IMPLEMENTASI *PARTICLE SWARM OPTIMIZATION* PADA NPC *HUNTER* UNTUK PENCARIAN NPC HEWAN *HUNTER OF FOREST*

SKRIPSI

Oleh :
ACHMAD IHWANY
NIM 11650087

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

HALAMAN PENGAJUAN

IMPLEMENTASI *PARTICLE SWARM OPTIMIZATION* PADA NPC *HUNTER* UNTUK PENCARIAN NPC HEWAN *HUNTER OF FOREST*

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 :
ACHMAD IHWANY
NIM 11650087

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

LEMBAR PERSETUJUAN

IMPLEMENTASI *PARTICLE SWARM OPTIMIZATION* PADA NPC *HUNTER* UNTUK PENCARIAN NPC HEWAN *HUNTER OF FOREST*

SKRIPSI

Oleh :
ACHMAD IHWANY
NIM.11650087

Telah Diperiksa dan Disetujui untuk Diuji :
Tanggal : Juni 2018

Dosen Pembimbing I	Dosen Pembimbing II
Fresy Nugroho, M.T	**Dr. Muhammad Faisal, M.T**
NIP. 19710722 201101 1 001	**NIP. 19740510 200501 1 007**

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

IMPLEMENTASI *PARTICLE SWARM OPTIMIZATION* PADA NPC *HUNTER* UNTUK PENCARIAN NPC HEWAN *HUNTER OF FOREST*

SKRIPSI

Oleh :
ACHMAD IHWANY
NIM.11650087

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

Susunan Dewan Penguji			Tanda Tangan
Penguji Utama	:	Yunifa Miftachul Arif, M.T NIP. 19830616 201101 1 004	( )
Ketua Penguji	:	Hani Nurhayati, M.T NIP. 19780625 200801 2 006	( )
Sekertaris Penguji	:	Fressy Nugroho, M.T NIP. 19710722 201101 1 001	( )
Anggota Penguji	:	Dr. Muhammad Faisal, MT NIP. 19740510 200501 1 007	( )

Mengesahkan,
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 : Achmad Ihwany

NIM : 11650087

Jurusan : Teknik Informatika

Fakultas : Sains dan Teknologi

Judul Skripsi : **IMPLEMENTASI *PARTICLE SWARM OPTIMIZATION* PADA NPC *HUNTER* UNTUK PENCARIAN NPC HEWAN *HUNTER OF FOREST***

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, Juli 2018
Yang membuat pernyataan,

Achmad Ihwany
NIM.11650087

MOTTO

"Hargailah Orang Lain,
Jika Ingin dihargai
Kembali"

(Iwan_Q)

HALAMAN PERSEMBAHAN

Dengan rasa syukur seraya mengharap ridho Ilahi kupersembahkan karya ini kepada :
Orang Tua dan Keluarga Besar H. Mustamar

Yang selalu memberikan kasih sayang, Doa, perhatian, support, dan tidak lupa selalu mengasih bimbingan dalam hal yang membuat semakian baik kedepannya

Tanpa Doa dan biaya dari beliau mungkin saya tidak akan pernah bisa masuk pada kampus yang sangat luar biasa ini. Semoga Allah SWT selalu menjaga dalam setiap langkahnya Orang Tua dan Keluarga Besar H. Mustamar

Terimakasih kepada yang selalu setia mendukung dan mendampingi sampai saat ini

Dwi Januar Bintari

Terimakasih untuk Bapak/Ibu Dosen **Pak Fresy** sebagai dosen pembimbing I dan **Pak Faisal** sebagai dosen Pembimbing II yang selalu sabar membimbing saya dalam menyelesaikan tugas akhir ini. dan **Pak Faisal** sebagai wali dosen yang senantiasa mengawasi perkembangan perkuliahan saya selama beberapa tahun ini. Terimakasih juga kepada ibu bapak dosen pengajar, uztad uztadzah yang telah memberikan ilmu dengan keikhlasan, semogo ilmu yang telah beliau beri ini dapat bermanfaat bagi nusa, bangsa dan agama.

Kepada teman seperjuangan:

M. Mirza (TI UIN), M. Faaris (TI UIN), Moh. Ali Majedi (TI UIN), Awwib Ahsana(TI UIN), Firdaus (TI UIN), Danial Abror (TI UIN), Hayuangga Tino (TI UIN), Nafial Wildan (TI UIN), Nur Hafid (TI UIN), Alvian Burhanuddin (TI UIN), Hamdi Musaad (TI UIN), Emil Enan. (TI UIN), Dwi Januar Bintari (UMM).

yang bersama-sama saling menyemangati satu sama lain dan saling mengingatkan jika lalai.

Kepada para **sahabat TI** angkatan 2011, yang selalu ada untuk membantu sesama. Dan kepada teman-temanku semua yang tidak bisa kusebutkan satu persatu yang selalu membantuku dan menyemangatiku di saat aku susah dan terpuruk.

Semoga Allah SWT melindungi, menyayangi dan menempatkan mereka semuanya pada surganya kelak dan melimpahkan rezeki kepada mereka semua.

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, Nabi besar yaitu, Nabi Muhammad SAW, yang telah menuntun kita dari gaptek menuju update, yaitu Addinul Islam Wal Iman.

Penelitian skripsi yang berjudul "**Implentasi *Particle Swarm Optimization* Pada NPC Hunter Untuk Pencarian NPC Hewan *Hunter Of Forest***" ini ditulis untuk memnuhi salah satu syarat guna memperoleh gelar Sarjana 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. Bapak Fresy Nugroho, 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.
2. Bapak Dr. Muhammad Faisal M.T, selaku dosen pembimbing 2 yang selalu memberi masukan, serta pengarahan dalam penyusunan laporan skripsi ini.
3. Bapak Dr. Muhammad Faisal M.T, selaku dosen wali yang juga selalu memberi pengarahan terkait akademik selama masa study.
4. Dr. Cahyo Crysdian selaku ketua jurusan Teknik Informatika yang mendukung dan mengarahkan skripsi ini.
5. Segenap civitas akademika Fakultas Saintek, Universitas Islam Negeri Maulana Malik Ibrahim Malang terutama seluruh dosen, terimakasih atas segala ilmu dan bimbingannya.

6. Bapak (Alm), Umik, Adik, dan serta seluruh keluarga besar H.Mustamar yang selalu memberikan doa, kasih sayang, semangat, dukungan moril, serta motivasi sampai saat ini, terimakasih banyak.

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, Juni 2018
Penulis

Achmad Ihwany

DAFTAR ISI

DAFTAR GAMBAR

DAFTAR TABEL

DAFTAR DIAGRAM

ABSTRAK

Achmad Ihwany, 11650087, ***Implementasi Particle Swarm Optimization Pada Hunter Untuk Pencarian NPC Hewan "Hunter Of Forest"***, Skripsi, Juruasan Teknik Informatika, Fakultas Saintek, Universitas Islam Negeri Maulana Malik Ibrahim Malang.

Pembimbing : (I) Fresy Nugroho, M.MT (II) Dr. Muhammad Faisal, M.T

Kata kunci :*Particle Swarm Optimization*, *NPC*

Particle swarm optimization (PSO), yang meniru karakter sekelompok burung atau ikan. Algoritma PSO meniru perilaku sosial organisme ini. Perilaku sosial terdiri dari aksi individu dan pengaruh dari individu-individu lain dalam suatu kelompok. Kata partikel membuktikan, misalnya, seekor burung dalam kawanan burung. Pada masing-masing individu atau partikel berkarakter memakai kecerdasannya (intelligence) sendiri dan juga diakibatkan tingkah laku gabungan kolektifnya. Maka dari itu, apabila salah satu dari partikel atau satu burung mendapatkan jalan yang terbaik atau jalan yang tercepat untuk menuju ke sumber makanan, maka burung yang lain atau kelompok yang lain juga akan menemukan jalur tebaik atau tercepat tadi, meskipun lokasinya mereka jauh di kelompok tadi. Non-Player Characters (NPC) atau disebut juga agen adalah suatu entitas dalam game yang tidak dikendalikan secara langsung oleh pemain. NPC dikendalikan secara otomatis oleh komputer. Untuk dapat memperoleh perilaku cerdas dari NPC digunakan kecerdasan buatan atau Artificial Intelligence (AI). Penggunaan AI pada NPC dilakukan dengan pemberian algoritma khusus sesuai dengan perilaku cerdas yang diharapkan.

Penelitian ini membahas mengenai Implementasi Algoritma Particle Swarm Optimization Untuk Menentukan Perilaku NPC Hunter Pada Game "Hunter Of Forest" Dengan tujuan agar dapat menentukan prilaku hunter pada game FPS Hunter Of Forest yang berbasis dekstop.

Hasil dari penelitian ini adalah, Penelitian ini berhasil jalur terpendek menuju hewan buruan yang diterapkan pada NPC Hunter secara dinamis menggunakan Algoritma Particle Swarm Optimization.

ABSTRACT

Achmad Ihwany, 11650087, ***Particle Swarm Optimization*** **Implementation In Hunter For NPC Search Animals *"Hunter Of Forest",*** Thesis, Informatics Engineering Journal, Faculty of Saintek, State Islamic University Maulana Malik Ibrahim Malang.

Counselor: (I) Fresy Nugroho, M.MT (II) Dr. Muhammad Faisal, M.T

Key words : *Particle Swarm Optimization, NPC*

Particle swarm optimization (PSO), which mimics the character of a group of birds or fish. The PSO algorithm mimics the social behavior of this organism. Social behavior consists of individual action and the influence of other individuals in a group. The word particle proves, for example, a bird in a flock of birds. In each individual or particle character using his own intelligence and also due to his collective joint behavior. Therefore, if one of the particles or one bird gets the best path or the fastest path to the food source, the other bird or other group will also find the fastest or fastest route, although their location is far in the group before . Non-Player Characters (NPC) or also called agents are an in-game entity that is not directly controlled by the player. NPC is automatically controlled by computer. To be able to obtain intelligent behavior from NPC used artificial intelligence or Artificial Intelligence (AI). The use of AI in NPC is done by providing specific algorithms according to the expected intelligent behavior.

This study discusses the Implementation of Particle Swarm Optimization Algorithm for Determining the Behavior of NPC Hunter in "Hunter Of Forest" Game In order to be able to determine the hunter behavior in the desktop-based FPS Hunter Of Forest game.

The result of this research is, This research succeed the shortest path to game hunting applied to NPC Hunter dynamically using Particle Swarm Optimization Algorithm.

الملخص

احواني، احمد ، ١١٦٥٠٠٨٧، بحث الحيوانات هنتر الغابة، تنفيذ الجسيمات سرب الأمثل في هنتر ل.
الرسائل العلمية. قسم الهندسة المعلوماتية ، كلية العلوم والتكنولوجيا. جامعة الدولة الإسلامية مولانا مالك إبراهيم مالانج

تحت المشرف : ١) فرشي نقروهو الماجستيرة ٢) محمد فيصل الماجستيرة

الكلمة ا ساسية: *NPC , تحسين سرب الجسيمات*

، والذي يحاكي شخصية مجموعة من الطيور أو الأسماك. (PSO) تحسين سرب الجسيمات السلوك الاجتماعي لهذا الكائن الحي. يتكون السلوك الاجتماعي من العمل PSO تحاكي خوارزمية الفردي وتأثير الأفراد الآخرين في المجموعة. كلمة الجسيمات تثبت ، على سبيل المثال ، طائرًا في قطيع من الطيور. في كل شخصية فردية أو جسيمية باستخدام ذكائه الخاص وأيضاً بسبب سلوكه الجماعي المشترك. لذلك ، إذا كان أحد الجسيمات أو طائر واحد يحصل على أفضل مسار أو أسرع مسار إلى مصدر الغذاء ، فسيجد الطائر الآخر أو المجموعة الأخرى أيضًا المسار الأسرع أو الأسرع ، أو تسمى أيضًا (NPC) على الرغم من أن موقعه بعيد جدًا في المجموعة قبل ، أحرف غير المشغِّل تلقائيًا بواسطة NPC بالكيانات هي كيان في اللعبة لا يتحكم فيه المشغل مباشرة. يتم التحكم في الكمبيوتر. لتكون قادرة على الحصول على سلوك ذكي من المجلس الوطني للمرأة استخدام الذكاء من خلال NPC في AIS الاصطناعي أو الذكاء الاصطناعي (منظمة العفو الدولية). يتم استخدام. توفير خوارزميات محددة وفقًا للسلوك الذكي المتوقع

في لعبة NPC تتناول هذه الدراسة تطبيق خوارزمية تحسين سرع الجسيمات لتحديد سلوك صائد المعتمدة على سطح صياد الغابة FPS حتى تتمكن من تحديد سلوك الصياد في لعبة " صياد الغابة." المكتب.

بشكل NPC Hunter نتيجة هذا البحث ، هذا البحث ينجح أقصر طريق إلى لعبة الصيد المطبق على حيوي باستخدام خوارزمية سرب تحسين الجسيمات.

BAB I

PENDAHULUAN

1.1 Latar Belakang

Negara Indonesia merupakan Negara yang mempunyai keaneka ragaman tingkat variasi bentuk kehidupan dengan jumlah yang paling tinggi dibandingkian dengan Negara-negara yang lainnya. Dengan sekian banyaknya keragaman tingkat variasi bentuk kehidupan yang ada di Indonesia, beberapa diantaranya sudah terancam kepunahan. Kepunahan sendiri disebabkan oleh beberapa faktor salah satunya adalah perburuan hewan secara liar dan juga penebangan hutan-hutan yang dilakaukan oleh manusia.

Pemerintah telah mengupayakan berbagai cara agar hewan-hewan tersebut tidak mengalami kepunahan. Beberapa jenis tumbuhan dan hewan yang terancam mengalami kepunahan telah dilindungi dalam undang-undang. Apabila ada seseorang ada yang melanggar undang undang tersebut, bisa terkena sanksi yang berupa hukuman pidana. Dengan sanksi yang diberikan oleh pemerintah tersebut diharapkan perburuan hewan dan tumbuhan langka, memberikan efek jera dan pemburuan ataupun perdagangan hewan dan tumbuhan yang masuk dalam daftar hewan yang dilindungi UU ini bisa diberhentikan.

Menurut aturan undang-undang, konservasi sumber daya alam (SDA) ialah pengelola sumber daya alam (SDA) tingkat variasi bentuk kehidupan yang penggunaannya dilakukan dengan cermat demi menanggung kelanjutan persediannya dan terus menjaga dengan menambahkan tingkat keragaman dengan

angkanya. Cagar alam ialah Kawasan Suaka Alam (KSA), adapun untuk saat ini cagar alam, taman hutan raya, dan taman wisata alam lainnya menjadi Kawasan Pelestarian Alam (KPA).

Penjualan hewan dilindungi atau ilegal yang sekelas hewan langka dalam beberapa tahun ini sangat mengkhawatirkan. Dianggap hewan-hewan yang berkelas endemic dari berbagai ragam pulau di Indonesia kini sudah semakin mudah menyebar ke berbagai Negara di seluruh dunia. Penyebara satwa liar tersebut terjadi melalui social media yang yang semakin besar-besaran perkembangannya dalam 10 tahun terakhir ini. Fakta tersebut kini semakin mengkhawatirkan karena kekuatan social media dewasa ini sudah diakui sangat besar oleh semua kalangan dunia.

Pengetahuan terkait pentingnya menjaga alam dan keanekaragaman hayati akan berdampak terhadap kelestarian lingkungan. Maka dari itu pengkajian ini untuk mengembangkan game FPS tentang perlindngan satwa langka yang ada di area konservasi dari perburuan liar berbasis desktop. Dimana game ini bertujuan untuk meningkatkan kesadaran dalam menjaga kelestarian hewan langka.

Sebagai mana firman Allah SWT pada Surat *Al A'raf* ayat 56 :

وَلَا تُفْسِدُوا فِي الْأَرْضِ بَعْدَ إِصْلَاحِهَا وَادْعُوهُ خَوْفًا وَطَمَعًا ۚ إِنَّ رَحْمَتَ اللَّهِ قَرِيبٌ مِنَ الْمُحْسِنِينَ

"Dan janganlah kamu membuat kerusakan di muka bumi, sesudah (Allah) memperbaikinya dan berdoalah kepadanya dengan rasa takut (tidak akan diterima) dan harapan (akan dikabulkan). Sesungguhnya rahmat Allah amat dekat kepada orang orang yang berbuat baik."(Q.S. Al-A'raf[7]: 56)

Dalam ayat ini menjelaskan bahwa bahwa Allah SWT, memerintahkan manusia untuk tidak membuat kerusakan di muka bumi ini setelah Allah menciptakan alam ini dengan sempurna, penuh harmoni, serasi dan sangat seimbang untuk mencukupi kebutuhan makhluk-Nya.

