, -101.6449, 101.9583)	(-163.8152, -96.75356, 103.3446)
126	(-69, -55)	(-145.0151, -90.20562, 83.43104)	(-148.3272, -93.21726, 87.58847)	(-137.4323, -94.73584, 79.03156)
127	(-69, -55)	(-132.9567, -88.00396, 71.81053)	(-121.3804, -91.54469, 65.7184)	(-126.8836, -91.67969, 66.27407)
128	(-69, -55)	(-128.3293, -81.93474, 66.94733)	(-109.8756, -78.94295, 48.33229)	(-106.3967, -86.49856, 48.71945)
129	(-69, -55)	(-115.8677, -82.69811, 54.77023)	(-101.745, -72.43821, 36.27319)	(-92.41206, -86.8036, 37.61065)
130	(-69, -55)	(-99.87622, -73.41113, 37.66466)	(-88.72565, -77.62526, 29.91053)	(-90.40789, -78.38882, 29.39059)
131	(-69, -55)	(-89.14587, -74.99373, 29.49297)	(-80.05798, -73.41845, 22.4383)	(-81.76227, -72.31671, 20.42949)

Iterasi	*Target*	NPC		
		NPC 1	NPC 2	NPC 3
	(X, Z)	(X1, Z1, J1)	(X2, Z2, J2)	(X3, Z3, J3)
132	(-69, -55)	(-85.92177, -75.73321, 27.74268)	(-76.40604, -71.84439, 18.68516)	(-80.20602, -68.39189, 15.49768)
133	(-69, -55)	(-84.52155, -72.11037, 23.99495)	(-74.84219, -63.60128, 11.57763)	(-77.01077, -64.89559, 12.74975)
134	(-69, -55)	(-87.51334, -69.74158, 22.61343)	(-75.54096, -61.56355, 8.602299)	(-75.65276, -57.72791, 8.503551)
135	(-69, -55)	(-84.91294, -67.99961, 20.35601)	(-71.12682, -59.00734, 5.027201)	(-73.54578, -55.07195, 6.246204)
136	(-69, -55)	(-83.29813, -62.93603, 16.20948)	(-70.00744, -57.2457, 3.87239)	(-70.45046, -51.53844, 3.101223)
137	(-69, -55)	(-81.34113, -62.69748, 16.78433)	(-71.07857, -56.3256, 2.419229)	(-70.13394, -54.31729, 3.734194)
138	(-69, -55)	(-79.85677, -62.87873, 14.43933)	(-69.02002, -57.52679, 2.479359)	(-69.17944, -58.05657, 2.736355)
139	(-69, -55)	(-78.77148, -61.16072, 13.42759)	(-69.02002, -57.52679, 2.479359)	(-69.06365, -56.44471, 2.137048)
140	(-69, -55)	(-77.58975, -59.51486, 11.45283)	(-69.02002, -57.52679, 2.479359)	(-69.06365, -56.44471, 2.137048)
141	(-69, -55)	(-76.59242, -59.84954, 9.334287)	(-69.02002, -57.52679, 2.479359)	(-69.06365, -56.44471, 2.137048)
142	(-69, -55)	(-76.86443, -60.08623, 8.889852)	(-69.02002, -57.52679, 2.479359)	(-69.06365, -56.44471, 2.137048)
143	(-69, -55)	(-72.84549, -58.63396, 6.538557)	(-69.02002, -57.52679, 2.479359)	(-69.06365, -56.44471, 2.137048)

