IMPLEMENTASI ALGORITMA *SINGLE-OBJECTIVE* PSO (*PARTICLE SWARM OPTIMATION*), UNTUK MENENTUKAN RUTE OPTIMAL NPC PADA GAME *ANDROID*

SKRIPSI

Oleh:
BISYRI SYAMSURI
NIM. 19650135

PROGRAM STUDI TEKNIK INFORMATIKA
FAKULTAS SAINS DAN TEKNOLOGI
UNIVERSITAS ISLAM NEGERI MAULANA MALIK IBRAHIM
MALANG
2023

IMPLEMENTASI ALGORITMA *SINGLE-OBJECTIVE* PSO (*PARTICLE SWARM OPTIMATION*), UNTUK MENENTUKAN RUTE OPTIMAL NPC PADA GAME *ANDROID*

SKRIPSI

Oleh:
BISYRI SYAMSURI
NIM. 19650135

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

PROGRAM STUDI TEKNIK INFORMATIKA
FAKULTAS SAINS DAN TEKNOLOGI
UNIVERSITAS ISLAM NEGERI MAULANA MALIK IBRAHIM
MALANG
2023

HALAMAN PERSETUJUAN

IMPLEMENTASI ALGORITMA *SINGLE-OBJECTIVE* PSO (*PARTICLE SWARM OPTIMATION*), UNTUK MENENTUKAN RUTE OPTIMAL NPC PADA GAME *ANDROID*

SKRIPSI

Oleh :
BISYRI SYAMSURI
NIM. 19650135

Telah Diperiksa dan Disetujui untuk Diuji
Tanggal:

Dosen Pembimbing I	Dosen Pembimbing II
Juniardi Nur Fadila, M.T	Dr. Yunifa Miftachul Arif, M.T
NIP. 19920605 201903 1 015	NIP. 19830616 201101 1 004

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

Dr. Fachrul Kurniawan, M.MT
NIP. 19771020 200912 1 001

HALAMAN PENGESAHAN

IMPLEMENTASI ALGORITMA *SINGLE-OBJECTIVE* PSO (*PARTICLE SWARM OPTIMATION*), UNTUK MENENTUKAN RUTE OPTIMAL NPC PADA GAME *ANDROID*

SKRIPSI

Oleh :
BISYRI SYAMSURI
NIM. 19650135

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

Susunan Dewan Penguji

Ketua Penguji	:	Dr. Fressy Nugroho, M.T NIP. 19710722 201101 1 001	( )
Anggota Penguji I	:	Hani Nurhayati, M.T NIP. 19780625 200801 2 006	( )
Anggota Penguji II	:	Juniardi Nur Fadila, M.T NIP. 19920605 201903 1 015	( )
Anggota Penguji III	:	Dr. Yunifa Miftachul Arif, M.T NIP. 19830616 201101 1 004	( )

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

Dr. Fachrul Kurniawan, M.MT
NIP. 19771020 200912 1 001

PERNYATAAN KEASLIAN TULISAN

Saya yang bertanda tangan di bawah ini:

Nama	:	Bisyri Syamsuri
NIM	:	19650135
Fakultas	:	Sains dan Teknologi
Program Studi	:	Teknik Informatika
Judul Skripsi	:	Implementasi Algoritma *Single Objective* PSO (*Particle Swarm Optimation*), Untuk Menentukan Rute Optimal NPC Pada Game *Android.*

Dengan jujur Saya menyatakan bahwa Skripsi yang saya buat adalah hasil karya saya sendiri dan tidak mengandung pengambilan data, tulisan, atau pemikiran orang lain yang saya klaim sebagai milik saya, kecuali jika saya mencantumkan sumber kutipan pada daftar pustaka.

Jika dimasa yang akan datang terbukti atau dapat dibuktikan bahwa Skripsi ini merupakan hasil plagiarism, saya siap menerima sanksi atas perbuatan tersebut

.

Malang, 22 Maret 2023
Yang membuat peryataan,

Bisyri Syamsuri
NIM. 19650135

HALAMAN MOTTO

"Always Give The Best, and Get The Best"

HALAMAN PERSEMBAHAN

بِسْمِ اللَّهِ الرَّحْمَنِ الرَّحِيم

Penulis ingin mempersembahkan karya ilmiah ini kepada orang tua, keluarga, dosen, sahabat, dan semua pihak yang telah membantu secara aktif dalam menyelesaikan penelitian ini

KATA PENGANTAR

Assalamualaikum Wr. Wb

Puja dan puji syukur penulis panjatkan kehadirat Ilahi Robbi Allah Subhanahu Wa Ta'ala, Tuhan Yang Maha Esa dan Tuhan Yang Maha Agung Atas karunia Taufik dan Hidayah-Nya kepada penulis sehingga dapat penyelesaikan skripsi dengan judul "Implementasi Algoritma *Single Objective* PSO (*Particle Swarm Optimation*), Untuk Menentukan Rute Optimal NPC Pada Game *Android*" dengan baik.

Banyak pihak yang terlibat dalam penulisan skripsi ini yang telah memberikan dukungan baik moril maupun materil. Untuk itu dalam kesempatan kali ini penulis ingin mengucapkan banyak terimakasih kepada:

1. Prof. Dr. H.M. Zainuddin, MA selaku rektor Universitas Islam Negeri Maulana Malik Ibrahim.
2. Dr. Sri Hariani, M.Si selalu dekan Fakultas Sains dan Teknologi Universitas Islam Negeri Maulana Malik Ibrahim.
3. Dr. Fachrul Kurniawan, M.MT selaku Ketua Program Studi Teknik Informatika Universitas Islam Negeri Maulana Malik Ibrahim Malang yang senantiasa memberikan motivasi serta dorongan untuk menyelesaikan skripsi ini.
4. Juniardi Nur Fadila, M.T selaku Dosen Pembimbing I yang telah dengan sabar memberikan arahan, saran, kritik, serta motivasi yang baik dalam penulisan hingga program yang dibuat dalam menyelesaikan skripsi ini.

5. Dr. Yunifa Miftachul Arif, M.T selaku Dosen Pembimbing II yang telah memberikan waktunya untuk memberikan ilmu serta arahan dalam proses penyelesaian penyusunan skripsi ini.
6. Para dosen dan staf akademika Jurusan Teknik Informatika yang telah memberikan ilmu yang sangat bermanfaat, yang daripada itu secara tidak langsung telah turut serta dalam penyusunan skripsi ini.
7. Kedua orang tua saya, rama Abd. Rahman dan ibu Khairunnisa' yang telah memberikan dukungan berupa doa, motivasi, arahan, semangat, dukungan, serta suntikan dana terbesar, dalam menyelesaikan perkuliahan dan penyusunan skripsi ini. Serta adik-adik saya Helya Sani dan Misika Albina yang telah memberikan motivasi dan semangat untuk terus menyelesaikan skripsi ini.
8. Saudara – saudara persambatan yang menishabkan dirinya untuk berkumpul secara sadar dalam grub beranama "Ahlussunah Wal Jama'ah" yang beranggotakan Harizul Thoriq, Fikri Rasyiqal, Zulfan Helmi , Arya Dicky, Rizky Alfin, Fauzi Deri, Ridhuan, Maulana Krisna, Sadad Anwar, Alfaros Faiz, Kustiawan Andi, Anam Syariful, Putri Ayu, Salsabila Nadifah, Saraswati Widya dan Fahmi yang selalu saling menyemangati satu sama lain serta tak luput memberikan saran, kritik, dan informasi dalam masa perkuliahan sampai proses penyusunan skripsi ini.
9. Keluarga besar Program Studi Teknik Informatika terutama Angkatan 2019 "ALIEN" yang telah memberikan dukungan untuk saling menyelesaikan skripsi.

10. Teman-teman HIMATIF 2021 yang selalu setia dalam setiap proses saya mengembangkan diri.
11. Para pihak yang terlibat baik secara langsung maupun tidak langsung dalam proses penyusunan skripsi ini.

Penulis sadar bahwa skripsi ini masih sangat jauh dari kata sempurna dan mungkin terdapat kesalahan di dalamnya. Oleh karena itu, penulis mengharapkan kritik dan saran yang membangun untuk mengembangkan skripsi ini agar lebih bermanfaat bagi dirinya dan pembaca pada umumnya.

Malang, 22 Maret 2023

Penulis

DAFTAR ISI

DAFTAR GAMBAR

DAFTAR TABEL

ABSTRAK

Syamsuri, Bisyri, 2023. **Implementasi Algoritma *Single Objective* PSO (*Particle Swarm Optimation*), Untuk Menentukan Rute Optimal NPC Pada Game *Android***. Skripsi, Program Studi Teknik Informatika, Fakultas Sains dan Teknologi, Universitas Islam Negeri Maulana Malik Ibrahim Malang. Pembimbing: (I) Juniardi Nur Fadila, M.T. (II) Dr. Yunifa Miftachul Arif, M.T.

Kata Kunci: NPC, *Single Objective Particle Swarm Optimation*, *Bee Colony Optimation, Ant Colony Optimation,* Rute Optimal, Game.

Penentuan rute optimal NPC terhadap *player* merupakan hal yang penting dalam skenario game untuk menambah tingkat kompetitif permainan. Dalam penelitian ini akan menjelaskan tentang pengukuran tingkat keefektifan algoritma optimation dalam mencari rute optimal NPC terhadap *player.* Salah satu algoritma optimation yang diimplementasikan dalam penelitian ini adalah *Single Obejctive Particle Swarm Optimation* (PSO). Algoritma PSO merupakan sebuah algoritma yang dapat berguna untuk dalam permasalahan optimisasi yang didasarkan pada simulasi perilaku serangga dan burung yang bergerak dalam sebuah koloni, di mana setiap individu (partikel) dalam koloni mewakili sebuah solusi potensial untuk masalah yang dihadapi. Nantinya PSO akan dibandingkan dengan 2 algoritma yang berbeda yaitu *bee colony optimation* dan *ant colony optimation,* dengan mempertimbangkan 3 parameter pengujian yaitu waktu tempuh NPC menemukan rute terhadap *player*, waktu tempuh NPC menemukan rute ber-*obstacle* terhadap *player*, dan Pemilihan rute NPC yang statis (sama) atau dinamis (tidak sama). Didapatkan hasil bahwa algoritma *single objective particle swarm optimation* mendapatkan hasil paling efektif dibandingkan dengan algoritma *bee colony optimation* dan *ant colony optimation* dalam penentuan rute optimal NPC terhadap *player.*

ABSTRACT

Syamsuri, Bisyri, 2023. **Implementation of Single Objective PSO (Particle Swarm Optimization) Algorithm to Determine the Optimal NPC Route in an Android Game**. Undergraduate Thesis, Department of Informatics Engineering, Faculty of Science and Technology, Maulana Malik Ibrahim State Islamic University Malang. Advisors: (I) Juniardi Nur Fadila, M.T. (II) Dr. Yunifa Miftachul Arif, M.T.