Ayat ini merupakan penyampaian kepada kita untuk menjalankan amanat dan menjaga titipan yang telah Allah berikan. Amanat tersebut yaitu menjaga titipan Allah berupa alam beserta isinya yang merupakan nikmat yang sangat besar. Karena dari pepohonanlah kita dapat memperoleh oksigen untuk bernapas. Selain itu tumbuhan pula dapat menjaga kelangsungan hidup kita, sama halnya dengan hewan, air, dan masih banyak lagi nikmat Allah yang disediakan di alam.

Di dalam penelitian ini, peneliti akan menerapkan algoritma *PSO* untuk pencarian hewan pada NPC hunter. Karena pencarian jalur terbaik menggunakan algoritma *PSO* dapat menghasilkan data yang cepat dan akurat, disebabkan inisiasi partikelnya menyesuaikan pada jarak terbaik.

1.2 Identifikasi Masalah

Berdasarkan dari pembahasan latar belakang di atas, maka identifikasi masalah pada penelitian ini ialah: Apakah algoritma *PSO* dapat diimplementasikan pada NPC hunter untuk melakakukan pencarian NPC hewan pada *game FPS hunter of forest*?

1.3 Batasan Masalah

a. Dalam *game* ini dibatasi hanya dengan melindungi hewan langka yang berada di kawasan konservasi dari perburuan liar.

1.4 Tujuan Penelitian.

a. Mengimplementasikan algoritma *PSO* algoritma pada NPC hunter untuk melakakukan pencarian NPC hewan pada *game FPS hunter of forest* yang berbasis desktop.

1.5 Manfaat Penelitian

Manfaat dari pembuatan aplikasi ini ialah untuk menyampaikan pengetahuan tentang perlindungan satwa langka yang ada di Indonesia dari perburuan liar dalam bentuk *game FPS* yang lebih menarik dan menyenangkan.

BAB II

TINJAUAN PUSTAKA

2.1 TEORI PENUNJANG

2.1.1 *Game*

a. Pengertian *Game*

Game berasal dari bahasa inggris yaitu memilik makna atraksi atau permainan atau pertandingan. Pengertiannya ialah aktifitas yang digunakan sebagai menenagkan pikiran yang lagi stres atau juga dibuat untuk menyenangkan yang mempunyai ketentuan dalam *game* tersebut makan ada yang kalah dan juga ada yang menang. *Game* juga mempunyai macam-macam perbedaan dari segi tujuannya seperti: *Education Games, Art Game,* dan lain-lain.

a. Elemen Dasar *game genre FPS*

Pendapat Teresa Dillon (Futurelab SandfordWilliamson 2005 games and learning) bagian-bagian dasar di dalam *game* adalah sebagai berikut :

1. *Game Rule*

Game rule adalah sebuah pengaturan petunjuk bagaimana mempraktikkan, fungsi objek dan karakter didalam *game* tersebut.

2. Plot

Plot berisi sebuah penjelasan mengenai keadaan yang hendak dimainkan oleh *player* pada sebuah *game* dengan rinci.

3. Theme

Dalam theme ini terdepat pesan baik dalam *game* tersebut.

4. Character

Terdapat beberapa peran, ada peran utama ataupun peran lainnya yang memeiliki cirri dan sifat tertentu.

5. Object

Object adalah bagian penting yang sering di gunakan para *player* guna untuk mencari solusi dalam *game* tersebut, dan *player* juga mempunya keahlian serta pengetahuan untuk memainkan *game* tersebut.

6. Text, grafik dan sound

Game kebanyakan adalah sebuah perpaduan dari alat tulis atau media teks, grafik atau sound, meskipun tidak semuanya ada dalam sebuah *game.*

7. Animation

Sebuah animasi ini harus selalu ada pada sebuah *game,* dikhususkan pada pergerakan karakter yang ada pada sebuah *game,* properti dari sebuah objek

8. User Interface

Adalah sebuah sifat atau fitur untuk menghubungkan *user* dengan *game.*

Adapun komponen utama dalam *game* ada 4 macam yaitu:

a. Adanya *player*,
b. Adanya area di mana para *player* berinteraksi,
c. Adanya peraturan dalam *game* tersebut,
d. Adanya misi dalam *game* tersebut yang ingin dicapai.

b. Jenis-Jenis *Game*

1. *Shooting* (tembak-tembakan)

Video game jenis ini memerlukan kecepatan refleks, koordinasi mata-tangan, juga timing, inti dari game jenis ini adalah tembak, tembak dan tembak. Contoh : *Call OF Dutty*, dan *Crysis*.

2. *Fighting* (Pertarungan)

Game yang permainannya memerlukan refleks dan koordinasi mata dan tangan dengan cepat, tetapi inti dari *game* ini adalah pernguasaan hafalan jurus. Contoh : *Mortal Combat* dan *Tekken*.

3. *Adventure* (Petualangan)

Game yang lebih menekankan pada jalan cerita dan kemampuan berfikir pemain dalam menganalisa tempat secara visual, memecahkan teka-teki maupun menyimpulkan berbagai peristiwa. Contoh : *Kings Quest*.

4. Simulasi, Kontruksi, Manajemen

Video game jenis ini seringkali menggambarkan dunia di dalamnya sedekat mungkin dengan dunia nyata dan memperhatikan dengan detil berbagai faktor. Contoh : *The Sims.*

5. Strategi

Game jenis ini memerlukan koordinasi dan strategi dalam memainkan permainan ini. Kebanyakan *game* strategi adalah *game* perang. Contoh : *Age Of Empire*.

6. Olahraga

Game ini merupakan adaptasi dari kenyataan, membutuhkan kelincahan dan juga strategi dalam memainkannya. Contoh : *Pro Evolution Soccer*dan *NBA*.

7. *Puzzle*

Game teka-teki, pemain diharuskan memecahkan teka-teki dalam *game* tersebut. Contoh : *Tetris, Minesweeper* dan *Bejeweled*.

8. *Edugames* (Edukasi)

Video game jenis ini dibuat dengan tujuan spesifik sebagai alat pendidikan, entah untuk belajar mengenal warna untuk balita, mengenal huruf dan angka, matematika, sampai belajar bahasa asing. *Developer* yang membuatnya, harus memperhitungkan berbagai hal agar *game* ini benar-benar dapat mendidik, menambah pengetahuan dan meningkatkan ketrampilan yang memainkannya. Target segmentasi pemain harus pula disesuaikan dengan tingkat kesulitan dan design visual ataupun animasinya. Contoh *Edugames* : *Bobi Bola, Dora theexplorer*. (Suindarti, 2011)

c. *Non-Player Characters (NPC)*

Autonomous character merupakan kelompok *otonomous agent* yang diwujudkan buat pemakaian media interaktif dan komputer animasi semacam *games* dan *virtual reality*. Yang mana agen tersebut digunakan untuk menggantikan aktor dalam skenario atau permainan dan mempunyai keahlian untuk improvisasi gerakan mereka. Pada sebuah *game* atau pertunjukan, karakter otonom umumnya disebut NPC (*Non-Player Character)*. Reynold (1999)

Pada adegan dari *game hunter of forest* ini, *hunter* merupakan contoh dari NPC. Perilaku seorang *hunter* yaitu para *hunter*akan berusaha mencari keberadaan satwa langka pada kawasan konservasi kemudian akanberusaha memburu satwa dilindungi dengan cara menembaki satwa yang dilindungi tersebut. Dan ketika *hunter* bertemu dengan player yang dalam *game* ini beraksi sebagai polisi hutan,

maka *hunter* akan memperhitungkan kondisi apakah *hunter* akan berburu, menyerang ataupun melarikan diri.

2.1.2 Konservasi

Konservasi ialah pemeliharaan atau pelestarian. Menurut harfiah, konservasi atau pelestarian berasal dari bahasa Inggris, yaitu: *Conservation* yang mempunya arti sebgai pemeliharaan atau perlindungan. Adapun *Conservation* menurut ilmu lingkungan adalah sebagai berikut:

- Usaha kemampuan pada pemakaian kekuatan, pembuatan, transmisi, dan distribusi yang berdampak kepada penurunan kekuatan di pihak lain mempersiapkan bantuan yang serupa levelnya.
- Usaha pemeliharaan dan penataan yang cermat kepada daerah dan sumber daya alam (SDA)
- (fisik) Penataan tentang kapasitas spesifik yang berjalan dengan normal semasa reaksi kiamia atau transformasi fisik.
- Usaha suaka dan pemeliharaan dalam waktu yang lama untuk daerah atau lingkungan.
- Satu kepercayaan bahwasanya lingkungan dari suatu daerah bisa dikelola, sementara itu keaneka-ragaman genetik dari jenis spesies bisa berjalan dengan melindungi lingkungan alaminya.

(https://id.wikipedia.org/wiki/Konservasi)

2.1.3 Konservasi Hewan Langka

Populasi hewan langka yang semakin menurun adalah salah satu kejadian yang harus diperhatikan. Menurut *International Union forConservation of Nature*, berkurangnya spesies yang saat ini terjadi diperkirakan 1.000 kali lebih cepat dibandingkan dengan yang seharusnya. Hal ini dikarenakan banyaknya sebuah masalah, seperti kerusakan habitat, perubahan iklim polusi, perdagangan *illegal*, dan lainnya. Menurut *World Wildlife Fund*, diketahui adanya perdagangan hewan langka setiap harinya semakin meningkat yang dilakukan oleh para pemburu sehingga mengakibatkan krisis yang mengancam kelangsungan hidup banyak spesies. Para pemburu yang biasanya dilakukan oleh masyarakat yang kurang mampu melakukan penjualan hewan disebabkan banyaknya suatu permintaan dari pembeli untuk hewan-hewan tersebut. Masalah ini merupakan yang kedua terbesar sesudah perusakan habitat dalam ancaman kepada kelangsungan hidup spesies secara keseluruhan. Hal itu menyebabkan kerusakan keanekaragaman hayati dan juga degradasi pada ekosistem yang ada. Begitu juga dengan masalah tentang hewan-hewan yang ada di seluruh dunia sudah mulai mengalami ancaman kepunahan. Sebanyak 8.462 jenis hewan telah dinyatakan terancam punah, termasuk mamalia, reptil, serangga, ikan, dan amphibi. (BBC, 2009)

Untuk mengatasi hal tersebut, di Indonesia sendiri pemerintah telah memakai Undang-Undang No. 5 Tahun 1990 dimanaUndang-Undang tersebut berbunyi tentang pelestarian Sumber Daya Alam Hayati dan Ekosistemnya, dan Pemerintah juga mendukung Undang-Undang ini tentang perlindungan hewan. Tetapi selain itu semua pihak pemerintah juga berpartisipasi guna untuk

mendukung proses pelestarian hewan langka. Salah satunya yaitu dengan cara memberi edukasi kepada masyarakat guna menjaga hewa-hewan langka agar tetap hidup di habitat aslinya. Karena tidak bisa dipungkiri lagi bahwasanya penyebab utama kerusakan alam dan pemburuan satwa liar adalah dilakukan oleh manusia maka dari itu manusia harus bertanggung jawab guna untuk melestarikan lingkungannya kembali. Tetapi kesadaran dalam melestarikan hewan-hewan langka tidak bisa diciptakan dengan isntan, melainkan perlu ditanamkan sejak dini. Adapun Edukasi pada anak-anak sendiri bisa dilakukan dengan berbagai cara diantaranya yaitu dengan bermain *game*. (http://www.bbc.com/indonesia/majalah-38441802)

2.1.4 Algoritma *Particle Swarm Optimization*

Particle swarm optimization (PSO), yang meniru karakter sekelompok burung atau ikan. Algoritma *PSO* meniru perilaku sosial organisme ini. Perilaku sosial terdiri dari aksi individu dan pengaruh dari individu-individu lain dalam suatu kelompok. Kata partikel membuktikan, misalnya, seekor burung dalam kawanan burung. Pada masing-masing individu atau partikel berkarakter memakai kecerdasannya (*intelligence*) sendiri dan juga diakibatkan tingkah laku gabungan kolektifnya. Maka dari itu, apabila salah satu dari partikel atau satu burung mendapatkan jalan yang terbaik atau jalan yang tercepat untuk menuju ke sumber makanan, maka burung yang lain atau kelompok yang lain juga akan menemukan jalur tebaik atau tercepat tadi, meskipun lokasinya mereka jauh di kelompok tadi.

Pada *Particle Swarm Optimization (PSO),* kelompok diperkirakan memiliki parameter khusus dan perkomponen tempat asalnya berada dalam posii yang *random* dan pada ruang multidimensi. Masing-masing komponen diperkirakan

mempunyai dua karakter yaitu: posisi dan kecepatan. Masing-masing komponen berputar pada ruang/space khusus dan akan selalu menghafal jalur terpendek yang sudah ditempuh atau didapatkan pada titik makanan atau nilai fungsi objektif. Masing-masing komponen memberikan berita atau jalur terpendeknya pada komponen yang lain serta menyamakan tempat dan kecepatan masing-masing bersumber pada berita yang masuk tentang tempat atau posisi tersebut. Contohnya adalah seperti berikut: perilaku sekelompok burung dalam kawanan burung. Walaupun masing-masing burung memiliki dependensi dalam hal kecerdasan, kebanyakan ia akan mengikuti kebiasaan (*rule*) sebagai berikut :

1. Posisi burung tidak berada amat dekat dengan burung-burung yang lainnya
2. Terbang seekor burung memusatkan pada umumnya yang dilakukan burung-burung lainnya
3. Seekor burung akan mengambil tempat pada umumnya burung-burung yang lainnya dan tetap memperhatikan rute kelompok burung sehingga jaraknya tidak terlalu jauh

Maka dari itu karakter gerombolan burung didasarkan pada perpaduan dari 3 faktor sederhana berikut:

- Kohesi – terbangnya secara bersama
- Separasi – jarak tidak terlalu dekat
- Penyesuaian(alignment) - mengikuti arah bersama

Maka *Particle Swarm Optimization (PSO)* ditingkatkan dengan berlandaskan pada acuan berikut:

1. Apablia ada seekor burung sedang menghampiri incaran atau makanan (atau bisa juga minimum atau maximum suatu fungsi) maka dengan gesit dia akan mengirimkan sebuah berita pada segerombolan burung lainnya pada kelompok khusus.
2. Segerombolan burung lainnya segera ikut ke sumber makanan yang telah di kirimkan tadi, melainkan tidak secara bersamaan.
3. Terdapat partikel yang akan bergantung atas pandangan masing-masing burung, ialah ingatan jalur yang dilintasi sebelumnya.

Dalam algoritma *Particle Swarm Optimization (PSO)* ini, pencarian jalan keluar dikerjakan sebab satu populasi yang terjadi pada sebagian komponen. Populasi akan didirikan dengan acak dan batas nilai minimum serta maximum. Masing-masing komponen mencerminkan tempat maupun jalan keluarnya pada persoalan yang dihadapi. Masing-masing komponen melaksanakan pencarian jalan keluarnya yang terbaik serta melewati kawwasan pencarian *(search space)*. Dalam kejadian ini akan dilakukan dengan cara masing-masing komponen melaksanakan adaptasi pada tempat terbaik dari komponen tersebut (*local best*) serta adaptasi pata tempat komponen terbaik dari semua kelompok (*global best*) sepanjang melewati ruang pencarian. Maka, pemencaran pengetahuan atau berita berlangsung di dalam komponen itu independen dan jarak satu komponen pada komponen utama dari semua kelompok semasa operasi pencarian jalan keluar. kemudian memulai operasi pencarian untuk mencari tempat terbaik pada masing-masing komponen pada jumlah iterasi khusus sampai memperoleh tempat yang relatif *steady* atau sampai batasan iterasi yang sudah ditentukan. Pada masing-masing iterasi, masing-masing jalan keluar yang mencerminkan dengan tempat

komponen, makan akan dipertimbangkan performansinya memakai gaya memasukkan jalan keluar tersebut kedalam *fitness function*.

Masing-masing komponen diperkerjakan sama halnya dengan sebentuk titik pada satu ukuran ruang khusus. Lalu akan ada dua faktor yang memberikan karakter kepada status komponen pada ruang penelusuran yaitu tempat komponen dan kecepatan komponen, Kennedy and Eberhart (1995).