Iterasi	*Target*	NPC		
		NPC 1	NPC 2	NPC 3
	(X, Z)	(X1, Z1, J1)	(X2, Z2, J2)	(X3, Z3, J3)
144	(-69, -55)	(-71.03658, -56.98425, 5.643163)	(-69.02002, -57.52679, 2.479359)	(-69.06365, -56.44471, 2.137048)
145	(-69, -55)	(-69.71819, -59.16353, 4.384053)	(-69.02002, -57.52679, 2.479359)	(-69.06365, -56.44471, 2.137048)
146	(-69, -55)	(-68.33788, -57.8153, 4.492281)	(-69.02002, -57.52679, 2.479359)	(-69.06365, -56.44471, 2.137048)
147	(-69, -55)	(-69.87112, -55.05366, 3.032324)	(-69.02002, -57.52679, 2.479359)	(-69.06365, -56.44471, 2.137048)
148	(-69, -55)	(-71.51033, -54.66248, 1.500194)	(-69.02002, -57.52679, 2.479359)	(-69.06365, -56.44471, 2.137048)
149	(-69, -55)	(-70.68038, -55.85938, 2.745535)	(-69.02002, -57.52679, 2.479359)	(-69.06365, -56.44471, 2.137048)
150	(-69, -55)	(-69.06266, -54.81573, 1.73042)	(-69.02002, -57.52679, 2.479359)	(-69.06365, -56.44471, 2.137048)

Dari data pengujian tersebut, Iterasi 1 – 49 merupakan perhitungan jarak NPC ke *player*, kemudian iterasi 50 – 90 merupakan perhitungan jarak NPC ke tempat wisata A dari tempat terakhirnya, dalam hal ini yaitu di sekitaran tempat *player*, untuk iterasi 91-104 merupakan perhitungan jarak NPC dari tempat wisata A ke tempat wisata B dan iterasi 105 – 123 merupakan perhitungan jarak NPC dari tempat wisata B ke tempat wisata C. Untuk Iterasi 124 – 150 merupakan jarak NPC dari tempat wisata C menuju titik *finish.*

Dari tabel tersebut, menunjukkan bahwa jarak NPC dengan *target* tujuannya semakin tinggi iterasinya, maka akan semakin dekat, namun tidak tepat posisinya dengan titik *target*nya dikarenakan adanya jarak berhenti NPC dengan *player* (1.0). sehingga jarak NPC dengan setiap *target* bernilai kisaran 1 sampai 3. Setelah melakukan uji coba dengan 3 NPC dan 5 *target* (*player*, wisata A, wisata B, wisata C, titik *finish*), menunjukkan hasil bahwa kesemua NPC dapat mencapai semua titik *target*.



Gambar 4.2 Perbandingan posisi awal dan akhir NPC dengan pemain

Posisi Player (x,z) :

- Player (-23,88, -36,73)

Posisi Awal NPC (x,z) :

- NPC 1 (-6.170293, -59.87256)
- NPC 2 (-6.551942, -57.40668)
- NPC 3 (-7.618847, -59.0334)

Posisi *Target* (x,z) :

- Tempat Wisata A (0.48, 1.68)
- Tempat Wisata B (-250.59, -156.27)
- Tempat Wisata C (-171.7, -98.3)
- Titik Finish (-69, -55)

Jarak awal NPC ke *player* :

- NPC 1 (29.16553)
- NPC 2 (27.00375)
- NPC 3 (27.62757)

Usai mengoperasikan sistem dan menghasilkan 49 iterasi, posisi NPC lebih dekat ke pemain sesuai pada gambar 4.2 (kanan).

Posisi akhir NPC (x,z) :

- NPC 1 (-23.74346, -36.15648)

- NPC 2 (-25.29359, -36,39986)
- NPC 3 (-22.39431, -37,76629)

Jarak akhir NPC ke *player (j)* :

- NPC 1 (2.226688)
- NPC 2 (1.877226)
- NPC 3 (2.065665)

Posisi NPC menjadi lebih dekat dengan *player* dan jarak tersebut telah melewati syarat jarak berhenti dengan NPC, hal ini juga berlaku untuk posisi NPC ke 4 *target* lainnya.

Gambar 4.3 perbandingan posisi awal dan akhir NPC dengan 4 *target* lainnya

Terlihat dari Gambar 4.3 dan Tabel 4.4 menunjukkan bahwa jarak NPC dengan *target* mengalami penurunan. Dari pengujian tersebut, menunjukkan bahwa NPC telah berhasil menuju tempat yang ditentukan dengan persentase 100%

Selain melakukan dengan 3 NPC dan 5 *target* (*player*, wisata A, wisata B, wisata C, titik *finish*), uji coba yang lainnya yaitu uji coba sistem untuk mengetahui waktu yang dibutuhkan dalam menyelesaikan semua *target* yang ada. Uji coba ini dimaksudkan untuk menguji efisiensi waktu yang akan digunakan dalam pengaplikasikan kedepan. Uji coba ini akan dilakukan dengan 12 kali percobaan dengan 5 *target* yang urutannya akan berubah kecuali *player* dan titik *finish* yang akan tetap urutannya, dan untuk wisata A tidak bisa ditaruh di antara wisata B dan C. kecepatan bergerak NPC yaitu 100.