Keywords: NPC, Single Objective Particle Swarm Optimization, Bee Colony Optimization, Ant Colony Optimization, Optimal Route, Game.

Determining the optimal route for NPCs towards players is crucial in game scenarios to increase the competitiveness of the game. This research will explain the measurement of the effectiveness level of optimization algorithms in finding the optimal route for NPCs towards players. One of the optimization algorithms implemented in this study is Single Objective Particle Swarm Optimization (PSO). The PSO algorithm is used to solve optimization problems based on simulating the behavior of insects and birds moving in a colony, where each individual (particle) in the colony represents a potential solution to the problem. PSO will be compared with two different algorithms, namely bee colony optimization and ant colony optimization, by considering three testing parameters: the time it takes for NPCs to find a route towards players, the time it takes for NPCs to find a route with obstacles towards players, and the selection of static or dynamic NPC route. The results show that the single objective particle swarm optimization algorithm is the most effective compared to bee colony optimization and ant colony optimization in determining the optimal route for NPCs towards players.

مستخلص البحث

سيمسوري، بيسيري، 2023. **تنفيذ خوارزمية PSO (Particle Swarm Optimization) الموجهة نحو هدف واحد، لتحديد الطريق الأمثل لـ NPC في لعبة .Android رسالة بحثية، قسم هندسة الحاسوب**، كلية العلوم والتكنولوجيا، جامعة الإسلامية الحكومية مولانا مالك إبراهيم مالانغ. المشرفون (I) :جونياردي نور فاديل، (II) .M.T الدكتورة يونيفا مفتاح العرف، M.T.

الكلمات الرئيسية: NPC، Single Objective Particle Swarm Optimation، Bee Colony Optimation، Ant Colony Optimation، الطريق الأمثل، لعبة.

NPC (Non Player Character) هي عنصر مهم في الألعاب. يمكن أن يلعب NPC دورًا رئيسيًا أو دورًا داعمًا أو حتى دور المعارض في اللعبة. كعدو عادةً ما يتمتع NPC بعدة مهام ، ومن بينها البحث عن اللاعب ومطاردته. يتطلب البحث عن اللاعب ومطاردته خوارزمية حتى يتمكن NPC من القيام بهذا الأمر. سيتم في هذا البحث شرح قياس مستوى فعالية خوارزمية الأمثلة في البحث عن المسار المثلى لـ NPC تجاه اللاعب. أحد خوارزميات الأمثلة التي سيتم تنفيذها في هذا البحث هو Single Objective Particle Swarm Optimization (PSO). يعد PSO خوارزمية تستخدم لحل مشاكل الأمثلة التي تعتمد على محاكاة سلوك الحشرات والطيور التي تتحرك في مستعمرة ، حيث يمثل كل فرد (جسيم) في المستعمرة حلًا محتملًا للمشكلة التي تواجهها. في نهاية المطاف ، سيتم مقارنة PSO مع خوارزميتين مختلفتين وهما خوارزمية Bee Colony Optimization وخوارزمية Ant Colony Optimization ، بناءً على ثلاثة عوامل اختبارية وهي وقت وصول NPC للعثور على المسار المؤدي إلى اللاعب ، ووقت وصول NPC للعثور على المسار المؤدي إلى اللاعب مع وجود عوائق، واختيار NPC للمسار الثابت (نفسه) أو الديناميكي (غير نفسه).

BAB I

PENDAHULUAN

1.1 Latar Belakang

Game adalah bentuk hiburan yang paling digemari oleh masyarakat saat ini. Menurut (Agustina and Wahyudi, 2015), “Game atau permainan adalah suatu cara belajar dengan menganalisa dengan sekelompok pemain maupun indvidual dengan menggunakan strategi-strategi yang rasional.” Berdasarkan kutipan tersebut game dapat digunakan sebagai sarana pembelajaran dan simulasi. Dalam pengembangan sebuah game, ada beberapa komponen yang harus ada, salah satunya adalah *Non-Player Character* (NPC). NPC adalah sebuah *game object* dalam game yang dikontrol oleh algortima.

NPC dapat dijumpai dalam berbagai jenis game, termasuk game aksi, game RPG, game petualangan, game simulasi. Dalam sebuah gama, NPC digunakan sebagai pemandu, lawan main, atau sebagai karakter yang memberikan misi (Siswanto and Suni, 2021). NPC juga dapat digunakan sebagai karakter yang bergerak secara otomatis sesuai dengan algoritma yang diterapkan. Gerak NPC dapat diatur agar memiliki interaksi dengan pemain, interaksi dengan NPC lain, seperti memberikan informasi atau mengikuti rutinitas yang ditentukan. Namun Dalam konteks penentuan rute gerak NPC, terdapat beberapa masalah yang harus diatasi.

Salah satu masalah yang sering dihadapi dalam penentuan rute gerak NPC adalah konflik rute (Wu, 2023). Dalam game, NPC dapat bergerak secara bersamaan dan dapat saling berinteraksi. Hal tersebut dapat menyebabkan adanya

konflik rute dari NPC dikarenakan saling berinteraksi dan saling menghalangi jalur gerak. Alhasil konflik rute menyebabkan NPC tidak dapat bergerak dengan baik dan menyulitkan pemain dalam menikmati game yang digunakan. Maka dari itu diperlukan adanya optimalisasi rute.

Optimalisasi rute juga merupakan masalah yang sering dihadapi dalam penentuan rute gerak NPC. NPC yang bergerak secara otomatis dalam game harus bergerak dengan cepat dan optimal. Rute optimal dalam NPC yang dimaksud adalah rute yang digunakan oleh karakter-karakter yang dikendalikan oleh sistem dalam game atau aplikasi untuk bergerak dari satu titik ke titik lainnya. Rute ini ditentukan oleh algoritma yang digunakan dalam aplikasi atau game tersebut dan dapat mengikuti berbagai kriteria yang ditentukan oleh pengembang, seperti jarak terpendek, waktu terpendek, atau evasi dari musuh (Harsadi, Asmiatun and Putri, 2021). Rute optimal NPC bertujuan untuk membuat NPC bergerak secara realistis dan membuat interaksi dengan pemain lebih menyenangkan. Rute Rute yang tidak efisien akan menyebabkan NPC bergerak dengan lambat dan menyulitkan pemain dalam menikmati game yang digunakan.

Dalam permasalahan optimalisasi rute NPC pada game, terdapat beberapa algoritma yang dapat diimplementasikan, diantaranya ada algoritma A*, algoritma dijkstra, *depth-first search* (DFS), *breath-first search* (BFS), algoritma *bee colony optimation*, algoritma *ant colony optimation* dan algoritma PSO (*Particle Swarm Optimization*) (Novalia, Rahmidani and Tasman, 2018). Setiap metode memiliki kelebihan dan kelemahan yang unik dan dapat diterapkan dalam kondisi yang beragam. Namun dalam penelitian ini algoritma yang akan digunakan adalah

algoritma PSO. PSO dipilih berdasarkan hasil dari penelitian yang dilakukan oleh (Pyke and Stark, 2021) bahwa algoritma PSO dapat meningkatkan kinerja NPC *pathfinding* hingga sekitar 20% dibanding algoritma lain yang digunakan dalam penelitian tersebut.

Algoritma *Particle Swarm Optimization* (PSO) adalah salah satu algoritma yang dapat digunakan untuk menyelesaikan masalah optimisasi(Nurahman, 2020). PSO adalah algoritma yang dapat digunakan untuk mencari solusi terbaik dari masalah optimisasi dengan cara mengoptimalkan kinerja dari sekumpulan partikel yang bergerak dalam ruang pencarian solusi dengan memperhitungkan posisi dan kecepatan acak serta menentukan *pbest*(posisi terbaik tiap partikel) dan *gbest*(posisi terbaik seluruh partikel) serte mempertimbangkan kriteria-kriteria seperti konflik rute, dan optimalisasi rute (Mu'min, Nugroho and Susiki, 2015). Implementasi algoritma PSO pada penentuan rute gerak NPC dalam game yang diteliti akan membantu dalam meningkatkan optimalisasi rute gerak NPC, dan mengurangi konflik rute. Selain itu, implementasi algoritma PSO juga dapat digunakan dalam aplikasi lain yang memerlukan pengoptimasian rute, seperti sistem logistik, sistem transportasi, dan lain-lain (Mu'min, Nugroho and Susiki, 2015).

Hal ini juga sesuai dengan ayat Al-Quran mengajarkan pentingnya usaha untuk mencari jalan yang benar atau rute yang optimal dalam kehidupan, seperti yang tertera dalam Surat Al-Isra ayat 57:

اُولٰۤىِٕكَ الَّذِيْنَ يَدْعُوْنَ يَبْتَغُوْنَ اِلٰى رَبِّهِمُ الْوَسِيْلَةَ اَيُّهُمْ اَقْرَبُ وَيَرْجُوْنَ رَحْمَتَهٗ وَيَخَافُوْنَ عَذَابَهٗ ۗ اِنَّ عَذَابَ رَبِّكَ كَانَ مَحْذُوْرًا

"Orang-orang yang mereka seru itu, mereka sendiri mencari jalan kepada Tuhan mereka siapa di antara mereka yang lebih dekat (kepada Allah) dan mengharapkan

rahmat-Nya dan takut akan azab-Nya; sesungguhnya azab Tuhanmu adalah suatu yang (harus) ditakuti" (Q.S Al-Isra [17]: 57).

Menurut tafsir Syaikh Dr. Shalih bin Abdullah bin Humaid (Imam Masjidil Haram) bahwa individu yang mereka seru dan sembah berasal dari kalangan Malaikat atau entitas lainnya yang secara aktif mencari sarana untuk mendekatkan diri kepada Allah melalui amal saleh. Bahkan, mereka saling berkompetisi untuk mencapai kedekatan yang lebih tinggi dengan-Nya melalui pelaksanaan berbagai perbuatan taat. Mereka juga sangat mengharapkan rahmat-Nya dan merasa takut terhadap azab-Nya. Sungguh azab Tuhamu itu -wahai Rasul- hendaknya dijauhi (Al-Mukhtashar, 2023). Berdasarkan tafsir yang telah dijelaskan dapat ditarik kesimpulan tentang pengimplementasian algoritma PSO juga dapat menjadi salah satu usaha untuk mencapai kebaikan dalam game yang di kembangkan, sesuai dengan ajaran Al-Quran untuk berbuat kebaikan dan mencari jalan ke Tuhan.