Dibawah ini adalah perumusan ilmu hitung yang menguraikan tempat dan kecerdasan/kecepatan komponen pada dimensi tertentu :

$(t) = xi1(t), xi2(t), \ldots, xiN(t)$ (1)

$V(t) = vi1(t), vi2(t), \ldots, viN(t)$ (2)

dimana

X = posisi partikel

V = kecepatan partikel

i = indeks partikel

t = iterasi ke-t

N = ukuran dimensi ruang

Dibawah ini adalah contok ilmu hitung yang menguraikan teknik memperbarui status komponen Kennedy and Eberhart [1995]:

$V(t) = V(t-1) + c1r1(XiL - Xi(t-1)) + c2r2(XG - Xi(t-1))$ (3)

$(t) = V(t) + Xi(t-1)$ (4)

dimana

$XiL = xi1\ ,2\ L\ ,\ldots,xiN\ L$ mencerminkan *local best* dari partikel ke-i. Adapun $XG = xi1\ G,2\ G,\ldots,xiN\ G$ mencerminkan *global best* dari semua kelompok. Adapun c1 dan c2 yaitu suatu konstanta yang bernilai jelas yang sering disebut sebagai learning faktor. Kemudian r1 dan r2 ialah suatu angka acak yang memiliki nilai antara 0 sampai 1. Persamaan (3) digunakan sebagai menjumlahkan kecepatan komponen baru yang bersumber kepada kecepatan sebelumnya, jarak antara tempat waktu ini dengan tempat terbaik komponen *(local best)*, serta jarak antara tempat saat ini dengan tempat terbaik semua kelompok *(global best)*. Kemudian komponen terbang mengarah tempat yang pertama berlandaskan perumpamaan (4). Kemudian algoritma *Particle Swarm Optimization (PSO)* ini dijalankan dengan sejumlah iterasi khusus sampai memperoleh parameter pemberhentian, sehingga akan memperoleh jalan keluar yang bertempat pada *global best*. Model untuk dicontohkan pada ruang dengan dimensi khusus dengan beberapa iterasi maka pada masing-masing iterasi, tempat komponen tentu semakin terarah pada sasaran yang akan ditempuh (minimasi atau maksimasi fungsi). Ini dilakukan sampai maksimum iterasi ditempuh atau juga bisa digunakan kriteria pemberhentian lainnya.

Algoritma *Particle Swarm Optimization (PSO)* mencakup metode sebagai berikut :

1. Bangkitkan tempat asal sebanyak komponen beserta kecerdasan asalnya dengan acak.
2. Catatan fitness dari setiap komponen berlandaskan tempatnya.

3. Tentukan komponen beserta fitness teroptimum, dan stabilkan menjadi *Gbest*. Pada masing-masing komponen, maka *Pbest* asal tentu cocok pada tempat asal.

Lakukan tindakan berikutnya secara berulang mencapai selesai kriteria mencukupi

1. Memakai *Pbest* serta *Gbest* yang ada, update kecerdasan masing-masing komponen memakai perbandingan .3. Kemudian dengan kecerdasan baru yang diperoleh, update tempat masing-masing komponen memakai perbandingan .4.
2. Catatan fitness pada masing-masing komponen.
3. Tentukan komponen melalui fitness teroptimum, dan stabilkan menjadi *Gbest*. Pada masing-masing komponen, maka *Pbest* dengan mencocokkan tempat saat ini dengan *Pbest* dari iterasi sebelumnya.
4. Cek selesai kriteria. Apabila terpenuhi, berhenti. Apabila tidak terpenuhi maka, kembali ke 1. Budi Santosa (2011)

2.1.5 Implentasi *Particle Swarm Optimization*

Misalkan ada fungsi sebagai berikut

minimasi *f(x)*

dimana

$$x^{(B)} \leq x \leq x^{(A)}$$

dimana $x(B)$ adalah batas bawah dan $x(A)$ adalah batas atas dari x. Prosedur *Particle Swarm Optimization* dapat dijabarkan dengan langkah-langkah sebagai berikut:

1. Asumsikan bahwa ukuran kelompok atau kawanan (jumlah partikel) adalah *N*. Untuk mengurangi jumlah evaluasi fungsi yang diperlukan untuk menemukan solusi, sebaiknya ukuran *N* tidak terlalu besar, tetapi juga tidak terlalu kecil, agar ada banyak kemungkinan posisi menuju solusi terbaik atau optimal. Jika terlalu kecil sedikit kemungkinan menemukan posisi partikel yang baik. Terlalu besar juga akan membuat perhitungan jadi panjang. Biasanya digunakan ukuran kawanan adalah 20 sampai 30 partikel.
2. Bangkitkan populasi awal x dengan rentang (B) dan (A) secara *random* sehingga didapat $x1, x2, \dots, xN$. Partikel j dan kecepatannya pada iterasi i dinotasikan sebagai $xj(i)$ dan $vj(i)$, sehingga partikel-partikel awal ini dinotasikan

$$x_1(0), x_2(0), \dots, x_N(0)$$

Vector

$$v_1(0), v_2(0), \dots, v_N(0)$$

disebut partikel atau vektor koordinat dari partikel (seperti kromosom dalam algoritma genetika). Selanjutnya lakukan evaluasi nilai fungsi tujuan untuk setiap partikel dan nyatakan dengan

$$f(x_1(0)), f(x_2(0)), \dots, f(x_N(0)).$$

3. Hitung kecepatan dari semua partikel. Semua partikel bergerak menuju titik optimal dengan suatu kecepatan tertentu. Awalnya semua kecepatan dari partikel diasumsikan sama dengan nol. Set iterasi $i = 1$.

4. Pada iterasi ke-i, temukan 2 parameter penting untuk setiap partikel *j* yaitu:

   a) Nilai terbaik sejauh ini dari (*i*) (koordinat partikel j pada iterasi *i*) dan nyatakan sebagai $P_{best,j}$ dengan nilai fungsi tujuan paling rendah (kasus minimasi), $f[x_j(i)]$, yang ditemui sebuah partikel *j* pada semua iterasi sebelumnya. Nilai terbaik untuk semua partikel *xj* (*i*) yang ditemukan sampai iterasi ke-i, G_{best} dengan nilai fungsi tujuan paling kecil/minimum diantara semua partikel untuk semua iterasi sebelumnya $f[x_j(i)]$.

   b) Hitung kecepatan partikel *j* pada iterasi ke *i* dengan rumus sebagai berikut

$$v_j(i) = v_j(i-1) + c_1 r_1[P_{best,j} - x_j(i-1)] + c_2 r_2[G_{best} - x_j(i-1)], \quad j = 1, 2, \dots, N \tag{6}$$

   dimana *c1* dan *c2* masing-masing adalah *learning rates* untuk kemampuan individu *(cognitive)* dan pengaruh sosial (kawanan), dan *r1* dan *r2* bilangan random yang berdistribusi uniforml dalam interval 0 dan 1. Jadi parameter *c1* dan *c2* menunjukkan bobot dari memory *(position)* sebuah partikel terhadap memory (posisi) dari kelompok *(swarm)*. Nilai dari *c1* dan *c2* biasanya adalah 2 sehingga perkalian $c1r1$ dan $c2r2$ memastikan bahwa partikel partikel akan mendekati target sekitar setengah selisihnya.

   c) Hitung posisi atau koordinat partikel *j* pada iterasi ke-i dengan cara

$$x_j(i) = x_j(i-1) + v_j(i), \quad j = 1, 2, \dots, N \tag{7}$$

   Evaluasi nilai fungsi tujuan untuk setiap partikel dan nyatakan sebagai

$$f[x_1(i)], f[x_2(i)], \dots, f[x_N(i)]$$

5. Cek apakah solusi yang sekarang sudah konvergen. Jika posisi semua partikel menuju ke satu nilai yang sama, maka ini disebut konvergen. Jika belum

konvergen maka langkah 4 diulang dengan memperbarui iterasi $i = i + 1$, dengan cara menghitung nilai baru dari *Pbest,j* dan *Gbest*. Proses iterasi ini dilanjutkan sampai semua partikel menuju ke satu titik solusi yang sama. Biasanya akan ditentukan dengan kriteria penghentian *(stopping criteria)*, misalnya jumlah selisih solusi sekarang dengan solusi sebelumnya sudah sangat kecil.

Contoh

Misalkan kita mempunyai persoalan optimasi dengan satu variabel sebagai berikut

$$\min f(x) = (100 - x)2 \quad (8)$$

dimana $60 \leq x \leq 120$

1. Tentukan jumlah partikel $N = 4$.

   Tentukan populasi awal secara secara *random,* misalkan

   $x1(0) = 80,$

   $x2(0) = 90,$

   $x3(0) = 110,$

   $x4(0) = 75.$

2. Evaluasi nilai fungsi tujuan untuk setiap partikel xj (0) untuk $j = 1,$ 2, 3, 4. dan nyatakan dengan

   $f1 = (80) = 400,$

   $f2 = (90) = 100,$

   $f3 = (110) = 100,$

   $f4 = (75) = 625,$

3. Tentukan kecepatan awal $v1(0) = v2(0) = v3(0) = v4(0) = 0.$

Tetapkan iterasi I = 1; lalu kelangkah nomer 4.

4. Temukan

   Pbest,1 = 80,

   Pbest,2 = 90,

   Pbest,3 = 110,

   Pbest,4 = 75,

   Gbest = 90.

   Hitung *v(j)* dengan *c1 = c2 = 1*. Misalkan nilai *random* yang didapat, *r1* = 0,4, *r2* = 0,5, dengan rumus

$$V_j(i) = V_j(i-1) + |c_1 r_1 [P_{best,j} - x_j(i-1)] + c_2 r_2 [G_{best,j} - x_j(i-1)]$$

   diperoleh

   $v1(1) = 0 + 0.4(80 - 80) + 0.5(90 - 80) = 5$

   $v2(1) = 0 + 0.4(90 - 90) + 0.5(90 - 90) = 0$

   $v3(1) = 0 + 0.4(110 - 110) + 0.5(90 - 110) = -10$

   $v4(1) = 0 + 0.4(75 - 75) + 0.5(90 - 75) = 7.5$

   Sedangkan untuk nilai *x* adalah

   $v1(1) = 80 + 5 = 85$

   $x2(1) = 90 + 0 = 90$

   $x3(1) = 110 - 10 = 100$

   $x4(1) = 75 + 7.5 = 82.5$

5. Evaluasi nilai fungsi tujuan sekarang pada partikel *xj* (1),

   $f1(1) = f(85) = 225$,

   $f2(1) = f(90) = 100$,

$f3(1) = f(100) = 0,$

$f4(1) = f(82.5) = 306.25.$

Sedangkan pada iterasi sebelumnya kita dapatkan

$f1(0) = f(80) = 400,$

$f2(0) = f(90) = 100,$

$f3(0) = f(110) = 100,$

$f4(0) = f(75) = 625.$

Nilai dari *f* dari iterasi sebelumnya tidak ada yang lebih baik sehingga Pbest untuk masing-masing partikel sama dengan nilai *x*-nya, Gbest = 100.

Cek apakah solusi *x* sudah *konvergen,* dimana nilai *x* saling dekat. Jika tidak, tingkatkan ke iterasi berikutnya $I = 2$. Lanjtkan ke lagkah 4.

1. *Pbest*,1 = 85,

   Pbest,2 = 90,

   Pbest,3 = 100,

   Pbest,4 = 82.5,

   Gbest = 100

   Hitung kecepatan baru dengan $r1 = 0.3$ dan $r2 = 0.6$ (ini hanya sekedar contoh untuk menjelaskan penghitungan, dalam implementasi angka ini dibangkitkan secara *random*).

   $v1(2) = 5 + 0.3(85 - 85) + 0.6(100 - 85) = 14$

   $v2(2) = 0 + 0.3(90 - 90) + 0.6(100 - 90) = 6.$

   $v3(2) = -10 + 0.3100 - 100) + 0.6(100 - 100) = -10$

   $v4(2) = 7.5 + 0.3(82.5 - 82.5) + 0.6(100 - 82.5) = 18$

Sedangkan untuk nilai x adalah

$x1(2) = 85 + 14 = 99$

$x2(2) = 90 + 6 = 96$

$x3(2) = 100 - 10 = 90$

$x4(2) = 82.5 + 18 = 100.5$

2. Evaluasi nilai fungsi tujuan sekarang pada partiekl $xj(2)$,

   $f1(2) = f(99) = 1$,

   $f2(2) = f(96) = 16$,

   $f3(2) = f(90) = 100$

   $f4(2) = f(100.5) = 0.25$.

   Jika dibandingkan dengan nilai f dari iterasi sebelumnya, ada nilai yang lebih baik dari nilai f sekarang yaitu $f3(1) = 0$, sehingga Pbest untuk partikel 3 sama dengan 100, dan Gbest dicari dari min{1, 16, 0, 0.25} = 0 yang dicapai pada $x3(1) = 100$. Sehingga untuk iterasi berikutnya Pbest = (99, 96, 100, 100.5) dan Gbest = 100.

Cek apakah solusi sudah *konvergen*, dimana nilai x saling dekat. Jika belum *konvergen*, set i= 3, masuk ke iterasi berikutnya. Lanjutkan ke langkah berikutnya dengan menghitung kecepatan v dan ulangi langkah-langkah selanjunya sampai mencapai *konvergen*.

2.1.6 Modifikasi *Particle Swarm Optimization*

Dalam implementasinya, ditemukan bahwa kecepatan partikel dalam *Particle Swarm Optimization* standard *diupdate* terlalu cepat dan nilai minimum fungsi tujuan yang dicari sering terlewati. Karena itu kemudian dilakukan modifikasi atau perbaikan terhadap algoritma *Particle Swarm Optimization*

standard. Perbaikan itu berupa penambahan suatu term inersia θ untuk mengurangi kecepatan pada formula update kecepatan. Biasanya nilai θ dibuat sedemikian hingga semakin besar iterasi yang dilalui, semakin mengecil kecepatan partikel. Nilai ini bervariasi secara linier dalam rentang 0.9 hingga 0.4. Secara matematis perbaikan ini bisa dituliskan.

$$v_j(i) = \theta v_j(i-1) + c_1 r_1 [P_{best,j} - x_j(i-1)] + c_2 r_2 [G_{best} - x_j(i-1)], \quad j = 1, 2, \ldots, N \tag{9}$$

Bobot inersia ini diusulkan oleh Shi and Eberhart [1998] untuk meredam kecepatan selama iterasi, yang memungkinkan kawanan burung menuju *(converge)* titik target secara lebih akurat dan efisien dibandingkan dengan algoritma aslinya. Formula 10.9 adalah modifikasi terhadap formula 10.6. Nilai bobot inersia yang tinggi menambah porsi pencarian global *(global exploration)*, sedangkan nilai yang rendah lebih menekankan pencarian lokal *(localsearch)*. Untuk tidak terlalu menitikberatkan pada salah satu bagian dan tetap mencari area pencarian yang baru dalam ruang berdimensi tertentu, maka perlu dicari nilai bobot inersia (θ) yang secara imbang menjaga pencarian global dan lokal. Untuk mencapai itu dan mempercepat konvergensi, suatu bobot inersia yang mengecil nilainya dengan bertambahnya iterasi digunakan dengan formula.

$$\theta(i) = \theta_{max} - \left(\frac{\theta_{max} - \theta_{min}}{i_{max}}\right) i \tag{10}$$

dimana θmax dan θmin masing-masing adalah nilai awal dan nilai akhir bobot inersia, $imax$ adalah jumlah iterasi maksimum yang digunakan dan i adalah iterasi yang sekarang. Biasanya digunakan nilai θmax = 0.9 dan θmin = 0.4. Perubahan atau modifikasi formula untuk *mengupdate* kecepatan ini seperti *step size* α dalam algoritma *Steepest Descent*, dimana nilai α yang terlalu besar akan memungkinan

suatu optimum lokal akan terlewati sehingga algoritma justru menemukan optimum lokal yang lain yang tidak lebih baik nilainya. Untuk implementasi ini digunakan fungsi Himmelblau.

$f = (x12 + x2 - 11)2 + (x1 + x22 - 7)2$. Budi Santosa (2011)

2.1.7 Algoritma *A Star*

Algoritma A* adalah algoritma yang dikemukakan oleh Hart, Nilsson, dan Raphael pada tahun 1968. Algoritma A* merupakan salah satu algoritma Branch & Bound atau disebut juga sebagai sebuah algoritma untuk melakukan pencarian solusi dengan menggunakan informasi tambahan (heuristik) dalam menghasikan solusi yang optimal. Hapsari Tilawah (2011)

a. Terminologi Dasar Algoritma A*

Beberapa terminologi dasar yang terdapat pada algoritma ini adalah starting point, current node, simpul, neighbor node, open set, closed set, came from, harga (cost), walkability, target point. Starting point adalah sebuah terminologi untuk posisi awal sebuah benda. Current node adalah simpul yang sedang dijalankan dalam algortima pencarian jalan terpendek. Simpul adalah petak-petak kecil sebagai representasi dari area pathfinding. Bentuknya dapat berupa persegi, lingkaran, maupun segitiga. Neighbor node adalah simpul-simpul yang bertetangga dengan current node. Open set adalah tempat menyimpan data simpul yang mungkin diakses dari starting point maupun simpul yang sedang dijalankan. Closed set adalah tempat menyimpan data simpul sebelum current node yang juga merupakan bagian dari jalur terpendek yang telah berhasil didapatkan. Came from adalah tempat menyimpan data ketetanggaan dari suatu simpul, misalnya y came from x artinya neighbor node y dari current node x. Harga (F) adalah nilai yang

diperoleh dari penjumlahan nilai G, jumlah nilai tiap simpul dalam jalur terpendek dari starting point ke current node, dan H, jumlah nilai perkiraan dari sebuah simpul ke target point. Target point yaitu simpul yang dituju. Walkability adalah sebuah atribut yang menyatakan apakah sebuah simpul dapat atau tidak dapat dilalui oleh current node. Hapsari Tilawah (2011)

b. Fungsi Heuristik

Algoritma A* menerapkan teknik heuristik dalam membantu penyelesaian persoalan. Heuristik adalah penilai yang memberi harga pada tiap simpul yang memandu A* mendapatkan solusi yang diinginkan. Dengan heuristik yang benar, maka A* pasti akan mendapatkan solusi (jika memang ada solusinya) yang dicari. Dengan kata lain, heuristik adalah fungsi optimasi yang menjadikan algoritma A* lebih baik dari pada algoritma lainnya. Namun heuristik masih merupakan estimasi / perkiraan biasa saja Sama sekali tidak ada rumus khususnya. Artinya, setiap kasus memiliki fungsi heuristik yang berbedabeda. Algoritma A* ini bisa dikatakan mirip dengan algoritma Dijkstra, namun pada algoritma Dijkstra, nilai fungsi heuristiknya selalu 0 (nol) sehingga tidak ada fungsi yang mempermudah pencarian solusinya. Nilai ongkos pada setiap simpul n menyatakan taksiran ongkos termurah lintasan dari simpul n ke simpul target (target node), yaitu: F (n) = nilai taksiran lintasan termurah dari simpul status n ke status tujuan. Dengan kata lain, F (n) menyatakan batas bawah (lower bound) dari ongkos pencarian solusi dari status n. Fungsi heuristik yang terdapat pada algoritma A* untuk menghitung taksiran nilai dari suatu simpul dengan simpul yang telah dilalui adalah:

$$F(n) = G(n) + H(n)$$

Dimana : F (n) = ongkos untuk simpul n.