Tabel 4.5 Waktu yang dibutuhkan untuk simulasi

No	Jumlah NPC	Urutan *target*	waktu yang dibutuhkan (menit.detik)
1	3	Player - wisata A - wisata B - wisata C - titik finish	2.03
2	3	Player - wisata A - wisata C - wisata B - titik finish	1.57
3	3	Player - wisata B - wisata C - wisata A - titik finish	1.55
4	3	Player - wisata C - wisata B - wisata A - titik finish	1.52
5	5	Player - wisata A - wisata B - wisata C - titik finish	2.09
6	5	Player - wisata A - wisata C - wisata B - titik finish	2.25
7	5	Player - wisata B - wisata C - wisata A - titik finish	1.57
8	5	Player - wisata C - wisata B - wisata A - titik finish	1.55
9	7	Player - wisata A - wisata B - wisata C - titik finish	2.13
10	7	Player - wisata A - wisata C - wisata B - titik finish	2.04
11	7	Player - wisata B - wisata C - wisata A - titik finish	2.00
12	7	Player - wisata B - wisata C - wisata A - titik finish	1.58

Dari percobaan diatas, waktu paling lama yang dibutuhkan untuk menyelesaikan 5 *target* adalah *target* dengan urutan *target player* - wisata A - wisata B - wisata C – titik *finish*, seperti pada tabel nomor 1, 5, dan 9 dengan rata-rata waktu yang dihabiskan 6,26 menit. Sedangkan waktu yang dibutuhkan paling cepat untuk menyelesaikan 5 *target* adalah dengan urutan *target player* – wisata B– wisata C- wisata A- titik *finish* seperti pada **Tabel 4.5** nomer 4, 8, dan 12 dengan rata-rata waktu 4,65 menit.

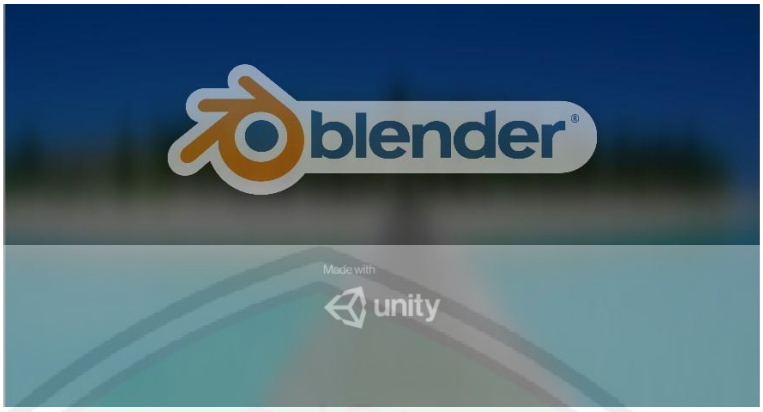
Dari 2 uji coba tersebut, membuktikan bahwa algoritma ABC dapat beroperasi dengan baik karena NPC sudah mencapai semua *target* dengan berkerumunan serta untuk waktu yang dibutuhkan paling cepat yaitu seperti pada **Tabel 4.5**.