Penelitian ini akan menjelaskan bagaimana algoritma PSO dapat digunakan untuk menentukan rute gerak NPC dalam game *android*. Tahap-tahap yang digunakan dalam implementasi algoritma PSO akan dijelaskan secara detail, mulai dari perancangan algoritma, implementasi, hingga pengujian. Metode PSO diharapkan dapat memberikan solusi yang efektif dan efisien dalam menentukan rute gerak NPC pada game *android* dengan menggunakan algoritma PSO. Selain itu, penelitian ini juga diharapkan dapat menjadi sumber referensi yang berguna bagi para pengembang game, teknologi industri, dan peneliti yang berminat dalam bidang pengoptimasian rute dan juga membantu dalam mengimplementasikan prinsip-prinsip Al-Quran dalam proses pengembangan game.

1.2 Pernyataan Masalah

Seberapa efektif algorithma *single objective PSO (Particle Swarm Optimation)* jika dibandingkan dengan metode *bee colony optimation* dan *ant colony optimation* dalam menentukan rute optimal NPC?

1.3 Tujuan Penelitian

Untuk mengukur tingkat keefektifan algoritma *single objective PSO (Particle swarm optimation)* jika dibandingkan dengan metode *bee colony optimation* dan *ant colony optimation* dalam menentukan rute optimal NPC.

1.4 Manfaat Penelitian

Manfaat penelitian ini sebagai berikut :

a. Dapat meningkatkan kualitas game *android* dengan memberikan pengalaman yang lebih baik bagi pemain.
b. Dapat memberikan solusi untuk masalah yang dihadapi dalam pengembangan game *android* dalam menentukan rute optimal Multi-NPC.
c. Dapat menjadi dasar untuk penelitian selanjutnya dalam bidang optimisasi algoritma untuk menentukan rute optimal dalam pengembangan game *android*.
d. Dapat menambah referensi dalam bidang optimisasi algoritma dan pengembangan game *android*.

1.5 Batasan Masalah

Agar mencapai tujuan penelitian dengan efektif, perlu ditetapkan batasan agar perancangan menjadi lebih terstruktur. Berikut adalah batasan yang ditetapkan:

a. Penelitian ini tidak akan mencakup pengujian dari aspek grafis dan *gameplay* dari game yang digunakan.
b. Penelitian ini hanya menjelaskan metode diimplementasikan pada *behavior* mengejar dari NPC.
c. Pengukuran tingkat akurasi dalam penelitian ini hanya pada waktu tempuh *NPC* menemukan rute.
d. Parameter yang digunakan dalam algoritma yang diimplementasikan pada penelitian ini adalah jumlah partikel(NPC), bobot inersia, nilai *fitness*, kecepatan(*velocity)*.

BAB II

STUDI PUSTAKA

Studi pustaka ini bertujuan untuk mengkaji penelitian sebelumnya yang akan digunakan sebagai referensi oleh peneliti guna memfasilitasi pemahaman terhadap topik yang berkaitan dengan penelitian serta membandingkannya dengan penelitian yang sedang dilaksanakan.

2.1 Penelitian Terdahulu

Jurnal "Penerapan *Particle Swarm Optimization* Untuk *Balancing Ability* Pada *Team Battle* Game RPG" yang diteliti oleh (Rizaldi, Jonemaro and Akbar, 2018), Pada penelitian ini bertujuan untuk mengeksplorasi penggunaan algoritma *Particle Swarm Optimization* (PSO) dalam melakukan pengujian kemampuan karakter secara otomatis pada Game RPG dengan tujuan mengurangi biaya dalam pengembangan Game. Dalam jurnal ini dibahas bahwa PSO dapat diterapkan kedalam optimalisai fungsi didalam sebuah game. Fungsi dalam game memang harus dioptimalkan agar dapat memberikan pengalaman bermain yang baik bagi para pemain. Penelitian ini menunjukkan bahwa dengan menggunakan algoritma PSO dapat mencari nilai *weight* yang tepat pada saat *traning controller*. Kemudian hasil dari nilai *weight* hasil dari implementasi PSO tersebut dapat menjalankan AI (*Artificial Intelegent*) yang mengerti peratuaran pada game. Hasil dari penelitian menunjukkan bahwa algoritma PSO dapat berjalan sesuai dengan tujuan penelitian yaitu untuk meningkatkan pencarian nilai *weight* yang tepat. Secara keseluruhan, jurnal ini menunjukkan bahwa algoritma PSO dapat digunakan dalam pencarian

nilai *weight*. Namun, pada hasil penelitian ini masih belum secara lengkap dijabarkan, seperti seberapa besar pengaruh nilai *weight* terhadap AI yang dijalankan, artinya perlu dilakukan riset lebih lanjut untuk mengevaluasi penerapan algoritma PSO.

Jurnal "Penerapan *Algoritma Particle Swarm Optimization* (PSO) untuk Optimisasi Pembangunan Negara dalam *Turn Based Strategy* Game" yang ditelitih oleh (Setiawan, Santoso and Adipranata, 2019) mengeksplorasi penggunaan algoritma *Particle Swarm Optimization* (PSO) dapat diterapkan dalam menentukan unit dan bangunan yang diperlukan, serta mengarahkan unit lawan untuk bergerak ke *tile* tertentu dan menjalankan fungsi pada unit tersebut. Dalam penelitian ini, beberapa parameter, seperti statistik kota, jenis *terrain*, dan sumber daya pada *tile* tertentu, diimplementasikan dalam algoritma PSO untuk mencapai hasil yang diinginkan. Hasil penelitian menunjukkan bahwa algoritma PSO mengungguli metode acak, terlihat dari perbedaan yang signifikan pada batas waktu 150 *turn*. Selain itu, perubahan nilai variabel pada rumus kecocokan (*fitness*) menghasilkan output permainan yang berbeda. Secara keseluruhan, jurnal ini menunjukkan bahwa belum ada bukti yang dapat menentukan apakah algoritma PSO lebih baik daripada metode acak karena keterbatasan jumlah putaran yang digunakan dalam penelitian ini. Oleh karena itu perlu dilakukan peningkatan *turn* serta *solution space* dalam menentukan tujuan dari implementasi PSO tersebut agar dapat menemukan nilai *fitness* terbaik sesuai dengan kebutuhan.

Jurnal "*Using Particle Swarm Optimization as Pathfinding Strategy in a Space with Obstacles*" yang diteliti oleh (David, 2021) merupakan penelitian

dengan membahas strategi dalam *pathfinding* merupakan salah satu komponen penting untuk meningkatkan efisiensi *path planning* untuk berbagai aplikasi. Penelitian ini bertuuan untuk menyelidiki efek parameter PSO (jumlah partikel, konstanta bobot, konstanta partikel, dan konstanta global) pada kinerja algoritma untuk memberikan *solution path*. Peningkatan parameter PSO membuat *swarm* bergerak lebih cepat ke titik target, namun membutuhkan waktu yang lama untuk menyatu karena terlalu banyak gerakan acak, dan sebaliknya. Jurnal ini juga mengevaluasi kinerja dari setiap teknik yang digunakan dalam penelitian ini, dan menunjukkan bahwa algoritma PSO dapat memberikan *solution path* untuk berbagai jenis tugas di dalam ruangan yang memiliki *obstacle.* Secara keseluruhan, jurnal ini menunjukkan bahwa algoritma PSO dapat digunakan untuk memecahkan masalah perencanaan jalur. Namun, perlu dilakukan riset lebih dalam lagi dikarenakan masih belum ada penjelasan tentang berapa banyak *obstacle* yang digunakan dan berapa besar ruangan yang dipakai dalam memecahkan masalah perencaan jalur. Semestinya sebuah algoritma optimasi tidak hanya dilihat dari total 4 parameter yang ada pada penelitian ini namun bisa 6 parameter yang diukur.

Jurnal "*A Comparative Study of Genetic Algorithm and Particle Swarm Optimization in Context of Plant Layout Optimization*" yang ditelitih oleh (Sharad Kumar and Mishra, 2019) mengeksplorasi penggunaan algoritma *Particle Swarm Optimization* (PSO) dalam mengoptimalkan masalah tata letak di mana fungsi multiobjektif diambil kedalam pertimbangan. Pada penelitian ini memiliki tujuan utama yaitu adalah membandingkan efektivitas dan efisiensi komputasi dari Algoritma Genetika (GA) dan PSO menggunakan pendekatan uji hipotesis formal.

Penelitian ini berusaha untuk memeriksa dugaan bahwa PSO memiliki efektivitas yang sama seperti GA tetapi dengan efisiensi komputasi yang jauh lebih baik (evaluasi fungsi yang lebih sedikit) dengan melakukan analisis statistik dan uji hipotesis formal. Hasil dari penelitian menunjukkan bahwa meskipun algoritma PSO dan GA rata-rata memberikan efektivitas yang sama (dari kualitas solusi), PSO lebih efisien secara komputasi daripada GA. Dalam jurnal ini juga disebutkan bahwa dalam PSO, partikel memperbarui diri mereka dengan kecepatan internal. Mereka juga memiliki memori, yang penting bagi algoritma. Selain itu, solusi potensial, yang disebut partikel, "diterbangkan" melalui ruang masalah dengan mengikuti partikel optimum saat ini. Namun dalam penelitian tersbut masih terdapat kekurangan dari proses generalisasi sample yang terlalu kecil sehingga berakibat kurangnya validitas dalam pengujian tujuannya. Maka dari itu perlu adanya randomisasi dari konstanta yang ada pada PSO agar pergerakan dari partikel dapat lebih variatif dan efisien.

2.2 Landasan Teori

2.2.1 Game

Game adalah sebuah aktivitas yang dilakukan untuk menghibur atau sebagai hobi, yang dapat dilakukan secara individu atau dalam kelompok. Game dapat berupa permainan fisik atau elektronik, yang dapat dimainkan di konsol atau perangkat elektronik lainnya. Game elektronik dapat memiliki beberapa jenis, seperti game PC, game konsol, game mobile, dan game online. Setiap game yang ada pasti memiliki sebuah *goal* atau tujuan yang harus dicapai, dan pemain diharuskan terlibat langsung dalam proses pencapaian *goal* tersebut sehingga

pemain merasa memiliki kontrol penuh apapn yang terjadi dalam permainan tersebut.

Game memiliki makna yang mendasar sebagai bentuk permainan. Istilah "permainan" dalam konteks ini merujuk pada konsep kecerdasan intelektual (*Intellectual Playability Game*), yang juga dapat diinterpretasikan sebagai lingkungan di mana pemain membuat keputusan dan melakukan tindakan (Dill, 2020). Permainan itu sendiri memiliki arti sebagai sebuah sistem di mana pemain terlibat dalam konflik buatan, di mana pemain berinteraksi dengan sistem dan konflik yang telah dirancang atau dibuat secara sengaja, dan di dalamnya terdapat tujuan dan peraturan yang bertujuan untuk mengatur perilaku pemain dan menentukan jalannya permainan (Avedon, E. M. and Smith, 1971).