G (n) = ongkos mencapai simpul n dari akar.

H (n) = ongkos mencapai simpul tujuan dari simpul n.

c. Algoritma

Algortima A* secara ringkas langkah demi langkahnya adalah sebagai berikut :

1. Tambahkan strating poin ke dalam openset
2. Ulangi langkah berikut
   a. Carilah biaya F terendah pada setiap simpul dalam open set. Node dengan biaya F terendah kemudian disebut current node.
   b. Masukkan ke dalam closed set
   c. Untuk setiap 8 simpul (neighbor node) yang berdekatan dengan current node:
   - Jika tidak walkable atau jika termasuk closed set, maka abaikan.
   - Jika tidak ada pada open set, tambahkan ke open set.
   - Jika sudah ada pada open set, periksa apakah ini jalan dari simpul ini ke current node yang lebih baik dengan menggunakan biaya G sebagai ukurannya. Simpul dengan biaya G yang lebih rendah berarti bahwa ini adalah jalan yang lebih baik. Jika demikian, buatlah simpul ini (neighbor node) sebagai came from dari current node, dan menghitung ulang nilai G dan F dari simpul ini.

   d. Stop ketika anda:

- Menambahkan target point ke dalam closed set, dalam hal ini jalan telah ditemukan, atau

- Gagal untuk menemukan target point, dan open set kosong. Dalam kasus ini, tidak ada jalan.

3. Simpan jalan. Bekerja mundur dari target point, pergi dari masing-masing simpul ke simpul came from sampai mencapai starting point. Itu adalah jalan Anda. Hapsri Tilawah (2011)

d. Kompleksitas Algoritma

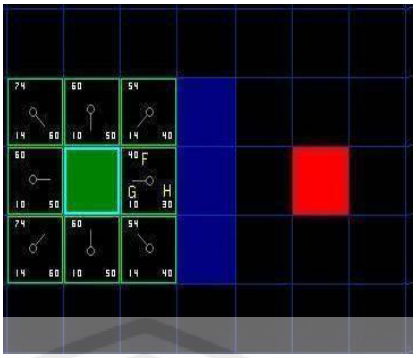
Kompleksitas waktu dari A * tergantung pada heuristiknya. Dalam kasus terburuk, jumlah simpul yang diperluas adalah eksponensial dari panjang solusi (jalur terpendek), tetapi polinomial ketika ruang pencarian adalah pohon, ada sebuah simpul tujuan tunggal, dan fungsi heuristik h memenuhi kondisi berikut:

$$| H (x) - h * (x) | = O (\log h * (x))$$

Dimana h* adalah heuristik optimal, biaya yang tepat yang didapat dari x ke tujuan. Dengan kata lain, kesalahan dari h tidak akan tumbuh lebih cepat dari logaritma dari "heuristik sempurna" h * yang mengembalikan jarak yang sebenarnya dari x ke tujuan. Hapsari Tilawah (2011)

e. Penerapan Algoritma A*

Asumsikan bahwa ada seseorang yang ingin pergi dari titik simpul ke simpul B dan terdapat tembok memisahkan kedua simpul. Hal ini digambarkan di bawah ini, dengan kotak hijau menjadi simpul awal A, dan kotak merah menjadi simpul tujuan B, dan kotak biru yang menjadi tembok.

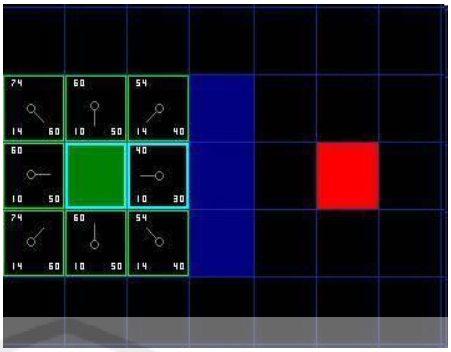


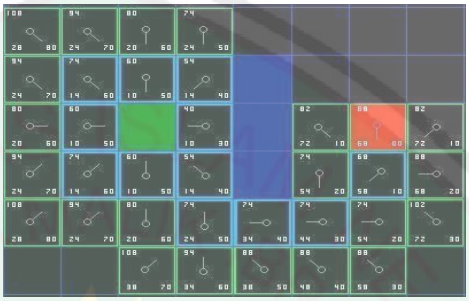
Kemudian memulai pencarian dengan melakukan hal berikut

1. Mulailah di simpul awal A dan menambahkannya ke "open set".
2. Lihatlah semua simpul berdekatan dengan simpul awal yang dapat dijangkau atau walkable, abaikan simpul yang merupakan tembok. Tambahkan simpul tersebut ke dalam “open set”. Untuk masing- masing simpul, simpan simpul A sebagai "came from"-nya.

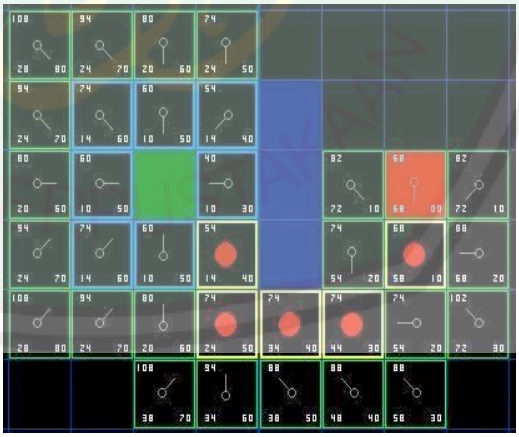
Hapus simpul A dari open set, dan menambahkannya ke dalam "closed set”.

3. Hapus simpul dari open set dan menambahkannya ke closed set.
4. Periksa semua simpul yang berdekatan. Abaikan simpul yang termasuk closed set atau not walkable (tembok).
5. Jika simpul yang berdekatan sudah ada pada open set, periksa apakah jalan tersebut merupakan yang lebih baik.





- Ulangi langkah 4), 5), dan 6) sampai simpul target termasuk ke dalam closed set, di mana simpul itu terlihat seperti ilustrasi di bawah ini.Untuk menentukan path solusi hanya dengan mulai di simpul target dan bekerja mundur bergerak dari satu simpul ke came from-nya sampai akhirnya kembali ke simpul awal. Seperti terlihat seperti ilustrasi berikut.



2.2 PENELITIAN TERKAIT

Diperoleh sejumlah riset yang ada terikatnya dengan riset yang dilakukan ini, ialah :

2.2.1 Pergerakan Otonom Pasukan Berbasis *Algoritma Boids* Menggunakan Metode *Particle Swarm Optimization*

Penelitian ini dilakukan oleh Mu'min, Mochammad Hariadi, Supeno Mardi Sisiki Nugroho. Jurusan Teknik Elektro, Fakultas Teknologi Industri, Insitut Teknologi Sepuluh Nopember pada tahun 2015.

Berlandaskan dari pengetesan yang dikerjakan bisa ditarik kesimpulan sebagai berikut: Algoritma *Particle Swarm Optimization (PSO)* dapat digunakan buat memaksimalkan model *boids* dengan memaksimalkan *koefisien* dari vektor berputar sebagai meminimkan fungsi *cost* dan *flocking* agar tampak optimal.

2.2.2 Implementasi Algoritma *Particle Swarm* untuk Menyelesaikan *Sistem Persamaan Nonlinear*

Penelitian ini dilakukan oleh Ardiana Rosita, Yudhi Purwananto, Rully Soelaiman. Jurusan Teknik Informatika, Fakultas Teknologi Informasi, Institut Teknologi Sepuluh Nopember pada tahun 2012.

Berlandaskan dari pengetesan yang dikerjakan bisa ditarik kesimpulan sebagai berikut: Algoritma *Particle Swarm* yang dianjurkan oleh Jaberipour yaitu kegunaannya sebagai menyempurnakan sistem persamaan nonlinear, Algoritma *Particle Swarm* yang dianjurkan oleh Jaberipour mempunyai kerja yang bagus dalam menelusuri jalan keluar yang terbaik. Pada perihal ini bisa kita buktiakn dengan hasil uji coba yang pertama, jalan keluar yang bisa mendekati jalan keluar eksak setiap kegunaan/fungsinya, Jumlah komponen yang difungsikan sama

algoritma *Particle Swarm* ialah nilai terendah 100 komponen tidak berpengaruh kepada nilai akhir.

2.2.3 PENGEMBANGAN ALGORITMA *PARTICLE SWARM OPTIMIZATION* UNTUK OPTIMALISASI *DISPERSI BATCH* PADA *PROSES PRODUKSI*

Penelitian ini dilakukan oleh Misra Hartati, Iwan Vanany, Budi Santosa, Jurusan Teknik Industri, Fakultas Teknologi Industri, Institut Teknologi Sepuluh Nopember pada tahun 2012.

Berlandaskan dari pengetesan yang dikerjakan bisa ditarik kesimpulan sebagai berikut: bentuk algoritma *metaheuristik* yang dianjurkan ialah algoritma *particle swarm optimization (PSO)*. Atas adanya pertolongan algoritma *Particle Swarm Optimizatin (PSO)* dapat membuat kalkulasi dispersi *batch* demi *problem* yang amat kompleks. Bisa dilihathasil dari perhitungan optimasi dispersi *batch* dengan memakai algoritma *Partcle Swarm Optimization (PSO)* yaitu sebanyak 20 dispersi yang terdiri dari 12 *dawnward dispersion* dan 8 *upward dispersion*. Serta meminimkan *batch dispersion* diinginkan dapat meringankan pada saat melaksanakan *tracking* dan *tracing* pada produk apabila ada masalah pencemaran dan bisa meminimkan produk gagl/penarikan produk. Namun kekurangannya pada riset ini ialah tidak memikirkan biaya yang akan dikeluarkan. Maka dari itu, saya berharap pada riset berikutnya agar supaya memikirkan biaya yang akan dikeluarkan maka dapat kelihatan sebera besar penurunan biaya yang akan terjadi pada operasi produksi dengan mengerjakan penanggulangan terhadap *dispersi bach.*

2.2.4 IMPLEMENTASI ALGORITMA *PARTICLE SWARM OPTIMIZATION* UNTUK PENENTUAN POSISI STRATEGIS *AGENT* PADA *SIMULASI ROBOT SEPAK BOLA DUA DIMENSI*

Penelitian ini dilakukan oleh Galih Hermawan, Jurusan Teknik Informatika, Fakultas Teknik dan Ilmu Komputer, UNIKOM pada tahun 2012

Berlandaskan dari pengetesan yang dikerjakan bisa ditarik kesimpulan sebagai berikut: produk metode penetapan tempat agent memakai algoritma *Particle Swarm Optimization (PSO)* yang diimplementasikan untuk tim yang belum memakai *Particle Swarm Optimization (PSO)* menurut garis besarnya mempunyai presentasi makin bagus dari pada tim yang belum menggunakan *Particle Swarm Optimization (PSO)*. Tetapi apabila disamakan oleh tim Helios si pemenang, presentasi justru ketinggalan jauh. Faktor tersebut dikarenakan keahlian agent selaku individu *(basic skill)* dan tertinggal jauh apabila disamakan dengan keahlian agents pada tim Helios. Sehingga cara menetukan tempat agent tidak begitu berakibat saat bertarung dengan tim Helios. Pada persoalan ini, *basic skill player* seperti: *dribbling*, *passing*, dan lain-lain harus dinaikkan. Manfaat dari algoritma *Particle Swarm Optimization (PSO)* bisa makin dinaikkan presentasinya dengan mengawasi faktor-faktor yang lainnya seperti: stamina, jumlah halangan yang ada di muka *player*, *speed* berlari *player*, *speed* bergeraknya bola, tempat tim lainnya, tempat musuh, dan lain-lain.

2.2.5 Pencarian Jalur Terbaik Menggunakan *Particle Swarm Optimization* untuk mengoptimasi Lalu Lintas Kendaraan

Penelitian ini dilakukan oleh Safril Rizki Waluyo, Jurusan Teknik Elektro, Fakultas Teknologi Industri, Institut Teknologi Sepuluh Nopember, pada tahun 2010.

Berlandaskan dari pengetesan yang dikerjakan bisa ditarik kesimpulan sebagai berikut: *Particle Swarm Optimization* (PSO) untuk mengoptimasi lalu lintas telah berhasil dibuat. Dan pada pengujian sistem yang dibuat lalu bisa ditangkap kesimpulannya seperti berikut: Penerapan *Particle Swarm Optimization* (PSO) sebagai obyek multi agent pada proses pencarian jalur menuju titik goal dapat dilakukan pada Blender dengan waktu tempuh sesuai dengan kecepatan yang ditentukan. Dari hasil pengujian didapatkan rata rata waktu yang digunakan untuk menempuh jalan 1 adalah 28.625 detik dan untuk menempuh jalan 2 adalah 1 menit 58 detik. Jumlah penggunaan partikel sebanding dengan besar dan kerumitan jalan raya. Pada Percobaan partikel yang dibutuhkan untuk menyelesaikan permasalahan jalan satu blok adalah 4 buah dan untuk menyelesaikan permasalahan jalan 3 blok dibutuhkan 25 buah.

2.3 METODE PENELITIAN

Peneliti membagi pengerjaan penelitian ini menjadi beberapa tahap, antara lain :

1. Studi *literature*

Pada tahap ini dilakukan berbagai pengumpulan data literatur-literatur terkait penelitian ini sebagai berikut:

- Literatur di dapatkan dari buku, jurnal, atau skripsi terdahulu

- Literatur berisi informasi tentang pembuatan *game* edukasi mengenai konservasi satwa langka dan juga tentang algoritma *Particle Swarm Optimization* yang akan diterapkan pada *NPC hunter* yang ada dalam *game*.

Untuk selanjutnya akan dilakukan analisis terhadap hasil pengumpulan data dari *literature* yang telah didapatkan.

2. Perancangan dan desain *game*

Pada tahap ini, perancangan *game* terdiri atas perancangan proses-proses utama dan desain *game* yang terdiri atas desain menu *game* dan desain utama dari *game* itu sendiri.

3. Pembuatan *game*

Pada tahap ini, akan dilakukan pembangunan *game* dengan menuliskan bahasa pemrograman pada *compiler* sehingga menghasilkan *game* yang sesuai dengan hasil perancangan.

4. Uji coba dan evaluasi

Pada tahap ini, akan diketahui hasil dari implementasi algoritma *Particle Swarm Optimization* pada *game* konservasi satwa langkaapakah sudah optimal atau belum. Terjadi kesalahan atau tidak pada *game* tersebut. Apabila tidak terjadi kesalahan, akan dilakukan pemeliharaan secara berkala dan aplikasi *game* tersebut disesuaikan dengan kondisi lingkungan tersebut diaplikasikan serta akan dilakukan pengembangan pada aplikasi *game* tersebut.

5. Penyusunan Laporan

Dalam pembuatan laporan ini nantinya diharapkan bisa bermanfaat bagi penelitian-penelitian lebih lanjut yang mana penelitian ini berisi hasil dari seluruh

dokumentasi pelaksanaan penelitian. Berikut adalah permodelan dari pembagian pengerjaan penelitian.