4.4 Implementasi Simulasi

4.4.1 Tampilan *Main Menu*

Gambar 4.4 Main Menu

4.4.2 Tampilan *Splash Screen*



Gambar 4.5 Splash Screen

4.4.3 Tampilan *Gameplay*

Gambar 4.6 Gameplay

4.4.4 Tampilan *Pause Menu*

Gambar 4.7 Pause Menu

4.4.5 Tampilan *Credit Menu*

Gambar 4.8 Credit Menu

4.4.6 Tampilan *How To Play*

Gambar 4.9 How to Play

4.4.7 Tampilan Game Over

Gambar 4.10 Game Over

4.5 Integrasi Dalam Islam

Berdasarkan uji coba yang telah dilakukan, jarak NPC dengan *target* tujuannya semakin tinggi iterasinya semakin dekat, namun tidak sesuai dengan titik *target*nya dikarenakan adanya jarak berhenti NPC dengan *player* (1.0). sehingga jarak NPC dengan setiap *target* bernilai kisaran 1 sampai 3. Setelah melakukan uji coba dengan 3 NPC dan 5 *target* (*player*, wisata A, wisata B, wisata C, titik *finish*), menunjukkan hasil bahwa semua NPC dapat mencapai semua titik *target*.

Selain uji coba jarak, ada juga uji coba yang dilakukan untuk efisiensi waktu wisata di Pulau Pari. Ada 12 pengujian dengan berbagai macam rute yang berbeda. Setelah melakukan pengujian tersebut, menunjukkan bahwa ditemukan rute yang tercepat.

Dari hasil di atas, dapat dipastikan apabila game simulasi ini layak untuk diaplikasikan dalam penentuan rute yang akan digunakan untuk wisata di Pulau Pari dengan menentukan efisiensi waktu dan jumlah pengunjung. Penentuan tersebut digunakan dengan harapan pengelola wisata dapat mengatur waktu dan jumlah pengunjung di Pulau Pari untuk mempermudah dalam mengelola wisata, namun tetap mementingkan efisiensi waktu agar tidak terjadi perpanjangan antrian di akhir dikarenakan pengelolaan waktu yang kurang tepat.

Tujuan dari dibangunnya *game* ini adalah untuk mengetahui pergerakan *Non-Playable Character* (NPC) pengunjung wisata Pulau Pari menggunakan algoritma *Artificial Bee Colony* (ABC) dalam mengatasi banyaknya pengunjung yang ada dan waktu yang dibutuhkan dalam menyelesaikan semua wisata. Dengan adanya

simulasi tersebut, diharapkan dapat memudahkan petugas dalam mengatur wisatawan di Pulau Pari, dan wisatawan tidak bingung dalam menuju ke wisata-wisata yang ada di Pulau Pari. Sesuai dalam Al-Qur'an surat Al-Isra' ayat 7, yang berbunyi:

...إِنْ أَحْسَنْتُمْ أَحْسَنْتُمْ لِأَنْفُسِكُمْ وَإِنْ أَسَأْتُمْ فَلَهَا

Artinya:

"Jika kamu berbuat baik (berarti) kamu berbuat baik bagi dirimu sendiri dan jika kamu berbuat jahat maka kejahatan itu bagi dirimu sendiri." (QS. Al-Isra'/ 17 : 7)

Maksud dari ayat tersebut yakni kita sebagai umat manusia seharusnya saling berbuat baik *(hablum minannas*). Apabila kita mempermudah pekerjaan orang lain, maka segala bentuk pekerjaan kita akan dimudahkan oleh Allah *Subhanahu Wa Ta'ala*. Dalam hal ini, peneliti memberikan kemudahan kepada petugas Pulau Pari dalam mengatur wisatawan. Selain itu, wisatawan pun juga ikut terbantu sewaktu mengunjungi wisata-wisata yang ada di Pulau Pari.

BAB V
KESIMPULAN DAN SARAN

5.1 Kesimpulan

Berdasarkan pengujian yang telah dilakukan,, diperoleh kesimpulan sebagai berikut:

1. Dalam uji coba jarak, hasilnya menunjukkan bahwa dalam semua percobaan, semua NPC berhasil dalam mencapai 5 tujuan yang ditentukan dengan persentase 100%,
2. Dalam uji coba waktu, hasilnya dari 12 percobaan, pada percobaan keempat, kedelapan dan keduabelas, dengan masing-masing berbeda jumlah NPC menunjukkan uji coba tercepat. Dari hasil tersebut didapatkan rute pada percbaan keempat, kedelapan dan kedua belas merupakan rute yang paling efisien dalam berwisata mengelilingi Pulau Pari dengan rata-rata waktu 4,65 menit.