Menurut (Santoso, Budhi and Intan, 2017) berdasarkan genrenya, game dapat dikempokkan menjadi 10 kategori, yaitu:

a. *Action - Shooting*: Jenis game ini membutuhkan kecepatan refleks, koordinasi mata-tangan, dan *timing* yang baik.
b. *Fighting* (pertarungan): Game ini juga memerlukan tingkat kecepatan refleks dan koordinasi mata-tangan yang tinggi, namun inti dari permainan ini terletak pada penguasaan mekanisme permainan, pemahaman karakter, dan timing yang akurat.
c. *Action - Adventure*: Jenis game ini mengalami perkembangan menjadi sebuah genre yang menggabungkan unsur aksi dan pertualangan. Dalam perkembangannya, game-game ini cenderung menggunakan grafis tiga dimensi (3D) dan sudut pandang orang ketiga.

d. *Adventure*: Pada jenis game ini, gerakan fisik seperti berlari, melompat, atau menembak tidak menjadi kebutuhan utama. Fokus utama dari game petualangan ini adalah pada aspek cerita dan kemampuan berpikir yang dimiliki oleh pemain.

e. Simulasi Konstruksi dan manajemen: Jenis game ini seringkali mencoba merepresentasikan dunia nyata dengan tingkat kedekatan dan perhatian terhadap berbagai faktor yang terperinci.

f. *Role Playing*: Jenis game ini cenderung mengedepankan peran karakter atau perwakilan pemain dalam permainan. Terdapat penekanan khusus pada aspek ini.

g. Strategi: Jenis game ini lebih mengandalkan kemampuan berpikir dan pengambilan keputusan yang hati-hati dan terencana. Jenis game ini dapat dibagi menjadi dua kategori:

   - *Real time Strategy,* di mana permainan berlangsung dalam waktu nyata dan semua pihak harus mengambil keputusan setiap saat.
   - *Turn based Strategy*, di mana permainan berlangsung secara bergantian, di mana setelah pemain mengambil langkah dan menggerakkan pasukannya, giliran lawan menunggu, dan sebaliknya.

h. *Puzzle*: Video game jenis ini sesuai namanya berintikan mengenai pemecahan teka-teki, baik itu menyusun balok, menyamakan warna bola, memecahkan perhitungan matematika, melewati labirin, hingga mendorong-dorong kota masuk ke tempat yang seharusnya, itu semua termasuk dalam jenis ini.

i. Simulasi kendaraan: Video game jenis ini bertujuan untuk menyajikan pengalaman atau interaktifitas yang semirip mungkin dengan kendaraan aslinya, meskipun terkadang kendaraan tersebut dapat bersifat eksperimental atau bahkan fiktif. Namun, terdapat penekanan khusus pada penampilan detail dan pengalaman yang realistis dalam menggunakan kendaraan tersebut.
j. Olahraga: Dalam bentuk yang singkat, padat, dan jelas, *sport* yang dimainkan di PC atau konsol merupakan permainan yang dirancang dengan tingkat realistis yang tinggi.

2.2.2 NPC (*Non-Player Character*)

NPC (*Non-Player Character*) adalah karakter yang ditambahkan dalam sebuah game yang dioperasikan oleh sistem atau algoritma daripada pemain. NPC dapat berperan sebagai tokoh utama, tokoh pendukung, atau bahkan musuh dalam game. NPC dapat digunakan untuk menyediakan misi, menjual barang, memberikan petunjuk, atau melakukan tugas lainnya yang dapat membantu pemain dalam menyelesaikan game.

NPC dapat dibedakan dari karakter pemain (PC) yang dioperasikan oleh pemain sendiri. NPC dapat ditentukan dengan AI (*Artificial Intelligence*) untuk membuat karakter tersebut berinteraksi dengan lingkungan dan pemain dalam cara yang lebih alami. NPC dapat ditetapkan dengan skrip yang ditentukan sebelumnya atau dapat diatur dengan algoritma *pathfinding* untuk menentukan jalur yang dapat ditempuh oleh NPC dalam mencapai tujuan tertentu.

NPC pada game yang diteliti ini merupakan sebuah *enemy*. Perilaku dari NPC tersebut nantinya akan berusaha mencari pemain, kemudian NPC akan berusaha mengejar pemain tersebut. Dan ketika NPC tersebut sudah menangkap pemain, nantinya baik pemain maupun NPC bisa saling serang demi mempertahankan hidup masing masing.

2.2.3 *Android*

Android adalah sistem operasi *mobile* yang dikembangkan oleh Google. *Android* dibuat berdasarkan kernel Linux dan dapat digunakan pada berbagai perangkat seluler seperti *smartphone,* tablet, *smartwatch*, dan perangkat lainnya. *Android* menyediakan antarmuka pengguna yang intuitif dan dapat digunakan untuk menjalankan berbagai aplikasi seperti game, aplikasi sosial media, navigasi, dll.

Android juga menyediakan platform pengembangan aplikasi yang dikenal dengan *android* SDK (*Software Development Kit*) yang memungkinkan pengembang untuk membuat aplikasi yang kompatibel dengan perangkat *android*(Imamum Huda, 2011). Aplikasi yang dibuat dengan menggunakan *android* SDK dapat diunduh dan diinstal melalui toko aplikasi seperti *Google Play Store*. Selain itu, *android* juga memiliki arsitektur terbuka yang memungkinkan pengembang untuk menyesuaikan sistem operasi sesuai dengan kebutuhan perangkat atau aplikasi yang dikembangkan.

Android juga memiliki versi yang berbeda-beda, seiring dengan perkembangan teknologi, Google terus mengeluarkanversi terbaru dari sistem

operasi *android* dengan fitur dan peningkatan yang lebih canggih. Beberapa versi populer dari *android* antara lain:

a. *Android Gingerbread* (versi 2.3 - 2.3.7)
b. *Android Honeycomb* (versi 3.0 - 3.2.6)
c. *Android Ice Cream Sandwich* (versi 4.0 - 4.0.4)
d. *Android Jelly Bean* (versi 4.1 - 4.3.1)
e. *Android KitKat* (versi 4.4 - 4.4.4)
f. *Android Lollipop* (versi 5.0 - 5.1.1)
g. *Android Marshmallow* (versi 6.0 - 6.0.1)
h. *Android Nougat* (versi 7.0 - 7.1.2)
i. *Android Oreo* (versi 8.0 - 8.1)
j. *Android Pie* (versi 9.0)
k. *Android* 10 (versi 10)
l. *Android* 11 (versi 11)

Setiap versi dari *android* memiliki fitur dan perbaikan yang berbeda-beda, sehingga perangkat yang menjalankan versi yang lebih baru akan mendapatkan fitur dan peningkatan yang lebih canggih dibandingkan dengan perangkat yang menjalankan versi yang lebih lama.

2.2.4 Optimal

Optimal adalah kondisi atau hasil terbaik yang dapat diperoleh dari suatu proses atau sistem yang ditentukan oleh kriteria yang telah ditentukan. Dalam matematika dan ilmu komputer, optimal sering digunakan dalam konteks optimisasi, yang berarti mencari solusi terbaik dari suatu masalah yang diberikan.

Optimal dapat ditentukan dengan berbagai cara, tergantung pada kriteria yang digunakan untuk menilainya. Dalam pemecahan masalah optimisasi, kriteria yang digunakan untuk menentukan optimal dapat berupa fungsi tujuan yang harus dioptimalkan, contohnya dalam pemecahan masalah TSP (*Traveling Salesman Problem*) kriteria yang digunakan adalah jarak tempuh yang paling pendek. Optimal juga dapat ditentukan dengan kriteria subjektif, misalnya dalam pengambilan keputusan, kriteria yang digunakan adalah kepuasan atau preferensi dari pengambil keputusan.

Dalam konteks penelitian ini, optimal maksudnya adalah pemilihan rute yang paling baik atau paling sesuai dengan kriteria yang telah ditentukan serta tingat utilitas dari pengimplementasian algoritma *single-objective* PSO kedalam game berbasis *android* yang. Dalam hal penentuan rute, algoritma PSO diharapkan dapat menemukan rute paling optimal untuk NPC dalam game yang dikembangkan. Kriteria yang digunakan untuk menemukan rute optimal bisa berupa waktu yang dibutuhkan untuk mencapao tujuan atau jumlah langkah yang dibutuhkan.

2.2.5 Algoritma *Particle Swarm Optimation*

Particle Swarm Optimation (PSO) adalah sebuah algoritma yang digunakan untuk menyelesaikan masalah optimisasi(Vanilia Cahya N, 2019). PSO didasarkan pada simulasi perilaku serangga dan burung yang bergerak dalam sebuah koloni, di mana setiap individu (partikel) dalam koloni mewakili sebuah solusi potensial untuk masalah yang dihadapi. Dalam PSO, setiap partikel memiliki posisi dan kecepatan yang mewakili solusi saat ini dan perubahan solusi dari waktu ke waktu. Kecepatan partikel diperbarui berdasarkan pengaruh dari dua faktor: interaksi antar

partikel dan interaksi dengan lingkungan. Interaksi antar partikel terjadi ketika partikel lain dalam koloni berpengaruh pada posisi dan kecepatan partikel yang bersangkutan. Interaksi tersebut direalisasikan melalui proses pengambilan rata-rata dari posisi terbaik yang pernah dicapai oleh setiap partikel dan posisi terbaik yang pernah dicapai oleh seluruh koloni (Vanilia Cahya N, 2019).

Interaksi dengan lingkungan terjadi ketika partikel berusaha untuk mencapai posisi terbaik yang ditentukan oleh lingkungan. Dalam PSO, lingkungan ditentukan oleh fungsi tujuan yang ingin dioptimalkan. Partikel akan berusaha untuk mencapai posisi terbaik yang meminimalkan atau maksimalkan fungsi tujuan tersebut. Analogi sederhana dari algoritma ini adalah mencontoh perilaku dari kawanan burung. Dalam sebuah kawanan burung pastinya memiliki dependensi kecepatan, kecerdasan, dan cenderung mengikuti kebiasaan burung lain secara masing masing, biasanya sebagian besar burung tersebut akan mengikuti kebiasaan sebagai berikut:

a. Posisi burung-burung tidak saling berdekatan secara signifikan.
b. Ketika terbang, burung cenderung bergerak secara kolektif dengan burung-burung lainnya.
c. Burung akan menempati posisi yang umumnya diikuti oleh burung-burung lainnya dan tetap memperhatikan rute kelompok burung agar jarak tidak terlalu jauh

Karakteristik gerombolan burung didasarkan pada kombinasi dari tiga faktor sederhana berikut:

a. Kohesi - terbang secara bersama-sama.
b. Separasi - menjaga jarak yang tidak terlalu dekat.

c. Penyesuaian (*alignment*) - mengikuti arah bersama

Oleh karena itu, *Particle Swarm Optimization* (PSO) ditingkatkan dengan memperhatikan hal-hal berikut:

a. Ketika burung menemukan sumber makanan atau target (atau dalam konteks PSO, nilai minimum atau maksimum dari sebuah fungsi), burung tersebut dengan cepat mengirimkan informasi kepada gerombolan burung lainnya dalam kelompok khusus.
b. Gerombolan burung lainnya segera mengikuti menuju sumber makanan yang telah dikomunikasikan, meskipun tidak secara bersamaan.
c. Terdapat partikel yang bergantung pada pandangan burung masing-masing, yaitu ingatan terhadap jalur yang telah dilalui sebelumnya.