Diagram 2.1 Pemodelan Pengerjaaan Penelitian

2.4 *Pathfinding*

Umumnya pada game yang membutuhkan pencarian lokasi mengaplikasikan *Pathfinding,* karena berfungsi untuk mencari rute dengan menggunakan algoritma pencarian, dimana penerapan pathfinding dipengaruhi beberapa faktor seperti map dan objek atau elemen yang terdapat pada suatu *game.*

BAB III

DESAIN DAN RANCANGAN GAME

3.1 Deskripsi Game

Game ini adalah *game* berjenis Aksi – *Shooting*. Tujuan dari *game* ini dibuat adalah agar bisa mencontohkan satu kejadian perlindungan satwa langka dari para pemburu liar. Dengan dibuatnya *game FPS* ini, *player* dilawankan dalam kondisi dimana para *hunter* sedang mencari satwa langka incaran mereka pada area konservasi. Dalam menjalankan permainan, *player* akan bertindak sebagai polisi hutan yang bertugas untuk melindungi satwa langka yang ada di kawasan konservasi dari serangan para *hunter*. Dalam situasi tersebut *player* harus berkeliling area konservasi untuk memberantas para *hunter*. Selain itu, *player* hendak dikasih notifikasi pertama kali memulai *game* dan sesudah membereskan tugas, yang dimana pada start *game player* akan mendapatkan notifikasi tentang adanya *hunter* yang memasuki area konservasi sedangkan notifikasi yang diberikan setelah menyelesaikan tugas berisi perintah untuk kembali ke markas untuk melaporkan situasi dan bersiap untuk tugas berikutnya. Dalam *game* ini terdapat beberapa situasi dimana *player* akan dihadapkan dengan jumlah *hunter* yang akan bertambah, dimana akan membuat situasi yang lebih seru dan menantang disetiap levelnya.

Game ini adalah berbasis dekstop guna untuk sarana menengahi antara pengetahuan dan pemahaman *game* tersebut melewati perangkat visual atau audio yang ada pada *game* tersebut. Incaran pemakai *game* tersebut ialah semua orang

yang mendalami tentang perlindungan hewan. *Game* ini adalah *game* single player dan dibuat dengan karakter dan objek 3D.

3.2 *Storyline*

Game bimbingan atau edukasi ialah menggambarkan mengenai *player* yang tengah melaksanakan tugas sebagai polisi hutan dimana *player* akan ditugaskan untuk memberantas para *hunter* yang memasuki area konservasi dengan tujuan memburu para satwa langka. Dimana disetiap level akan ada *hunter* yang memasuki area konservasi dan akan bertambah jumlahnya pada level berikutnya setelah *player* menyelesaikan tugas di level tersebut. Sistem penyelesaian tugas polisi hutan adalah dapat mengusir ataupun membunuh para *hunter* yang memasuki area konservasi sehingga area konservasi tersebut aman dari para *hunter*, apabila *player* dapat menyelesaikan tugas maka dia akan mendapatkan notifikasi untuk kembali ke markas melaporkan situasi area konservasi dan akan melanjutkan tugas berikutnya tetapi apabila *player* belum menyelesaikan tugas dan darah/nyawanya habis karena terkena serangan *hunter* maka akan kalan atau *game over* dan *player* harus memulai dari start permainan.

3.3 *Finite State Machine* (FSM)

3.3.1 *Finite State Machine* (FSM) Hunter

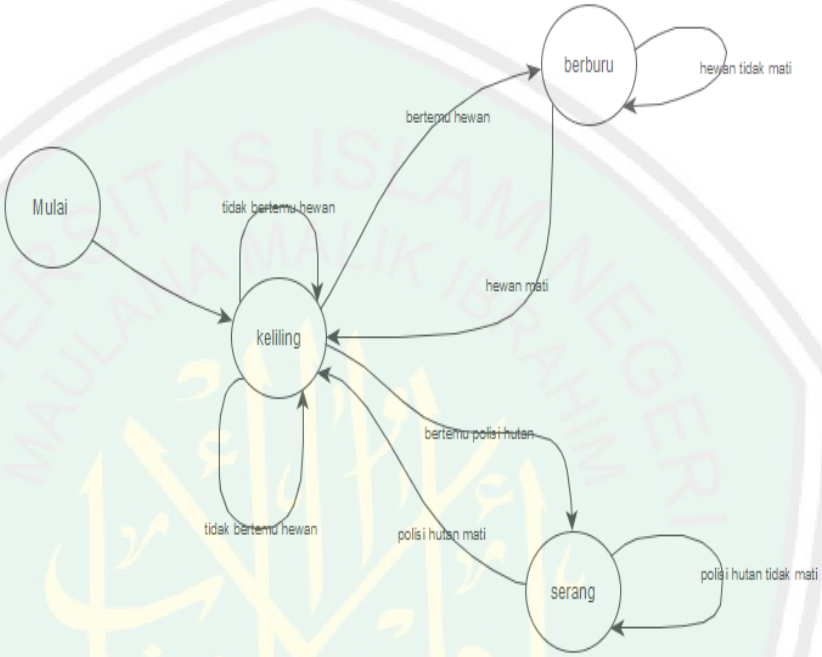
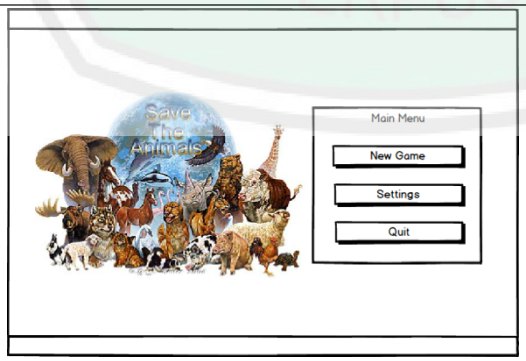


Diagram 3.1 *FSM* Hunter

3.4 Rancangan *Interface*

1.		Tampilan Awal *Game* Terdapat 3 *button* yaitu: New *Game*, *Setting, Quit*	• *Button New Game* merupakan tombol untuk memulai *game* • *Button Setting* merupakan tombol untuk membuka jendela pengaturan

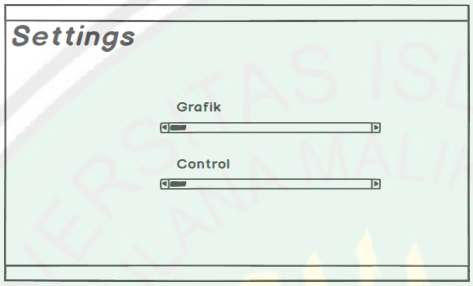
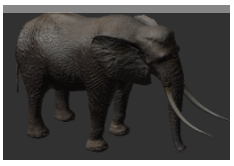
			• *Button Quit* merupakan tombol untuk keluar dari *game*
2		Jendela *Setting* Terdapat 2 *control* bar yaitu Grafik dan *Control*	• Grafik untuk mengatur kualitas gambar pada *game* • *Control* untuk mengatur sensitivitas *mouse* pada *player*

Table 3.1 Rancangan *Interface*

3.5 Deskripsi Karakter dan Objek

3.5.1 Karakter Hewan

Karakter hewan merupakan NPC yang ada di dalam *game*



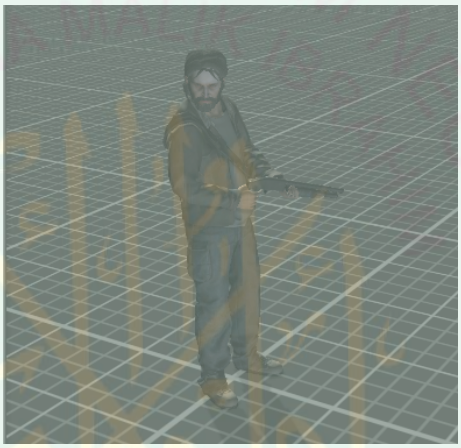
Gambar 3.1 karakter hewan Gajah

Gambar 3.2 karakter hewan Singa

Gambar 3.3 karakter hewan Badak

3.5.2 Karakter *Hunter*

Karakter *hunter* merupakan karakter musuh yang memburu hewan



Gambar 3.4 karakter *Hunter*

3.5.3 Objek Senjata

Senjata dibawah ini digunakan untuk alat perang karakter utama dengan musuh.

Gambar 3.5 *Shootgun*

Gambar 3.6 *Snipper*

Gambar 3.7 *Pistol*

Gambar 3.8 *Scars*

Gambar 3.9 *RPG*

Gambar 3.10 *AK-47*

3.6 *StoryBoard*

Game ini berlatar pada suatu wilaah konservasi yang melindungi berbagai jenis hewan langka dari ancaman perburuan liar, akan tetapi semakin hari semakin banyak terjadi perburuan liar yang terjadi di area konservasi ini. Hal ini semakin menjadikan populasi satwa langka semakin sedikit dan menuju kepada kepunaahan. Oleh karena itu polisi hutan *(player)* ditugaskan untuk lebuh memperdekat penjagaan dikawasan area konservasi.

Sebagai bentuk tindakan untuk menjaga kelangsungan populasi hewan langka di area konservasi, maka polisi hutan *(player)* ditugaskan untuk menangkap ataupun menyerang para pemburu liar *(hunter)* yang melakukan perburuan liar di dalam area konservasi.

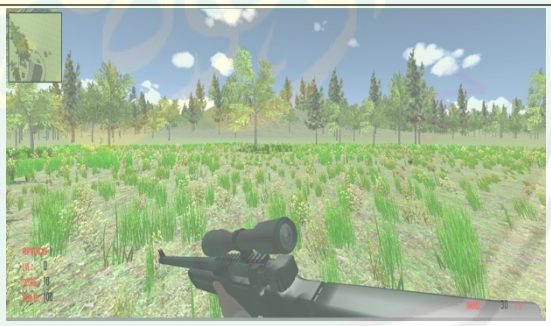
Berikut Skenario *Game* yang akan dimainkan :

- Terdapat beberapa *hunter* yang melakukan aktivitas perburuan liar di dalam area konservasi.
- Jumlah *hunter* akan selalu bertambah banyak disetiap levelnya.

- *Player* akan di-*spawn* pada markas polisi hutan.
- *Player* harus mencari lokasi *hunter* untuk dapat menangkap ataupun melawan mereka.
- *Score*/ nilai pada *game* ini ditentukan oleh seberapa banyak *hunter* yang berhasil dikalahkan.

Berikut *scane* yang terjadi didalam *game :*

Table 3.2 Rancangan *StoryBoard*

Scene	Gambar	Deskripsi
Chapter 1		*Spawn area Player* merupakan titik awal *Player* ketika *game* baru saja dimulai
Chapter 2		*Player* bertugas berpatroli diarea konservasi, *player* mendengar suara tembakan, *player* mencari tahu asal suara tembakan

Chapter 3		*Player* melihat aktivitas perburuan liar oleh *hunter*
*Chapter*4		Terjadi saling serang antara *player* dan *hunter*
*Chapter*5		Setiap *player* berhasil mengalahkan *huter*, maka *player* mendapatkan tambahan *score*
*Chapter*6		*Player* kembali melanjutkan patroli

3.7 Perancangan Algoritma *Particle Swarm Optimization* dalam game

Algoritma *Particle Swarm Optimization* (PSO) dikenalkan sama James Kennedy dan Russell Eberhart pada tahun 1995. Adapun algoritma ini mencontoh

pada kelakuan binatang, ibarat sekawanan burung dan sekelompok ikan pada satu gerombolan (kelompok).

Particle Swarm Optimization (PSO) merupakan bagian pada teknik komputasi evolusioner yang memiliki kecocokan dengan Genetic Algorithm, yaitu diawali dengan membangkitkan populasi dengan acak. Namun *Particle Swarm Optimization (PSO)* tiada mempunyai operator evolusi, ialah crossover dan pemindahan. Hal lain yang berbeda dengan Genetic Algorithm atau teknik komputasi evolusioner yang lain adalah masing-masing komponen pada *Particle Swarm Optimization (PSO)* ada kaitannya pada suatu kecepatan. Komponen-komponen terbilang berpindah dengan pencarian ruang dengan kecerdasan yang dinamis berlandaskan perilaku historisnya. Maka sebab itu, kompone-komponen memiliki kecondongan untuk berpindah ke ruang pencarian yang lebih baik sesudah melalaui proses pencarian sebelumnya. (Sevkli dan Guner, 2006)

Kesederhanaan algoritma dan performansinya yang baik, menghasilkan *Particle Swarm Optimization* (PSO) sudah banyak menarik peminatnya di dunia para peneliti dan juga sudah diaplikasikan dalam beberapa persoalan optimisasi. *Particle Swarm Optimization* (PSO) juga sudah marak sebagai teknik optimasi global dengan alasan beberapa permasalahan besar dapat diselesaikan dengan baik. (Sevkli dan Guner, 2006)

Ada beberapa istilah yang difungsikan dalam *Particle Swarm Optimization* (PSO) bisa diartikan sebagai berikut:

1. Kawanan/swarm adalah komunitas atau populasi atau kumpulan komponen.

2. Komponen adalah elemen atas satu kawanan/*swarm*. Masing-masing komponen mencerminkan satu jalan keluar yang mungkin atas persoalan yang sudah diselesaikan. Tempat awal satu komponen merupakan representasi jalan keluar saat itu. Jumlah komponen menunjukkan ukuran kawanan/swarm dan dinotasikan sebagai sw_size.
3. Personal best adalah posisi terbaik pada tiap partikel dalam tiap iterasi yang dipersiapkan guna memperoleh jalan keluar yang teroptimum.
4. *Global best* adalah tempat teroptimum komponen atas kawanan/*swarm*. Global best juga bisa didapatkan dengan mengambil tempat terbaik dari Personal best.
5. Vektor kecepatan (Velocity) adalah vektor yang membangkitkan cara optimisasi yang memutuskan arah dimana suatu komponen dibutuhkan untuk bergeser guna membenahi ke tempatnya semula.
6. Inertia weight (w) adalah tolak ukur yang difungsikan sebagai memriksa akibat dari adanya *velocity* yang dikasihkan untuk suatu komponen.

3.7.1 Pencarian Jarak Terdekat antara NPC Hunter dengan NPC Hewan

Particle Swarm Optimizatin (PSO) mencontohkan perilaku sekelompok burung. Seperti contoh: Kawana seekor burung yang selaku *random* sedang berburu makanan pada satu kawasan/daerah. Namun cuma ada satu butir makana saja pada tempat yang dijelajahi. Kawanan burung tidak mengetahui makanan ini berada di mana. Akan tetapi kawanan burung tersebut tahu berapa banyaknya makanan di setipa iterasi. Rencana teroptimum agar bisa menhasilkan makanan ialah membuntuti kawanan burung yang tedekat dengna titil/sumber makanan yang dicari.

Particle Swarm Optimization (PSO) mencontoh dari jalan cerita dan dipergunakan buat mengatasi suatu problem optimasi. Pada *Particle Swarm Optimization (PSO),* masing-masing jalan keluar satu-satunya membuat "sebentuk burung" pada bagian pelacakan/pencarian. Saya mengartikan "partikel atau komponen". Seluruh partikel tau komponen mempunyai jumlah fitness yang nilai sama fungsi fitness wajib dimaksimalkan dan mempunyai kecerdasan perjalanan langsung pada partikel atau komponen. Seluruh partikel melayang melewati ruang problem serta mencontoh partikel atau komponen yang maksimal saat ini

Algoritma *Particle Swarm Optimizatin* (PSO) memakai dua variabel acak r1 dan r2 yang dua-duanya menciptakan nilai *random* serta jangka sekitar 0 dan 1. Variabel acak ini difungsikan guna memberikan pengaruh pada sifat *stochastic* dari algoritma *Particle Swarm Optimization (PSO).* Angka pada r1 dan r2 disamakan pada konstanta c1 dan c2 yang mempunyai jarak angka antara $0 < c1, c2 \leq 2$. Konstanta terbilamg semacam koefisien kecepatan yang menulari renggang paling besar yang bisa diterima untuk sebuah partikel atau komponen pada sebuah perulangan.

3.7.2 Hasil Perhitungan Manual

Particle swarm optimization (PSO), yang meniru karakter sekelompok burung atau ikan. Algoritma *PSO* meniru perilaku sosial organisme ini. Perilaku sosial terdiri dari aksi individu dan pengaruh dari individu-individu lain dalam suatu kelompok. Kata partikel membuktikan, misalnya, seekor burung dalam kawanan burung. Pada masing-masing individu atau partikel berkarakter memakai kecerdasannya (*intelligence*) sendiri dan juga diakibatkan tingkah laku gabungan

kolektifnya. Maka dari itu, apabila salah satu dari partikel atau satu burung mendapatkan jalan yang terbaik atau jalan yang tercepat untuk menuju ke sumber makanan, maka burung yang lain atau kelompok yang lain juga akan menemukan jalur tebaik atau tercepat tadi, meskipun lokasinya mereka jauh di kelompok tadi..

Misalkan kita mempunyai fungsi berikut.