5.2 Saran

Penulis memiliki beberapa saran kepada peneliti selanjutnya agar penelitian ini dapat dikembangkan menjadi lebih baik.

1. Menambahkan NPC atau *target* lebih banyak lagi agar dapat digunakan untuk pengujian yang lebih kompleks
2. Diperlukan pengembangan *game* dikarenakan masih belum sempurnanya *game* yang telah dibuat, selain itu pengebangan dilakukan untuk meningkatkan pengalaman penggunaan simulasi ini, khususnya untuk pengelola wisata lain.

DAFTAR PUSTAKA

Ardhianto, Eka., Wiwien H., Edy W. (2012). *Augmented Reality Objek 3 Dimensi dengan Perangkat Artoolkit dan Blender*. Jurnal Teknologi Informasi DINAMIK, Volume 17, No.2, Juli 2012 : 107-117.

Darmawan, Erico., Risal, Laurentius. (2014). *Pemograman Berorientasi Objek C# Yang Susah Jadi Mudah*. Bandung: Informatika Bandung.

Deitel, Paul., Deitel, Harver. (2010). *C# 2012 For Programmers*, London : Pearson Education.

Depdiknas. (2005) *Kumpulan Metode Pembelajaran/ Pendampingan*. Jakarta: Balai Pustaka.

Djamaluddin, H, S. (2016). *Pergerakan NPC Menggunakan Algoritma Boids Dan Artificial Bee Colony Pada Simulasi Mengelilingi Ka'bah (Thawaf)*. Skripsi. Malang : Universitas Islam Negeri Maulana Malik Ibrahim.

Emshoff, A. Simon, (1970). *Rancangan Ulang dan Simulasi, Social Work jurnal*. ISSN : 2339-0042.

Hasan, Innamul. (2017). *Implementasi Algoritma Artificial Bee Colony Untuk Membangkitkan Perilaku NPC Pada Game Survival Horror "Left Alone" Sebagai Media Pengenalan Rumah Cut Nyak Dhien*. Skripsi. Malang :Universitas Islam Negeri Maulana Malik Ibrahim.

Jatiningsih, WIlujeng. (2015). *Pergerakan Serangan Kelompok NPC Berbasis Agen Otonom Menggunakan Algoritma Artificial Bee Colony*. Tesis. Surabaya : Institut Teknologi Sepuluh Nopember.

Karaboga, Dervis. (2010). *Artificial Bee Colony Algorithm*. Scholarpedia. Hal : 2299.

Khosnevis, Behrokh. (1994). Descrate System Simulation. New York : McGraww Hill.

Rahmalia, DInata,. Herlambang, Teguh. (2018), *Optimasi Masalah Transportasi Distribusi Semen Menggunakan Algoritma Artificial Bee Colony*. ISSN : 2477-5401.

Sabri, Aimi Najwa. (2018). *A Study On Bee Algorithm and A* Algorithm for Pathfinding in Games*. Conference: IEEE Symposium on Computer Applications & Industrial Electronics (ISCAIE2018).

Seputra, Yulius Eka Agung. (2014). *Buku Pintar Pemrograman C#*. Yogyakarta : Mediakom.

Sudarmilah, Endah., R. Ferdiana., L. E. Nugroho., A. Susanto (2013). *Tech review: Game platform for upgrading counting ability on Preschool Children*. Prosidingon The 5th International Conference onInformation Technology and Electrical Engineering (lCITEE 2013).

Suparni (2019, Maret). *Wisatawan Mulai Padati Pulau Pari*. Retrieved 29 Januari, 2020, from enjoyjakarta [online] :http://jakarta-tourism.go.id/visit/blog/2019/03/wisatawan-mulai-padati-pulau-pari.

Yao, B,. Yan, Q,. Zhang, M,. Yang, Y. (2017) *Improved Artificial Bee Colony Algorithm For Vehicle Routing Problem With Time Windows*. PLoS ONE 12(9): e0181275. https://doi.org/10.1371/journal.pone.0181275.