Setiap komponen dalam PSO melakukan pencarian jalur keluar terbaik dan melewati ruang pencarian (*search space*). Dalam konteks ini, setiap komponen melakukan adaptasi pada tempat terbaik komponen itu sendiri (*personal best*) dan adaptasi pada tempat terbaik dari seluruh kelompok (*global best*) sepanjang perjalanan pencarian. Sebagai hasilnya, penyebaran pengetahuan atau informasi terjadi di dalam setiap komponen secara independen, dengan jarak yang relatif jauh antara satu komponen dengan komponen utama dari semua kelompok selama operasi pencarian jalur keluar. Iterasi khusus dilakukan untuk mencari tempat terbaik pada setiap komponen hingga mencapai kondisi yang relatif stabil atau mencapai batas iterasi yang telah ditentukan. Pada setiap iterasi, setiap jalur keluar akan dievaluasi kinerjanya menggunakan fungsi kebugaran (*fitness function*).

Secara umum, PSO dapat digunakan untuk menyelesaikan masalah optimisasi yang memiliki lebih dari satu variabel dan memiliki ruang pencarian yang besar(Slowik, 2011). PSO juga dapat digunakan untuk mengatasi masalah yang memiliki kendala *non-linier* dan menemukan solusi yang optimal dalam jumlah iterasi yang relatif sedikit.

- *Single Objective Particle Swarm Optimation*

Single-objective PSO *(Particle Swarm Optimation)* adalah varian dari algoritma PSO yang digunakan untuk mencari solusi optimal dari masalah optimasi yang memiliki satu tujuan atau objektif saja. Dalam *single-objective* PSO, setiap partikel dioptimalkan untuk mencapai nilai yang paling baik dari satu fungsi tujuan saja. Pada *single-objective* PSO hanya memiliki satu nilai *fitness* paling optimal, berbeda dengan *multi-objective* PSO yang memiliki beberapa nilai *fitness* yang harus dioptimalkan secara bersamaan. Maka dari itu *single-objective* PSO memiliki keunggulan dalam kompleksitas komputasi yang lebih rendah dibandingkan dengan *multi-objective* PSO.

Nilai *fitness* digunakan untuk mengukur seberapa baik posisi partikel tersebut dalam menyelesaikan masalah optimisasi yang dihadapi. Nilai *fitness* dalam PSO dihitung dengan mengevaluasi fungsi obyektif pada posisi partikel saat ini. Fungsi obyektif ini mewakili masalah optimisasi yang akan dipecahkan, misalnya untuk menemukan nilai minimum atau maksimum dari suatu fungsi. Jadi, semakin baik posisi partikel dalam menyelesaikan masalah optimisasi, maka nilai fitness yang dihasilkan akan semakin baik. Setiap partikel juga menyimpan informasi tentang posisi terbaik yang pernah dicapainya (*pbest*) dan posisi terbaik

yang pernah dicapai oleh partikel lain dalam *swarm* (*gbesy*). Nilai *fitness* dari posisi terbaik ini juga digunakan dalam proses optimisasi untuk menentukan arah perpindahan partikel selanjutnya.

Secara umum, dalam PSO, setiap partikel akan bergerak mengikuti posisi terbaik yang pernah dicapainya sendiri dan posisi terbaik yang pernah dicapai oleh partikel lain dalam *swarm*. Dengan demikian, *swarm* akan bergerak menuju solusi optimal dari masalah optimisasi yang dihadapi.

2.2.6 Algoritma *Bee Colony Optimation*

Salah satu algoritma optimasi yang dibangun berdasarkan konsep *Swarm Intelligence* (SI) dan menarik minat peneliti adalah Algoritma *Bee Colony Optimization* (BCO). Pada tahun 2005, Dervis Karaboga mengusulkan algoritma ini. Algoritma *Bee Colony Optimization* mensimulasikan proses pencarian nektar oleh koloni lebah. Dalam mencari sumber makanan, koloni lebah terbagi menjadi tiga kelompok, yaitu lebah pengintai, lebah penjelajah, dan lebah pekerja. Ketiga kelompok ini bekerja bersama untuk menentukan kualitas dan lokasi sumber nektar, serta membandingkannya dengan sumber-sumber lain. Akhirnya, dipilihlah nektar dengan jarak optimal berdasarkan fungsi evaluasi yang telah ditetapkan (Yu, Chen and Zhang, 2018).

Algoritma *bee colony* adalah salah satu algoritma yang digunakan untuk pencarian jalur. Algoritma ini termasuk dalam kategori algoritma optimasi yang didasarkan pada perilaku kumpulan lebah madu dalam koloni untuk menemukan sumber makanan. Solusi yang mungkin dalam algoritma ini dinyatakan dalam bentuk posisi sumber makanan, sedangkan nilai-nilainya merepresentasikan jumlah

nektar yang terdapat dalam sumber makanan tersebut. Pendekatan yang digunakan dalam algoritma *bee colony optimization* adalah metaheuristik berbasis populasi, yang terinspirasi oleh perilaku kawanan lebah madu dalam mencari makanan. Implementasi standar algoritma *bee colony optimization* melibatkan tiga tahapan utama, yaitu::

a. Pada tahap pertama, inisialisasi solusi dilakukan dengan menghasilkan posisi sumber makanan secara acak. Untuk memperbarui solusi yang memungkinkan, setiap *employed bee* memilih kandidat posisi sumber makanan baru yang berbeda dari sebelumnya.
b. Pada tahap kedua, setiap *onlooker bee* memilih satu sumber makanan dari yang telah diperoleh oleh *employed bee*. Setelah memilih sumber makanan, *onlooker bee* pergi ke sumber makanan tersebut dan memilih posisi kandidat baru untuk sumber makanan. Pada tahap ini, ada batasan yang telah ditetapkan. Batasan ini merupakan pembatas dalam siklus algoritma *bee colony* dan mengontrol jumlah solusi tertentu yang tidak diperbarui. Setiap sumber makanan yang tidak memenuhi batasan tersebut akan dihapus dan tidak diikutsertakan dalam proses pembaruan solusi. Batasan ini memastikan bahwa hanya solusi yang memenuhi kriteria yang telah ditetapkan yang akan diperhitungkan lebih lanjut.
c. Tahap terakhir dalam algoritma *bee colony optimization* adalah tahap *update global best*. Pada tahap ini, setelah semua *employed bee* dan *onlooker bee* selesai melakukan pembaruan solusi, koloni lebah akan memperbarui informasi tentang solusi terbaik yang ditemukan secara

global. Informasi ini akan digunakan sebagai panduan dalam iterasi-iterasi berikutnya untuk memperbarui dan memperbaiki solusi yang dihasilkan.

Algoritma *bee colony optimization* bekerja berdasarkan prinsip pelepasan sejumlah lebah dalam proses penelusuran. Pada tahap awal, lebah pertama (*bee*-1) melakukan penelusuran maju (*forward*) dengan mencari jalur secara berurutan. Namun, pola penelusuran ini berbeda untuk lebah-lebah berikutnya. Lebah-lebah tersebut melakukan penelusuran mundur (*backward*) dengan mengacu pada posisi akhir lebah sebelumnya (*bee*-1). Dalam penelusuran mundur ini, lebah mencari titik yang memiliki pilihan jalur lebih dari satu dan belum dilalui oleh lebah-lebah sebelumnya. Jika titik seperti itu ditemukan, titik tersebut dijadikan sebagai titik asal baru untuk melakukan penelusuran maju oleh lebah yang bersangkutan. Dengan demikian, algoritma *bee colony optimization* menggunakan kombinasi penelusuran maju dan mundur oleh lebah-lebah untuk mencari jalur optimal dalam proses optimisasi.

Secara simultan, dilaksanakan proses replikasi jalur yang sebelumnya ditempuh oleh lebah (*bee*-1). Tahap replikasi dimulai dari titik awal yang baru hingga mencapai titik awal perjalanan sebelumnya atau posisi asalnya. Upaya pencarian secara progresif dilakukan sampai kriteria penghentian tercapai.

Interaksi informasi antara lebah memiliki peran yang penting. Ketika mereka memeriksa sarangnya, lebah dapat membedakan beberapa komponen yang ada di dalamnya. Salah satu komponen yang memiliki peran krusial adalah pertukaran informasi yang dikenal sebagai dancing area atau disebut *juga waggle dance*. Lebah yang telah menyelesaikan perjalanan pencarian masing-masing akan

melakukan *waggle dance*, yaitu gerakan tarian tubuh yang menggoyangkan-goyangkan, yang dilakukan oleh lebah-lebah yang berhasil menemukan tujuan tertentu. Lebah yang tidak berhasil menemukan tujuan tidak akan melakukan *waggle dance.* Durasi dari *waggle dance* menjadi indikator waktu yang menunjukkan bahwa jalur yang dilewati oleh lebah tersebut merupakan jalur terpendek.

Algoritma BCO dimulai dengan pembentukan sebuah populasi yang terdiri dari sekelompok agen lebah. Agen lebah ini terbagi menjadi tiga jenis berdasarkan peran atau tugas yang diemban, yang meliputi:

2.2.6.1 *Scout Bees*

Scout bees adalah lebah pertama yang bertindak sebagai agen, dan tugas mereka adalah melakukan pencarian secara acak terhadap posisi sumber makanan di sekitar sarang. Informasi tentang lokasi sumber makanan yang ditemukan oleh *scout bees* akan diteruskan kepada agen lebah berikutnya yang disebut e*mployed bees*.

2.2.6.2 *Employed Bees*

Employed bees, sebagai lebah agen, memainkan peran penting dalam berinteraksi langsung dengan sumber makanan yang sebelumnya telah ditemukan oleh *scout bees*. Tugas utama *employed bees* adalah mengumpulkan dan menyimpan informasi yang terkait dengan setiap sumber makanan. Informasi ini mencakup jarak dan arah dari sarang, tingkat profitabilitas atau nilai ekonomi dari sumber makanan, serta nilai kepentingan informasi tersebut untuk disebarluaskan.