Kita mempunyai persamaan pertikel matematika seperti berikut

Table 3.3 Partikel Mematikan

F[partikel, iterasi]	Posisi	Nilai Partikel
$F_{[1,0]}$	-4	F(4) = 21
$F_{[2,0]}$	-3	11
$F_{[3,0]}$	2	21
$F_{[4,0]}$	5	75

Kemudian nilai awal kecepatan/*velocity* nya

Table 3.4 Nilai Awal Kecepatan

	Velocity
$V_{[1,0]}$	0
$V_{[2,0]}$	0
$V_{[3,0]}$	0
$V_{[4,0]}$	0

Kemudian *Pbest* dan *Gbest* nya sebagai berikut

Table 3.5 *Pbest Gbest*

Pbest			Gbest	
Posisi	Nilai Partikel		Posisi	Nilai Partikel
-4	F(4) = 21		-3	11
-3	11			

2	**21**			
5	**75**			

Akan dilakukan iterasi sebanyak 5 kali, sehingga didapatkan rho(i) untuk masing-masing iterasi sebagai berikut

Table 3.6 Iterasi

Jumlah iterasi	**5**
Rhomax	**0.9**
Rhomin	**0.1**

Table 3.7 Iterasi

Iterasi ke (i)	**rho**
1	**0.74**
2	**0.58**
3	**0.42**
4	**0.26**
5	**0.1**

Tiap iterasi akan dilakukan update berupa V dan Posisi dengan aturan berikut

$V_{[partikel,i]} = rho_{[i]}*V_{[i-1]}+c1*r1*(Pbest_{[partikel,i]}-posisi_{[partikel]})+c2*r2(Best-posisi_{[partikel]})$

Dengan update posisi berikut

Posisi[partikel_baru,i] = Posisi[partikel_lama,i-1] + V[partikel,i]

Misalkan pada iterasi ke i=1

Hitung kecepatan/*velocity* **baru**

$V[1,1] = rho_{[1]}*V_{[0]}+c1*r1*(Pbest_{[1]}-posisi_{[1]})+c2*r2(Best-posisi_{[1]})$

0.74 * 0.0 + (1.0 * 0.1 * (-4.0 - -4.0)) + (1.0 * 0.2 * (-3.0 - -4.0)) = 0.2

$V[2,1] = rho_{[1]} * V_{[0]} + c1 * r1 * (Pbest_{[2]} - posisi_{[2]}) + c2 * r2 (Best - posisi_{[2]})$

0.74 * 0.0 + (1.0 * 0.1 * (-3.0 - -3.0)) + (1.0 * 0.2 * (-3.0 - -3.0)) = 0

$V[3,1] = rho_{[1]} * V_{[0]} + c1 * r1 * (Pbest_{[3]} - posisi_{[3]}) + c2 * r2 (Best - posisi_{[3]})$

0.74 * 0.0 + (1.0 * 0.1 * (2.0 - 2.0)) + (1.0 * 0.2 * (-3.0 - 2.0)) = -1

Dan seterus nya…

V[4,1] = 0.74 * 0.0 + (1.0 * 0.1 * (5.0 - 5.0)) + (1.0 * 0.2 * (-3.0 - 5.0)) = -1.6

Kemudian update Posisi **Baru**

Posisi[1,1] = Posisi[1,0] + V[1,1] = -4.0 + 0.2 = -3.8

Posisi[2,1] = Posisi[2,0] + V[2,1] = -3.0 + 0.0 = -3.0

Posisi[3,1] = Posisi[3,0] + V[3,1] = 2.0 + -1.0 = 1.0

Posisi[4,1] = Posisi[4,0] + V[4,1] = 5.0 + -1.6 = 3.4

Kemudian kita akan mencari nilai *Pbest* dengan cara compare partikel lama dan partikel baru

Ingat yang dibandingkan bukan posisi tapi nilai partikel nya

Table 3.8 Nilai Partikel

Fungsi Partikel	**2*x^2=4*x+5**				
	Partikel Lama			**Partikel Baru**	
Partikel ke	**posisi**	**Nilai**		**Posisi**	**nilai**
1	**-4**	**21**		**-3.8**	**18.68**

2	**-3**	**11**		**-3**	**11**
3	**2**	**21**		**1**	**11**
4	**5**	**75**		**3.4**	**41.72**

Sehingga menjadi berikut

Table 3.9 Hasil Pbest

Pbest	
Posisi	**Nilai**
-3.8	**18.68**
-3	**11**
1	**11**
3.4	**41.72**

Sedangkan untuk mencari *Gbest* sebagai berikut, yaitu nilai partikel terkecil dari *Pbest*, bukan posisi

Table 3.10 Mencari Nilai Gbest

Pbest			**Gbest**	
Posisi	**Nilai**		**posisi**	**nilai**
-3.8	**18.68**		**-3**	**11**
-3	**11**			
1	**11**			
3.4	**41.72**			

Hal yang penting dari *Pbest* dan *Gbest* adalah posisi, sedangkan nilai I hanya untuk pembanding saja

Misalkan pada iterasi ke i=2

Hitung kecepatan/*velocity* **baru**

V[1,2] = 0.58 * 0.2 + (1.0 * 0.1 * (-3.8 - -3.8)) + (1.0 * 0.2 * (-3.0 - -3.8))

= 0.276

V[2,2] = 0.58 * 0.0 + (1.0 * 0.1 * (-3.0 - -3.0)) + (1.0 * 0.2 * (-3.0 - -3.0))

= 0.0

V[3,2] = 0.58 * -1.0 + (1.0 * 0.1 * (1.0 - 1.0)) + (1.0 * 0.2 * (-3.0 - 1.0))

= -1.38

V[4,2] = 0.58 * -1.6 + (1.0 * 0.1 * (3.4 - 3.4)) + (1.0 * 0.2 * (-3.0 - 3.4))

= -2.208

Kemudian update Posisi **Baru**

Posisi[1,2] = Posisi[1,1] + V[1,2] = -3.8 + 0.276 = -3.524

Posisi[2,2] = Posisi[2,1] + V[2,2] = -3.0 + 0.0 = -3.0

Posisi[3,2] = Posisi[3,1] + V[3,2] = 1.0 + -1.38 = -0.38

Posisi[4,2] = Posisi[4,1] + V[4,2] = 3.4 + -2.208 = 1.192

Table 3.11 Nilai Partikel

Fungsi partikel	**2*x^2+4*x+5**				
	Partikel Lama			**Partikel Baru**	
Partikel ke	**Posisi**	**Nilai**		**Posisi**	**Nilai**
1	**-3.8**	**18.68**		**-3.524**	**15.7412**
2	**-3**	**11**		**-3**	**11**
3	**1**	**11**		**-0.38**	**3.7688**
4	**3.4**	**41.72**		**1.192**	**12.6097**

		Pbest		Gbest	
		Posisi	**Nilai**	**Posisi**	**Nilai**
		-3.52	**15.7412**	**-3.8**	**3.7688**
		-3	**11**		
		-0.38	**3.7688**		
		1.192	**12.6097**		

Misalkan pada iterasi ke i=3

Hitung kecepatan/*velocity* **baru**

V[1,3] = 0.42 * 0.276 + (1.0 * 0.1 * (-3.524 - -3.524)) + (1.0 * 0.2 * (-0.38 - -3.524)) = 0.744

V[2,3] = 0.42 * 0.0 + (1.0 * 0.1 * (-3.0 - -3.0)) + (1.0 * 0.2 * (-0.38 - -3.0)) = 0.524

V[3,3] = 0.42 * -1.38 + (1.0 * 0.1 * (-0.38 - -0.38)) + (1.0 * 0.2 * (-0.38 - -0.38)) = -0.579

V[4,3] = 0.42 * -2.208 + (1.0 * 0.1 * (1.192 - 1.192)) + (1.0 * 0.2 * (-0.38 - 1.192)) = -1.241

Kemudian update Posisi **Baru**

Posisi[1,3] = Posisi[1,2] + V[1,3] = -3.524 + 0.744 = -2.78

Posisi[2,3] = Posisi[2,2] + V[2,3] = -3.0 + 0.524 = -2.476

Posisi[3,3] = Posisi[3,2] + V[3,3] = -0.38 + -0.579= -0.959

Posisi[4,3] = Posisi[4,2] + V[4,3] = 1.192 + -1.241 = -0.049

Table 3.12 Nilai Partikel

Funsi Partikel	**2*x^2+4*x+5**					

	Partikel Lama			**Partikel Baru**		
Partikel ke	**Posisi**	**Nilai**		**Posisi**	**Nilai**	
1	**-3.524**	**15.74**		**-2.78**	**9.3368**	
2	**-3**	**11**		**-2.476**	**7.35715**	
3	-0.38	3.769		-0.959	3.00336	
4	1.192	12.61		-0.049	4.8088	
		Pbest			**Gbest**	
		Posisi	**Nilai**		**Posisi**	**Nilai**
		-2.78	**9.3368**		**-0.959**	**3.00336**
		-2.48	**7.35715**			
		-0.96	**3.00336**			
		-0.05	**4.8088**			

Berikut jika kita menggunakan jumlah iterasi maksimal = 50, akan menghasilkan berikut

Table 3.13 Hasil Pbest dan Gbest

Pbest			**Gbest**	
Posisi	**Nilai**		**Posisi**	**Nilai**
-1.00210285	**3.00000884**		**-099943127**	**3**
-1.00317228	**3.00002013**			
-0.99943127	**3.00000065**			
-1.00171801	**3.0000059**			

BAB IV

IMPLEMENTASI DAN PEMBAHASAN

4.1 Implementasi

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

4.1.1 Kebutuhan Perangkat Keras Untuk Uji Coba

Perangkat keras yang diperlukan untuk uji coba aplikasi *game* ini adalah sebagai berikut:

Tabel 4.1 Kebutuhan Perangkat Keras Untuk Uji Coba

No.	Perangkat Keras	Spesifikasi
1.	*Processor*	Intel(R) Core(TM) i3-2310M CPU @ 2.10 GHz
2.	*RAM*	4 Gb
3.	*VGA*	AMD(R) HD Graphics 3000
4.	*HDD*	500 Gb
5.	*Monitor*	14’
6.	*Speaker*	On
7.	*Mouse & Keyboard*	On

4.1.2 Kebutuhan Perangkat Lunak

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

Tabel 4.2 Kebutuhan Perangkat Lunak

No.	Perangkat Lunak	Spesifikasi
1.	Sistem Operasi	*Windows 10 64 Bit*
2.	*GameEngine*	*Unity 5.6.0f3 (64-bit)*
3.	Konsep desain 2D	*Moockup*
4.	Desain 3D	*Blender 2.7*
5.	*Script Writer*	*Mono Develop*dan *Notepad++*

4.2 Implementasi Algoritma *Particle Swarm Optimization* pada *game*

Proses implementasi algoritma *Particle Swarm Optimization* adalah penerapan sebuah sistem yang telah dibangun berdasarkan rancangan yang diususlkan. Pada tahap ini dilakukan implementasi kecerdasan buatan pada NPC Hunter dengan menggunakan algoritma *Particle Swarm Optimization.*

Berikut ini akan dijelaskan penggunaan *method* dan fungsi pada table

Tabel 4.3 Keterangan *class* Algoritma *Particle Swarm Optimization*

No	Method/Fungsi	Keterangan

1	`function ApplyDamage (damage : float) {`
	`if (hitPoints <= 0.0)`
	`return;`
	`hitPoints -= damage;`
	`if (hitPoints <= 0.0)`
	`{`
	`Detonate();`
	`}`
		Method yang digunakan untuk mengatur status (kesehatan) pada HUNTER

	}	

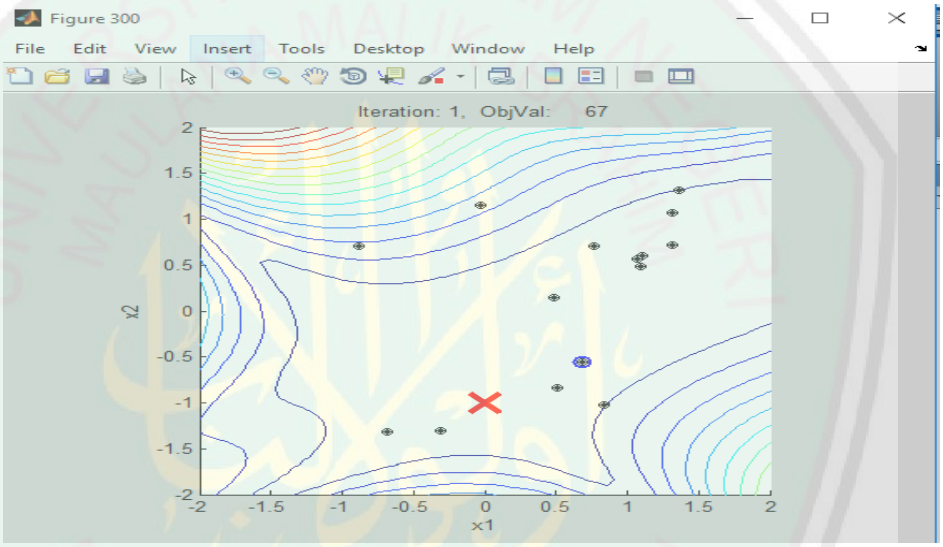
2	public bool UpdateRandomRange(float _random_range)
	{
	if(RandomRange != _random_range)
	{
	RandomRange = _random_range;
	return true;
	}
	else
	RandomRange = _random_range;
	return false;
	}
		Method ini berfungsi untuk mengatur jarak (range) pada spawn area HUNTER

3	public float MovePositionDistance{
	get{ return (Owner == null ||
	CurrentMove == null ? 0 :
	PositionTools.GetDistance(MovePosition,
	Owner.transform.position,
	DesiredIgnoreLevelDifference)); }
	}
	public float TargetMovePositionDistance{
	get{ return (Owner == null ||
	CurrentTarget == null ? 0 :
	CurrentTarget.TargetMovePositionDistance
	To(Owner.transform.position)); }
	}
	protected float DetourPositionDistance{
	get{ return (Owner == null ||
	CurrentMove == null ||
	CurrentMove.Detour.Position ==
	Vector3.zero ? 0 :
	PositionTools.GetDistance(
	CurrentMove.Detour.Position,
	Owner.transform.position,
	CurrentMove.IgnoreLevelDifference)); }
	}
		Method ini berfungsi untuk mengatur perpindahan HUNTER menuju target dengan jarak tertentu

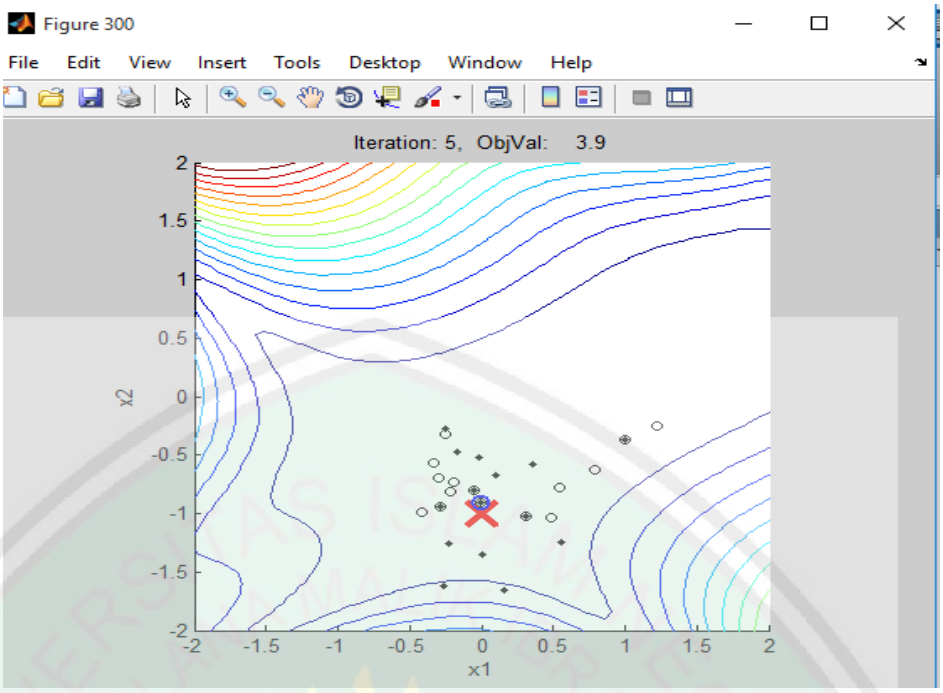
4	function Update() {
	if (minimapCamera == null) return;
	adjustSize =
	Mathf.RoundToInt(Screen.width/10);
	minimapCamera.pixelRect = new
	Rect(offsetX, (Screen.height -
	(minimapSize * adjustSize)) - offsetY,
	minimapSize * adjustSize, minimapSize *
	adjustSize);
		Method untuk menampilkan mini map

4.3 Pengujian Game

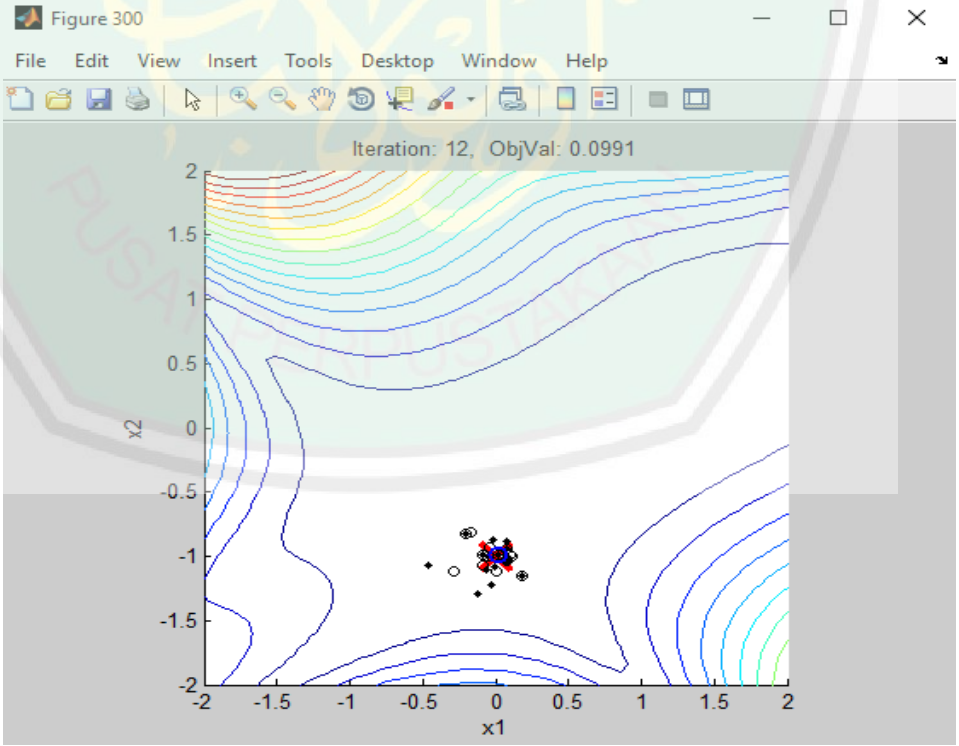
Pengujian ini dilakukan untuk mengetahui NPC Hunter melakukan pergerakan secara bersama menuju target yaitu NPC Hewan. Pengujian ini dilakukan dengan menggunakan lebih dari 1 NPC Hunter, dan sebagai perbandingan 1 NPC Hunter menggunakan algoritma *A-Star,* dan seluruh NPC Hunter menyebar di hutan dan setiap NPC Hunter menggunakan algoritma *Particle Swarm Optimization* dan bergerak secara bersama menuju target yaitu NPC Hewan.



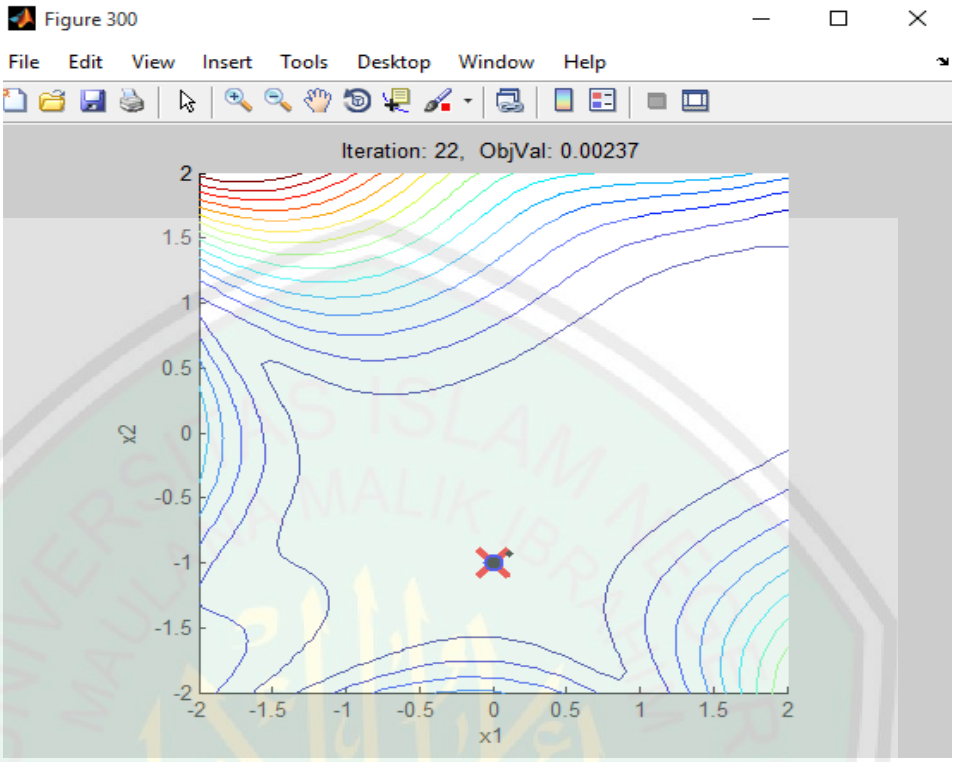
Gambar 4.1 Pergerakan *Particle Swarm Optimization* (a)



Gambar 4.2 Pergerakan *Particle Swarm Optimization* (b)

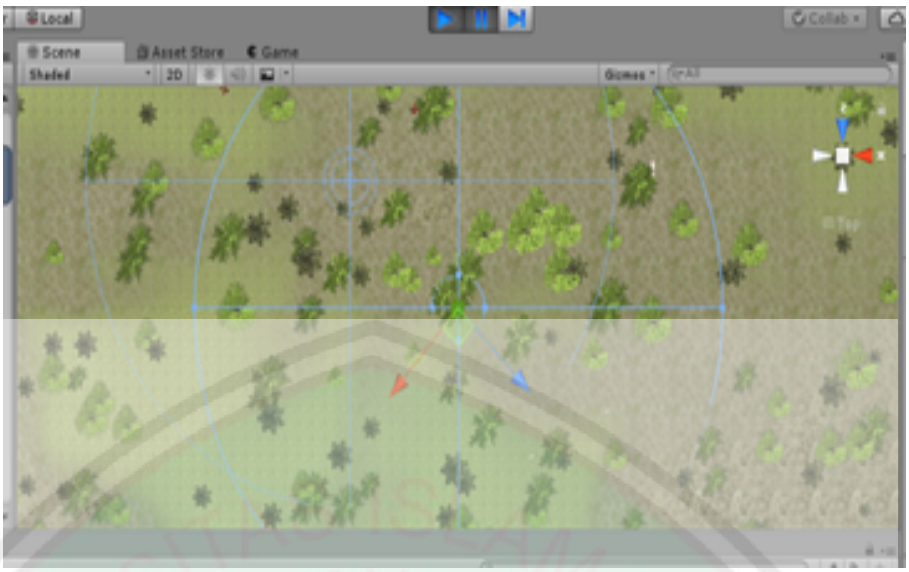


Gambar 4.3 Pergerakan *Particle Swarm Optimization* (c)

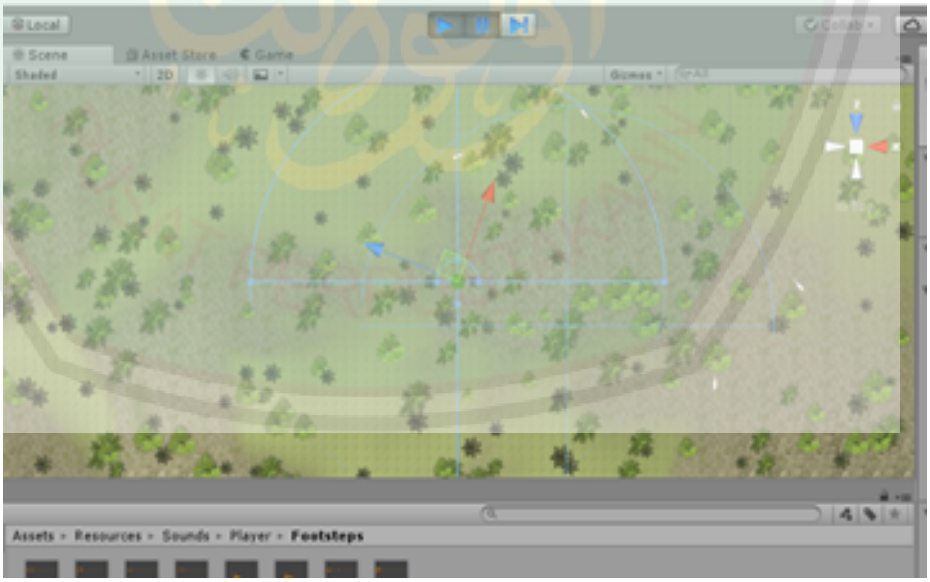


Gambar 4.4 Pergerakan *Particle Swarm Optimization* (d)

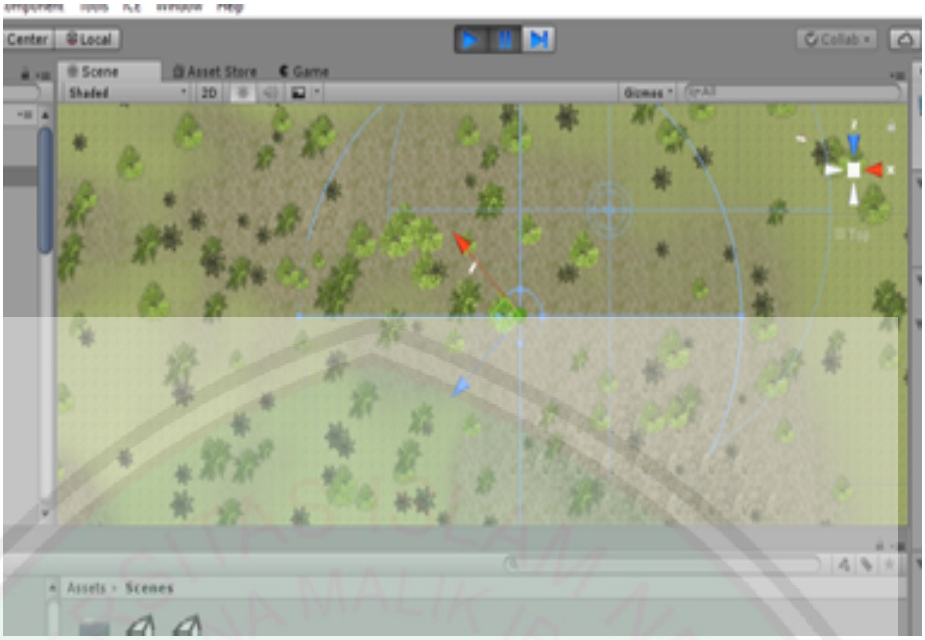
Pada **gambar 4.1** sampai **gambar 4.4** menggambarkan hasil dari perhitungan algoritma *Particle Swarm Optimization* menggunakn matlab dan kemudain digambarkan pada *game* 3D Hunter Of Forest, dalam percobaan ini memberikan gambaran bahwa *Particle Swarm Optimization* bergerak secara bersamaan menuju target yang akan dituju yaitu NPC Hewan.



Gambar 4.5 Posisi Awal NPC Hunter dan NPC Hewan

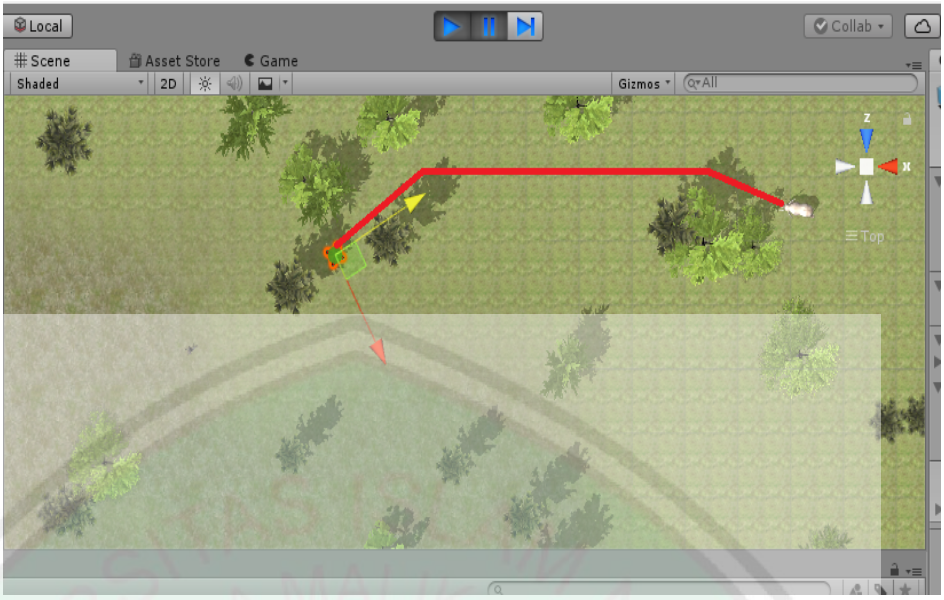


Gambar 4.6 Pergerakan NPC Hunter Menuju NPC Hewan

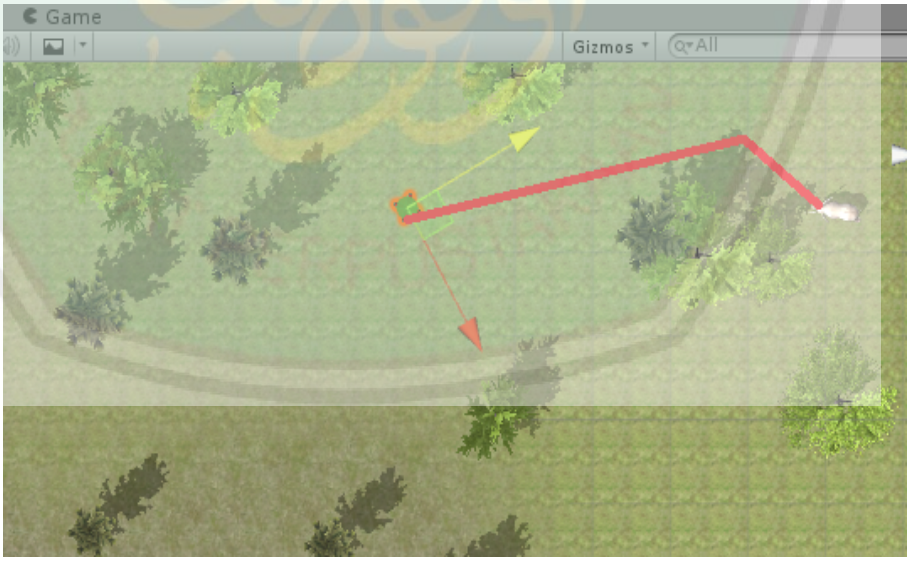


Gambar 4.7 NPC Hunter Sampai di NPC Hewan

Gambar 4.8 Posisi awal NPC *A-Star* dan NPC Hewan



Gambar 4.9 NPC *A-Star* Menuju ke Arah NPC Hewan (a)



Gambar 4.10 NPC *A-Star* Menuju ke Arah NPC Hewan (b)

Gambar 4.11 NPC *A-Star* Sampai di NPC Hewan

Dengan menerapkan algoritma *Particle Swarm Optimization* diharapkan NPC Hunter dapat bergerak secara bersamaan seperti perilaku sekawanan burung atau ikan, menuju target yaitu NPC Hewan. Pengujian ini dilakukan untuk mengetahui algoritma *Particle Swarm Optimization* telah berfungsi, sehingga bisa mengetahui perilaku atau pergerakan secara bersamaan NPC Hunter.

NPC Hunter dapat mengejar NPC Hewan seperti pada **gambar 4.6** terlihat NPC Hunter bergerak menuju NPC Hewan. Hal ini menunjukkan bahwa setiap iterasinya NPC Hunter melakukan pergerakan secara bersamaan menuju posisi yang optimal atau target yaitu NPC Hewan.

Sebagai pembanding maka diterapkan algoritma *A-Star* pada salah satu NPC Hunter, yang bertujuan untuk mengetahui tingkat efisiensi dari algortima *Particle Swarm Optimization.* Seperti pada **gambar 4.8** sampai **gambar 4.11** yang menggambrakn algoritma *A-Star* pada NPC Hunter dari posisi awal hingga bisa menuju target yaitu NPC Hewan.