Oleh karena itu, jumlah *employed bees* harus sesuai dengan jumlah sumber makanan yang telah ditemukan. Dalam proses penyebaran informasi tentang sumber makanan, *employed bees* melakukan tarian yang dikenal sebagai *waggle dance* di pusat sarang lebah, sementara *Onlooker Bees* menjadi penonton dan memilih sumber makanan yang akan diambil.

2.2.6.3 *Onlooker Bees*

Onlooker bees, sebagai lebah agen, bertugas dalam memilih dan memanfaatkan sumber makanan yang informasinya telah disimpan oleh lebah agen *employed bees*.

2.2.7 Algoritma *Ant Colony Optimation*

Algoritma *Ant Colony Optimization* (ACO) merupakan sebuah algoritma heuristik yang digunakan untuk menyelesaikan permasalahan optimisasi jarak. Algoritma ini merupakan salah satu implementasi dari *Swarm Intelligence* yang terinspirasi oleh strategi semut dalam mencari sumber makanan (Moshinsky, 1959). Semut melakukan pencarian makanan melalui jalur khusus dan meninggalkan jejak berupa *feromon* yang memungkinkan mereka kembali ke sarang. Untuk menentukan jalur terpendek dari sumber makanan, semut menggunakan jumlah jejak kaki atau *feromon* yang ada di jalur tersebut. Semakin tinggi jumlah jejak kaki atau *feromon*, semakin besar kemungkinan bahwa jalur tersebut adalah jalur terpendek.

Dalam Algoritma ACO, ada beberapa faktor dan langkah yang diperlukan untuk menjalankan proses optimisasi. Beberapa faktor yang terlibat dalam

algoritma ini meliputi jejak yang ditinggalkan oleh semut antar lokasi, siklus semut yang ditetapkan, kontrol intensitas jejak semut, kontrol visibilitas, visibilitas antar lokasi, jumlah semut, penguapan jejak semut yang ditetapkan, dan jumlah maksimum siklus. Rumusan dan persamaan Algoritma ACO yang digunakan dalam penelitian ini mengacu pada penelitian sebelumnya yang telah mempelajari penentuan faktor yang efektif dalam *Ant Colony Optimation* (Shrivastava and Kumar, 2018) dan dilakukan implementasi Algoritma *Ant Colony Optimation* untuk menyelesaikan kasus *Multi Traveling Salesman* Problem (Kencana, Harini and Mayuliana, 2017). Berikut adalah formulasi Algoritma ACO:

a. Proses Algoritma dimulai dengan mengatur parameter awal yang mencakup jumlah semut, jumlah titik, siklus semut yang ditetapkan, intensitas jejak semut, penguapan jejak semut, jumlah siklus maksimum, intensitas jejak semut antar titik, dan pengendalian visibilitas. Dalam probabilitas, terdapat dua nilai penting yang terlibat, yaitu nilai *feromon* yang ditentukan oleh pengguna dan nilai visibilitas.
b. Selanjutnya, rute-rute awal dibentuk secara acak berdasarkan nilai probabilitasnya.
c. Setiap semut melakukan pencarian panjang rute yang dilaluinya.
d. Dilakukan pengecekan kondisi berhenti, di mana salah satu kondisi yang memenuhi adalah mencapai jumlah iterasi maksimum (banyak iterasi). Jika kondisi berhenti belum terpenuhi, algoritma akan terus berjalan:
    - Perhitungan matriks perubahan intensitas *feromon* untuk setiap semut.

- Perhitungan matriks perubahan intensitas *feromon* global berdasarkan kontribusi setiap semut.
- Proses pembaruan nilai *feromon* berdasarkan perhitungan sebelumnya.

BAB III

DESAIN PENELITIAN

3.1 Deskripsi Game

Game ini merupakan sebuah game petualangan yang dikembangkan untuk perangkat *android*. Misi dari game yang harus diselesaikan yaitu mencari jalan agar pemain berhasil membeli makanan di kota dengan berbagai tantangan dan menemukan cara untuk bertahan diri dari serangan musuh. Musuh yang harus dihadapi oleh pemain merupakan mahluk hutan yang akan menjadi rintangan selama di game nanti. Mahluk hutan tersebut akan berperan sebagai NPC yang diberi beberapa *behavior* seperti *attack, patrol (walk* dan *idle), run*, dan *call friend* di dalamnya.

Konsep pengembangan game ini menggunakan sistem semi *open-world*, dimana pemain diberi akses untuk dapat menjelajahi seluruh hutan dengan bebas namun masih dengan batasan yang ditentutan. Display dalam game ini menggunakan display *isometric* yang menggambarkan perspektif 3 dimensi pada tampilan 2 dimensi. Perspektif dalam game akan memberikan tampilan yang terlihat lebih realistis berdasarkan lingkungan pemain dalam sudut yang berbeda. Dalam hal grafis, game ini mengambil konsep *hand-painted* untuk menampilkan grafis yang diciptakan oleh tangan dan dicat dengan manual. Game ini akan dianggap sukses dan berhasil apabila pemain berhasil bertahan hidup dari serangan musuh dan menyelesaikan misi yang diberikan yaitu membeli makanan di kota.

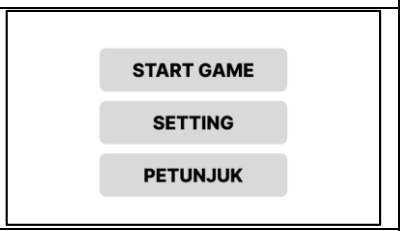
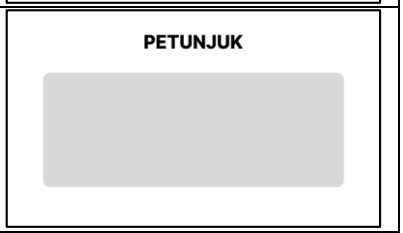
3.2 Skenario Game

Game yang diteliti memiliki latar belakang pemain yang akan berperan sebagai seorang manusia yang bertempat tinggal dihutan dan ingin membeli makanan di kota. Pemain harus mencari jalan keluar dari hutan dengan melewati tantangan dan menghindari bahaya yang ada di dalam hutan. Karakter yang ada dalam game yang diteliti ini hanya ada 2 yaitu pemain dan mahluk hutan sebagai NPC. Game yang diteliti ini memiliki plot yang akan pemain kembangkan sendiri. Tentunya masing-masing dari aksi yang dipilih memiliki plot yang berbeda dan misi yang ingin dicapai oleh pemain.

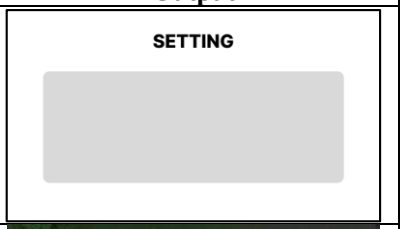
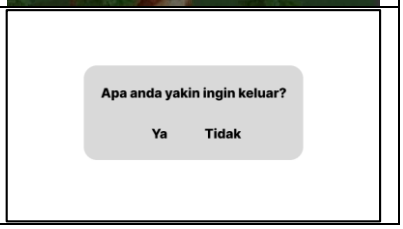
- *Storyboard*

Didalam *storyboard* akan dijelaskan cerita atau skenario akan ditampilkan dalam bentuk visual. *Storyboard* biasanya terdiri dari serangkaian gambar atau ilustrasi yang mewakili adegan-adegan yang akan ditampilkan dalam game. Dibawah ini merupakan rancangan dari *storyboard* game yang akan diteliti:

Tabel 3.1 Storyboard

Scene	**Input**		**Audio**	**Output**
1	Menu Utama	Menu yang akan menampilkan berbagai menu yang bisa dipilih pemain	Audio *background* musik	
2	Menu Petunjuk	Berisi tentang tata cara penggunaan *controller* game	Audio *background* musik	

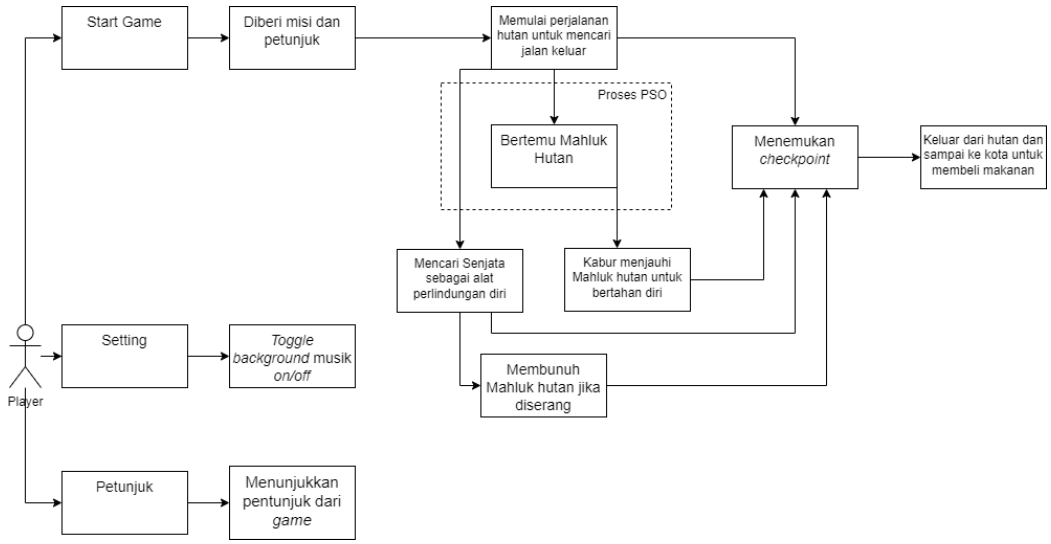
Tabel 3.1 Lanjutan

Scene	Input		Audio	Output
3	Menu *setting*	Berisi tentang pengaturan *background music*	Audio *background* musik	
4	Menu *start game*	Menu yang akan menampilkan *in game* aplikasi, pemain dapat melakukan misi yang diberikan.	Berisi Audio *background* music, *swing* audio, dan *death* audio	
5	Menu keluar	Menu yang menampilan pilihan kepada pemain apa ingin keluar aplikasi atau tidak	-	

Pada tabel 3.1 merupakan *storyboard* dari game yang akan diteliti dengan desain beberapa *output* yang masih berupa *wireframe*. Halaman dan menu yang akan dibuat di game yang diteliti ini akan berdasarkan desain *output storyboard* ini.