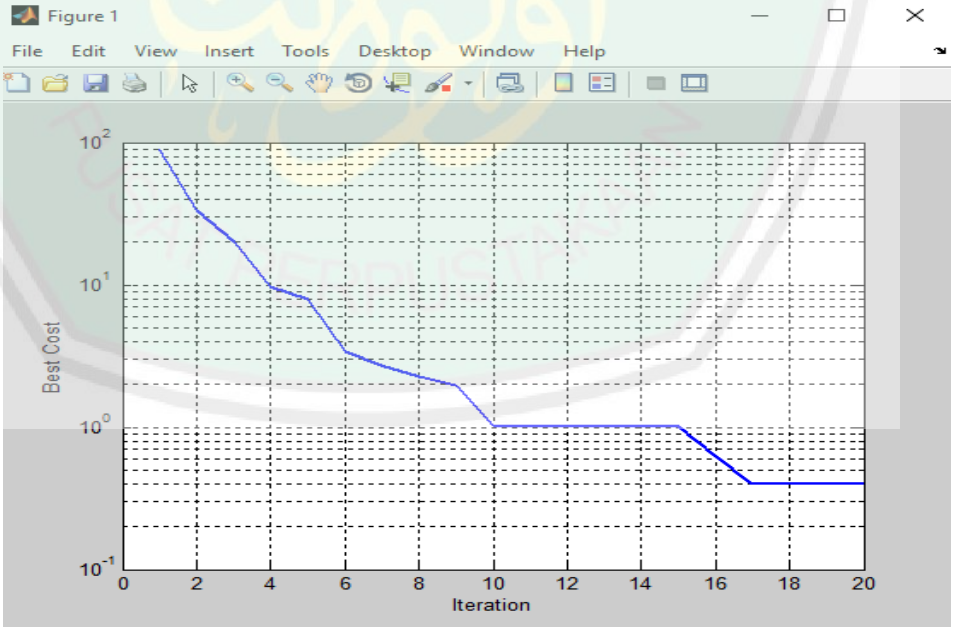
Tabel 4.4 Hasil Pengujian Pergerakan NPC Hunter dengan *Particle Swarm Optimization*

No	Iteration	Best Cost
1	1	75.3749
2	2	36.4166
3	3	16.2423
4	4	7.8127
5	5	7.1047
6	6	3.2536
7	7	3.2536
8	8	3.2536
9	9	3.2536
10	10	1.8697
11	11	1.8697
12	12	1.8697
13	13	1.3814
14	14	1.3814

15	15	1.3814
16	16	1.1405
17	17	1.1405
18	18	1.1405
19	19	0.9747
20	20	0.9747

Pada **Tabel 4.4** dapat disimpulkan bahwa nilai rata-rata dari hasil pengujian adalah 8.5549%, dimana nilai keberhasilan terendah adalah 75,3749%, dan nilai keberhasilan tertinggi adalah 0.9747%.

Tabel 4.5 Grafik Pergerakan *Particle Swarm Optimization*



Tabel 4.6 Perbandingan Tingkat Akurasi

No.	Pengujian	Tingkat Akurasi *Particle Swarm Optimization* (%)	Tingkat Akurasi Astar (%)
1	Pengujian ke 1	24.6251	50.235
2	Pengujian ke 2	63.5834	55.834
3	Pengujian ke 3	83.7577	63.577
4	Pengujian ke 4	92.1873	65.873
5	Pengujian ke 5	92.8953	72.953
6	Pengujian ke 6	96.7464	76.764
7	Pengujian ke 7	96.7464	76.764
8	Pengujian ke 8	96.7464	76.764
9	Pengujian ke 9	96.7464	76.764
10	Pengujian ke 10	98.1303	78.103
11	Pengujian ke 11	98.1303	78.103
12	Pengujian ke 12	98.1303	78.103
13	Pengujian ke 13	98.6186	88.618
14	Pengujian ke 14	98.6186	88.618
15	Pengujian ke 15	98.6186	88.618
16	Pengujian ke 16	98.8595	95.958
17	Pengujian ke 17	98.8595	95.958
18	Pengujian ke 18	98.8595	95.958
19	Pengujian ke 19	99.0253	97.350
20	Pengujian ke 20	99.0253	97.350

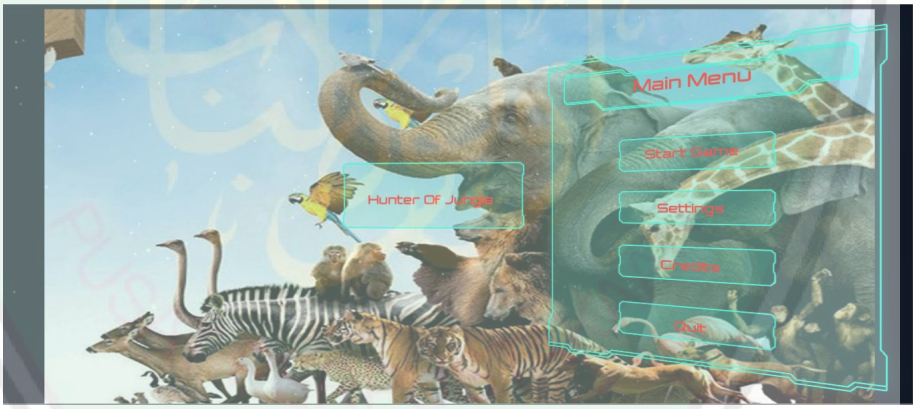
Pada **Tabel 4.6** menunjukkan perbandingan tingkat akurasi antara PSO dan A*, dimana PSO memiliki tingkat akurasi lebih baik yaitu 99.0253 % jika dibandingkan dengan tingkat akurasi pada A* yaitu 97.350%.

Pada **tabel 4.4** merupakn hasil dari perhitunan algoritma *Particle Swarm Optimization,* dari percobaan tersebut menggunakan 20 iterasi, dimana iterasi yang pertama hasil *Best Cost* nya kurang akurat,dan pada iterasi yang ke 20 hasilnya akurat. Seperti pada **tabel 4.5** menunjukan grafik tingkat akurasi dari masing-masing iterasi.

4.4 Implementasi *Game*

Berikut adalah tampilan *game* yang telah selesai dibuat

4.4.1 Tampilan Menu Awal



Gambar 4.12 Tampilan Menu Awal

Tampilan menu awal merupakan tampilan yang pertama kali akan muncul ketika *game* dijalankan. Pada tampilan menu awal terdapat 4 tombol yaitu *Start Game, Setting, Credit, Quit.*

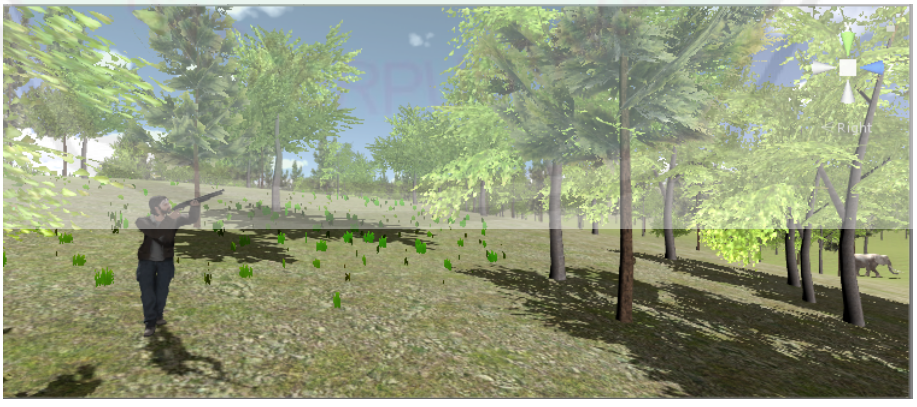
4.4.2 Tampilan *Game* Pada Bagian Awal

Tampilan *game* pada bagian awal merupakan tampilan awal ketika *game* ini berjalan. *NPC* hewan tersebar pada posisi yang telah ditentukan di sekitar area hutan belantara.

Gambar 4.13 *Tampilan game pada bagian awal (Player)*

4.4.3 Tampilan *Hunter* Pada Saat Berburu

Tampilan *hunter* pada saat berburu merupakan tampilan ketika *NPC hunter* mendapatkan *output* perilaku berburu.



Gambar 4.14 *NPC hunter* berburu

4.4.4 Tampilan *Hunter* Pada Saat Menyerang

Tampilan *hunter* pada saat berburu merupakan tampilan ketika *NPC hunter* mendapatkan *output* perilaku menyerang.

Gambar 4.15 *NPC hunter* menyerang

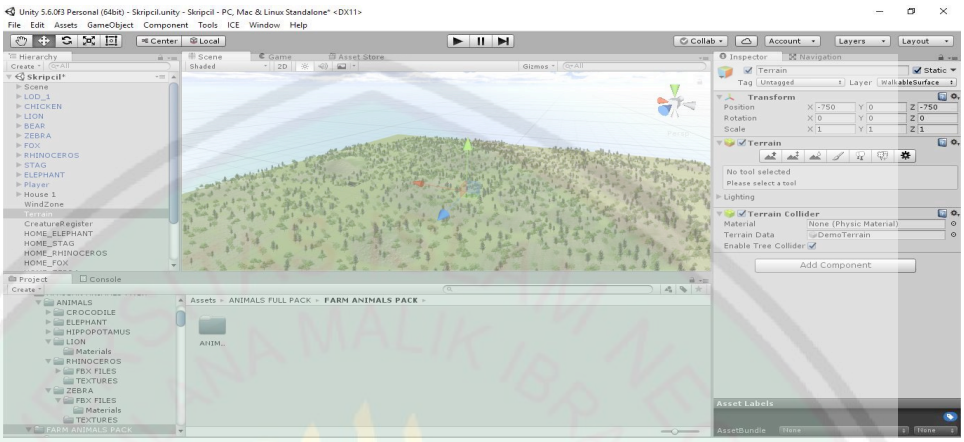
4.4.5 Tampilan *Hunter* Pada Saat Kabur

Tampilan *hunter* pada saat berburu merupakan tampilan ketika *NPC hunter* mendapatkan *output* perilaku kabur.

Gambar 4.16 *NPC* hunter kabur

4.4.6 Pembangunan *Environment*

Pembangunan *Enviroment* merupakan tampilan *editor unity* pada saat membuat *Enviroment* Hutan Belantara.



Gambar 4.17 Tampilan Pembangunan *Enviroment*

4.5 Integrasi dengan Islam

Islam merupakan agama yang sesuai dengan fitrah manusia, syariatnya bukan saja mendorong manusia untuk mempelajari sains dan teknologi, kemudian membangun peradaban, bahkan mengatur umatnya agar selamat dan menyelamatkan baik di dunia maupun di akhirat kelak. Semua aktifitas termasuk mengkaji dan mengembangkan sains dan teknologi dapat bernilai ibadah bahkan menjadi nilai perjuangan di sisi Allah SWT. Keduanya mempunyai wilayah masing-masing, terpisah antara satu dan lainnya, baik dari segi objek formal-material, metode penelitian maupun kriteria kebenaran.

Dalam agama Islam, semua perkara sudah terdapat dalam Al-Qur'an dan diperjelas dalam hadist atau segala perkataaan atau tingkah laku dari Nabi Muhammad Saw. Islam juga mengajarkan tentang menjaga kelestarian alam dari

kerusakan baik di darat maupun di laut yang sudah lebih dahulu mengajarkan kepada umatnya. Selain itu, Al-Quran juga ikut membicarakan tentang kerusakan di darat dan di laut, sebagai bukti bahwa Allah sangat menganjurkan hambanya untuk menjaga alam yang telah disediakan untuk manusia. Seperti peringatan dalam Al Quran Surat Al A'raf [7], ayat 56, Allah SWT berfirman :

وَلَا تُفْسِدُوافِي الْأَرْضِ بَعْدَ إِصْلَاحِهَا وَادْعُوهُ خَوْفًا وَطَمَعًا ۚ إِنَّ رَحْمَتَ اللَّهِ قَرِيبٌ مِنَ الْمُحْسِنِينَ

"Dan janganlah kamu membuat kerusakan di muka bumi, sesudah (Allah) memperbaikinya dan berdoalah kepadanya dengan rasa takut (tidak akan diterima) dan harapan (akan dikabulkan). Sesungguhnya rahmat Allah amat dekat kepada orang orang yang berbuat baik."(Q.S. Al-A'raf[7]: 56)

Dalam ayat ini menjelaskan bahwa bahwa Allah SWT, memerintahkan manusia untuk tidak membuat kerusakan di muka bumi ini setelah Allah menciptakan alam ini dengan sempurna, penuh harmoni, serasi dan sangat seimbang untuk mencukupi kebutuhan makhluk-Nya.

Firman Allah dalam Al Quran Surat Al-An'am [6], ayat 38:

وَمَا مِن دَآبَّةٍ فِي الأَرْضِ وَلاَ طَائِرٍ يَطِيرُ بِجَنَاحَيْهِ إِلاَّ أُمَمٌ أَمْثَالُكُم مَّا فَرَّطْنَا فِي الكِتَابِ مِن شَيْءٍ ثُمَّ إِلَى رَبِّهِمْ يُحْشَرُونَ

"Dan tiadalah binatang-binatang yang ada di bumi dan burung-burung yang terbang dengan kedua sayapnya, melainkan umat (juga) seperti kamu. Tiadalah kami lalaikan sesuatu pun dalam Al-Kitab, kemudian kepada Allah mereka dihimpunkan."(Q.S Al-An'am[6]: 38)

Dalam ayat ini menjelaskan bahwa Allah SWT, semua mahluk yang ada di muka bumi baik mereka binatang yang terbang atau binatang yang berjalan di

darat, melainkan mereka itu juga merupakan umat seperti kita manusia. Kepada Allahlah semua kelak dihimpun atau dikembalikan.

Pada penjelasan ayat di atas dapat di ambil kesimpulan bahwasanya kita sebagai makhluk hidup harus saling menyayangi, bukan menyayangi sesame saja tetapi juga menyayangi semua makhluk yang diciptakan oleh Allah SWT, dan kita juga akan disayangi oleh Allah SWT dan malaikatnya. Maka dari itu kita harus menjaga kelastarian alam agar tetap bertahan sampai ke anak cucu kita.

BAB V

KESIMPULAN DAN SARAN

5.1 Kesimpulan

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

1. Penelitian ini berhasil mengimplementasikan Algoritma *Particle Swarm Optimizaion* untuk pencarian NPC Hewan *Hunter Of Forest.* Pada hal ini dilakukan sebanyak 20 kali pengujian, dengan nilai rata-rata 8.5549%, dimana nilai keberhasilan terendah adalah 75.3749%, dan nilai keberhasilan tertinggi adalah 0.9747%.

5.2 Saran

Dalam penelitian pembuatan *game* ini masih banyak yang nantinya perlu untuk dilakukan pengembangannya, antara lain:

1. Menambah macam-macam satwa langka lagi yang lebih banyak untuk memberikan pengetahuan tentang macam-macam satwa langka di Indonesia.
2. *Game* ini diperlukan penambahan animasi yang lebih menarik lagi, sehingga kesannya bisa menjadi lebih nyata bagi pengguna.
3. Pengembangan *game* dengan *level* permainan yang lebih banyak dan menantang.

DAFTAR PUSKATA

Hurd, Daniel dan Jenuings, Erin. 2009. *Standardized Educational Games Ratings: Suggested Criteria*. Karya Tulis Ilmiah.

Suindarti, *Game Edukasi Meningkatkan Daya Ingat Anak "Bermain Bersama Dido*" Dengan Macromedia Director, AMIKOM, Yogyakarta, 2011.

Bratton, D. and Kennedy, J., 2006. Defining a Standard for Particle Swarm Optimization, Proceedings of IEEE Swarm Intelligence Symposium. 1-4244-0708-7, IEEE.

Iwan Setiawan,ST., MT. 2006. *Perancangan Software Embedded System Berbasis FSM.*

J. Kennedy, R.C. Eberhart, "Particle swarm optimization," IEEE International Conference on Neural Networks, IEEE Service Center, Piscataway, NJ, 1942-1948.

Reynolds, C. W. (1999). Steering Behaviors For Autonomous Characters. *Game Developers Conference*, (pp. 763-782).

Eberhart,R.C., and Kennedy, J. A new optimizer using particle swarm theory. Proceedings of the sixth International Symposium on Micro Machine and Human Science, Nagoya, Japan, 39-43. Piscataway, NJ:IEEE Service Center.

J. Kennedy and R. C. Eberhart. *Particle swarm optimization.* In Proceedings of the 1995 *IEEE International Conference on Neural Network. IEEE Service Center,* Piscataway, 1995.

Ardiana Rosita, Yudhi Pyrwanto, dan Rully Soelaiman. Implementasi Algoritma *Particle Swarm Optimization* untuk Menyelesaikan System Persamaan Nonlinear. Surabaya. 2012.

Misra Hartati, Iwan Vanany, dan Budi Santoso. Pengembangan Algoritma *Particle Swarm Optimization* untuk Optimasi Dispersi Betch pada Proses Produksi. Surabaya. 2012.

Galih Hermawan. Implementasi Algoritma *Particle Swarm Optimization* untuk Penentuan Posisi Strategis Agent pada Simulasi Robot Sepak Bola Dua Dimensi. Jakarta. 2012.

Safril Rizki Waluyo. Pencarian Jalur Terbaik Menggunakan *Particle Swarm Optimization* untuk Mengoptimasi Lalu Lintas Kendaran. Surabaya. 2010.

Syahri Mu'min, Mochammad Hariadi, dan Supeno Mardi Susiki Nugroho. Pergerakan Otonom Pasukan Berbasis *Algoritma Boids* Menggunakan Metode *Particle Swarm Optimization.* Surabaya. 2015.