3.3 Desain Sistem

Desain sistem merupakan proses yang digunakan untuk merencanakan dan mengatur komponen-komponen yang akan digunakan dalam suatu sistem. Desain sistem meliputi pemilihan komponen, konfigurasi, dan interaksi antar komponen untuk mencapai tujuan yang diinginkan. Berikut merupakan diagram blok untuk desain rancangan sistem game.



Gambar 3.1 Desain Sistem

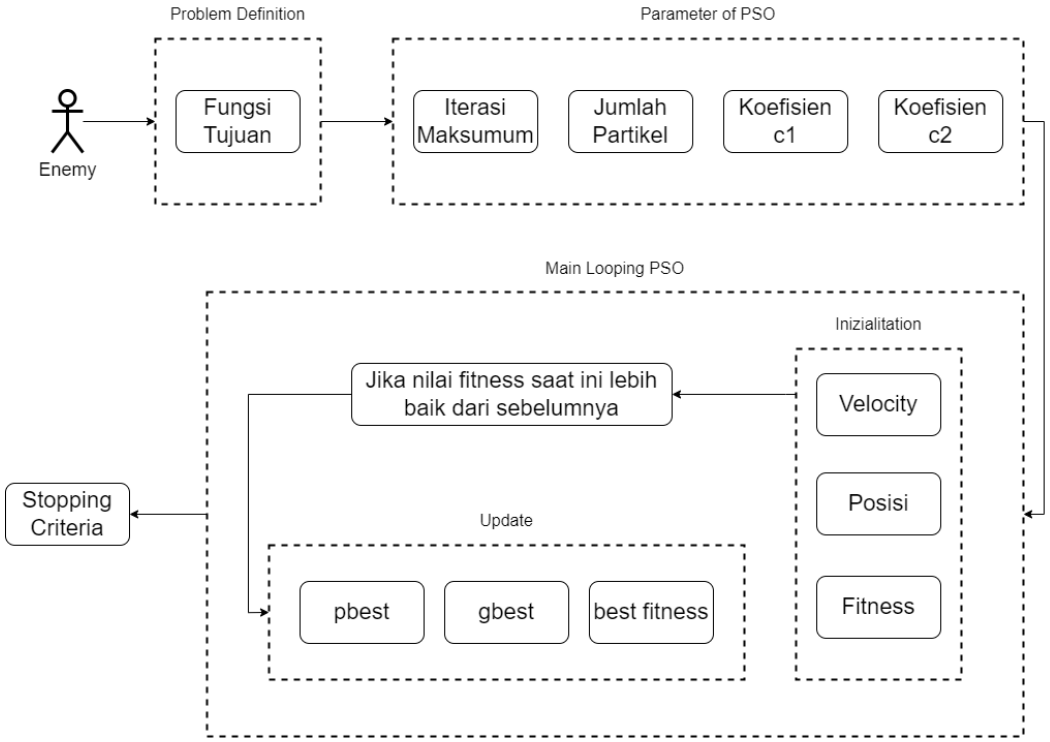
Berdasarkan desain diagram blok pada gambar 3.1, didapati bahwa rancangan dari sistem dimulai dari pemain diminta untuk memilih salah satu menu yang ada di dalam game. Jika pemain memilih tombol petunjuk, pemain akan diarahkan ke menu petunjuk dimana berisi tentang petunjuk penggunaan *controller* game. Jika pemain memilih tombol *setting*, pemain akan ditampilkan menu setting untuk mengatur hal penunjang dalam bermain game contohnya, pemain bisa mengatur apakah ingin ada *background* musik atau tidak. Jika pemain memilih tombol *start game* pemain akan langsung diarakan ke dalam gamenya. Setelah pemain berada di posisi *in game*, maka pemain dapat melakukan misi yang diberikan. Selanjutnya pemain dapat memulai pencarian jalan untuk menyelesaikan misi yang diberikan.

Pada saat pemain menjalankan misi yang diberikan, terdapat kondisi dimana pemain bertemu dengan NPC. Di kondisi ini ketika pemain masuk kedalam radius

serang NPC, NPC akan melakukan *decision* mengejar. Penentuan tersebut ditentukan berdasarkan penerapan algoritma *particle swarm optimation*.

3.4 Implementasi *Single Objective Particle Swarm Optimation*

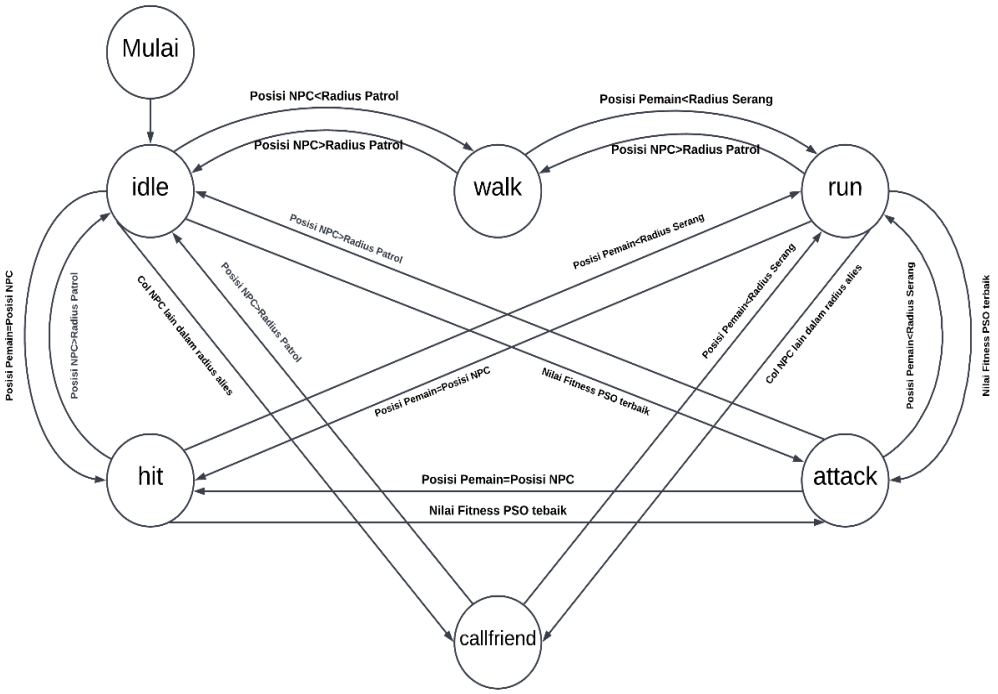
Implementasi *single objective particle swarm optimation* memiliki tujuan untuk menentukan keputusan dengan solusi optimal dari masalah optimasi. PSO didasarkan pada konsep partikel yang bergerak di dalam ruang pencarian solusi. Setiap partikel mewakili sebuah solusi potensial dari masalah optimasi. Partikel-partikel tersebut bergerak dalam ruang pencarian solusi, di mana setiap partikel memiliki kecepatan dan posisi yang unik. Kecepatan dan posisi partikel diperbarui berdasarkan interaksi dengan partikel lain dan dengan solusi terbaik yang telah ditemukan sebelumnya.



Gambar 3.2 Diagram Blok Proses PSO

3.4.1 *Finite State Machine* (FSM) NPC

Finite State Machine merupakan permodelan yang dapat mengontrol perilaku objek game. Dengan adanya FSM dapat memberikan cara dalam mengorganisir dan mengontrol transisi perilaku objek game dengan lebih mudal. Pada game yang diteliti terdapat total 6 *state* yaitu *idle, run, walk, attack, hit* dan *callfriend*. Berikut merupakan rancangan dari FSM pada game yang diteliti.



Gambar 3.3 *Finite State Machine NPC*

Implementasi *Finite state machine* pada gambar 3.3 diatas akan melakukan *state* yang dipengaruhu oleh *variable* posisi dari NPC. Ketika posisi NPC > radius patrol, *state* yang akan dijalankan adalah *idle.* Ketika posisi NPC < radius patrol maka yang akan dijalankan adalah *walk state. State run* dipengaruhi oleh *variable* posisi pemain, yaitu jika posisi pemain < radius serang. Untuk *state hit* akan dijalankan ketika posisi pemain = posisi NPC. Sedangkan ketika nilai *fitness* dari

hasil pengolahan algoritma sudah dinyatakan yang terbaik dari setiap iterasi, maka *state* yang dijalankan adalah *attack.* Dan yang terakhir adalah *state callfriend*, dimana *state* ini akan berjalan jika *collider* dari NPC lain berada dalam radius *allies.*

3.4.2 Desain *Variable*

Desain *variable* merupakan proses pemilihan dan pengaturan nilai dari parameter-parameter yang digunakan dalam suatu percobaan atau eksperimen. *Variable* ini dapat berupa kondisi lingkungan, jumlah sampel, atau kondisi lain yang mempengaruhi hasil percobaan. Desain *variable* harus diatur dengan baik agar hasil percobaan dapat diinterpretasikan dengan benar dan dapat di generalisasikan ke kondisi lain.

Tabel 3.2 Rancangan *Variable*

Nama *Variable*	**Keterangan**
Target	Digunakan sebagai pengganti dari fungsi tujuan dalam PSO, dalam target ini berisi tentang posisi dari *player* yang akan menjadi *stopping criteria* dari proses algoritma PSO. *Variable* ini bertipe data *int.*
MaxIt	Dalam *variable* ini nantinya berisi nilai iterasi maksimal dari proses evaluasi nilai *fitness. Varibale* ini bertipe data *double*
Enemy	Merupakan *variable* yang berisi total partikel yang akan digunakan dalam proses algoritma PSO, *variable* ini bertipe data *list*.
Velocity	Merupakan *variable* dinamis yang akan diupdate di setiap iterasi. *Variable* ini beritpe data *int.*
C1 dan C2	*Variable* ini merupakan nilai konstanta yang akan mempengaruhi proses evaluasi *velocity* tiap partikel. *Variable* ini bertipe data *double.*
Route	Merupakan *variable* dinamis yang akan diupdate setiap iterasi. *Variable* ini adalah nilai posisi saat ini dari *enemy. Variable* ini bertipe data *int.*
BestRoute	Merupakan *variable* dinamis yang akan diupdate setiap iterasi berdasarkan nilai dari *pbest route* hasil komparasi *pbest route enemy* yang dulu dengan yang sekarang. *Variable* ini bertipe data *int.*
Fitness	Merupakan *variable* dinamis yang akan diupdate setiap iterasi. *Variable* ini adalah nilai *fitness* dari hasil kalkulasi antara target dan posisi dari *enemy. Variable* ini bertipe data *double.*

Tabel 3.2 Lanjutan

Nama *Variable*	Keterangan
BestFitness	Merupakan *variable* dinamis yang akan digunakan sebagai tempat *update* nilai dari hasil minimum *fitness pbest enemy* dari setiap iterasi. *Variable* ini bertipe data *double.*
BestRouteSwarm	Merupakan *variable* dinamis yang akan digunakan sebagai tempat *update* nilai dari hasil terbaik *BestRoute* dari semua partikel yang ada dari setiap iterasi. *Variable* ini bertipe data *int.*
BestFitnessSwarm	Merupakan *variable* dinamis yang akan digunakan sebagai tempat *update* nilai *fitness* dari *BestRouteSwarm* dari setiap iterasi. *Variable* ini bertipe data *double*.

3.4.3 Perhitungan Manual

Dimisalkan ada persamaan partikel matematika yang terdapat dalam tabel berikut.

Tabel 3.3 Inisialisasi posisi awal partikel

F(partikel, iterasi)	Posisi
F(1,0)	-5
F(2,0)	-3
F(3,0)	2
F(4,0)	4

Kemudian penginisialisaian nilai dari kecepatan/*velocity*-nya.

Tabel 3.4 Inisialisasi *velocity* awal partikel

V(partikel, iterasi)	*Velocity*
V(1,0)	0
V(2,0)	0
V(3,0)	0
V(4,0)	0

Selanjutnya diberi soal untuk mencari nilai minimum dari persamaan diatas, dengan artinya penentuan *pbest* dan *gbest* selanjutnya akan dilihat dari nilai minimum dari setiap partikel. Contohnya seperti tabel 3.5.

Tabel 3.5 Fungsi objektif dengan *pbest* dan *gbest* awal

Fungsi Minimum Partikel	$Y = x^2$			
Partikel	***Pbest***		***Gbest***	
	Posisi	Nilai *Fitness*	Posisi	Nilai *Fitness*
1	-5	25	2	4

Tabel 3.5 Lanjutan

Fungsi Minimum Partikel	$Y = x^2$			
Partikel	***Pbest***		***Gbest***	
	Posisi	Nilai *Fitness*	Posisi	Nilai *Fitness*
2	-3	9	2	4
3	2	4		
4	4	16		

Akan dilakukan percobaan dengan iterasi sebanyak 3 kali. Tiap iterasi akan melakukan *update velocity* (V) dan posisi serta mencari nilai *fitness* dari *pbest* dan *gbest* dengan mengikuti rumus sebagai berikut:

a. Rumus untuk *update velocity* (V).

$$V(partikel - i) = V(i - k) + c1 \times r1\left(pbest\left(partikel(i)\right) - posisi\left(partikel(i - k)\right)\right) + c2 \times r2\left(gbest - posisi\left(partikel(i)\right)\right)$$

Keterangan:

$V(i - k)$ = Nilai *velocity* dari partikel i di iterasi ke k,

C_j = Nilai konstanta,

r1 = Bilangan random (0-1),

$posisi\left(partikel(i - k)\right)$ = Posisi sekarang dari partikel i di iterasi ke k,

$pbest\left(partikel(i)\right)$ = Posisi terbaik dari partikel i,

Gbest = Posisi terbaik dari kawanan.

b. Rumus untuk *update* posisi.

$$Posisi(partikelbaru - i) = posisi(partikellama - i) + V(partikel - i)$$

Keterangan:

$posisi(partikellama - i)$= Posisi sekarang dari partikel i di iterasi ke k,

$V(partikel - i)$ = Hasil dari nilai *velocity* terbaru.

Dimisalkan pada iterasi ke i = 1

a. Perhitungan nilai ***new*** *velocity*

$$V(1,1) = 0 + (1 \times 0{,}1)\big(-5 - (-5)\big) + 1 \times 0{,}2\big(2 - (-5)\big) = 0{,}014$$

$$V(2,1) = 0 + (1 \times 0{,}1)\big(-3 - (-3)\big) + 1 \times 0{,}2\big(2 - (-3)\big) = 0{,}010$$

$$V(3,1) = 0 + (1 \times 0{,}1)(2 - 2) + 1 \times 0{,}2(2 - 2) = 0{,}1$$

$$V(4,1) = 0 + (1 \times 0{,}1)(4 - 4) + 1 \times 0{,}2(2 - 4) = -0{,}04$$

b. Perhitungan nilai *update* ***new*** posisi

$$Posisi(1,1) = -5 + 0{,}014 = -4{,}9$$

$$Posisi(2,1) = -3 + 0{,}010 = -2{,}9$$

$$Posisi(3,1) = 2 + 0{,}1 = 2{,}1$$

$$Posisi(4,1) = 4 + (-0{,}04) = 3{,}9$$

Kemudian adalah proses *update* nilai *pbest* dengan meng*compare* nilai *fitness* partikel lama dan partikel baru.

Tabel 3.6 Perbandingan *pbest* lama dengan *pbest* baru

Fungsi Minimum Partikel	$Y = x^2$			
Partikel	***Pbest* Partikel Lama**		***Pbest* Partikel Baru**	
	Posisi	Nilai *Fitness*	Posisi	Nilai *Fitness*
1	-5	25	-4,9	24
2	-3	9	-2,9	8,4
3	2	4	2,1	4,41
4	4	16	3,9	15,21

Sehingga mendapatkan hasil *pbest*-nya sebagai berikut:

Tabel 3.7 *Update Pbest* iterasi ke 1

Pbest	
Posisi	Nilai *Fitness*
-4,9	24
-2,9	8,4
2	4
3,9	15,21

Sedangkan untuk mencari nilai *gbest*-nya dengan cara mencari nilai *fitness* dari *pbest* **baru**, bukan posisi.

Tabel 3.8 *Update Gbest* iterasi 1

Partikel	***Pbest***		***Gbest***	
	Posisi	Nilai *Fitness*	Posisi	Nilai *Fitness*
1	-4,9	24	2	4
2	-2,9	8,4		
3	2	4		
4	3,9	15,21		

Nilai *fitness* hanya digunakan untuk pembanding dalam menentukan *pbest* dan *gbest*, namun tetap nilai posisi adalah yang terpenting.

Dimisalkan pada iterasi ke i = 2

a. Perhitungan nilai ***new*** *velocity*

$$V(1,2) = 0{,}014 + (1 \times 0{,}1)\big(-4{,}9 - (-4{,}9)\big) + 1 \times 0{,}2\big(2 - (-4{,}9)\big) = 0{,}15$$

$$V(2,2) = 0{,}010 + (1 \times 0{,}1)\big(-2{,}9 - (-2{,}9)\big) + 1 \times 0{,}2\big(2 - (-2{,}9)\big) = 0{,}10$$

$$V(3,2) = 0{,}1 + (1 \times 0{,}1)(2 - 2) + 1 \times 0{,}2(2 - 2) = 0{,}2$$

$$V(4,2) = -0{,}04 + (1 \times 0{,}1)(3{,}9 - 3{,}9) + 1 \times 0{,}2(2 - 3{,}9) = 0{,}02$$

b. Perhitungan nilai *update* ***new*** posisi

$$Posisi(1,2) = -4{,}9 + 0{,}15 = -4{,}7$$

$$Posisi(2,2) = -2{,}9 + 0{,}10 = -2{,}8$$

$$Posisi(3,2) = 2 + 0{,}2 = 2{,}2$$

$$Posisi(4,2) = 3{,}9 + (-0{,}02) = 3{,}8$$

Tabel 3.9 Perbandingan *pbest* lama dengan *pbest* baru pada iterasi 2

Fungsi Minimum Partikel	$Y = x^2$			
Partikel	***Pbest* Partikel Lama**		***Pbest* Partikel Baru**	
	Posisi	Nilai *Fitness*	Posisi	Nilai *Fitness*
1	-4,9	24	-4,7	22
2	-2,9	8,4	-2,8	7,8
3	2	4	2,2	4,8
4	3,9	15,21	3,8	14,44

Tabel 3.10 *Update Pbest* iterasi ke 2

Pbest	
Posisi	Nilai *Fitness*
-4,7	22
-2,8	7,8
2	4
3,8	14,44

Sedangkan untuk mencari nilai *gbest*-nya dengan cara mencari nilai *fitness* dari *pbest* **baru**, bukan posisi.

Tabel 3.11 *Update Gbest* iterasi 2

Partikel	***Pbest***		***Gbest***	
	Posisi	Nilai *Fitness*	Posisi	Nilai *Fitness*
1	-4,7	22	2	4
2	-2,8	7,8		
3	2	4		
4	3,8	14,44		

Proses seperti yang digambarkan pada tabel 3.3 sampai tabel 3.11 akan terus berulang sampai pada *stopping criteria*. *Stopping criteria* bisa dari berbagai macam parameter, mulai dari jumlah iterasi maksimal, atau fungsi tujuan nilai *fitness*-nya tercapai atau dalam kondisi *konvergen.*

3.4.4 Karakteristik Algoritma Pembanding

3.4.4.1 Algoritma *Bee Colony Optimation*

Bee Colony Optimiation (BCO) adalah sebuah algoritma *swarm intelligence* yang terinspirasi oleh perilaku koloni lebah madu dalam mencari makanan. Karakteristik dari algoritma BCO antara lain:

a. Pemilihan solusi: Lebah pekerja memilih solusi berdasarkan fungsi objektif dan kriteria tertentu. Solusi yang terpilih dapat menjadi solusi lokal atau solusi global terbaik.

b. Komunikasi antar lebah: Lebah dalam koloni berkomunikasi dengan cara memberi tanda di lokasi makanan dan melakukan tarian dengan gerakan tertentu. Komunikasi ini memungkinkan koloni untuk menemukan solusi yang lebih baik secara bersama-sama.
c. Pemilihan lokasi: Lebah pekerja memilih lokasi berdasarkan ketersediaan makanan dan jarak dari sarang. Hal ini serupa dengan seleksi alam dalam evolusi.
d. Evaporasi *pheromone*: Setelah beberapa waktu, *pheromone* yang ditandai pada lokasi makanan akan menguap dan menghilang secara bertahap, sehingga memungkinkan koloni untuk mencari solusi baru.

BCO telah berhasil diaplikasikan pada berbagai permasalahan optimasi, seperti pengaturan jadwal, *routing*, dan *clustering*.

3.4.4.2 Algoritma *Ant Colony Optimation*

Ant Colony Optimiation (ACO) adalah algoritma optimasi yang terinspirasi dari perilaku koloni semut dalam mencari jalur menuju sumber makanan. Beberapa karakteristik algoritma ACO antara lain:

a. Pemilihan jalur berdasarkan jejak feromon: Semut akan memilih jalur berdasarkan tingkat feromon yang ditinggalkan oleh semut-semut sebelumnya. Semakin tinggi tingkat feromon, semakin besar kemungkinan jalur tersebut dipilih oleh semut berikutnya.
b. Peningkatan dan penurunan feromon: Feromon akan meningkat setiap kali semut melewati suatu jalur yang dipilih. Hal ini akan membuat jalur

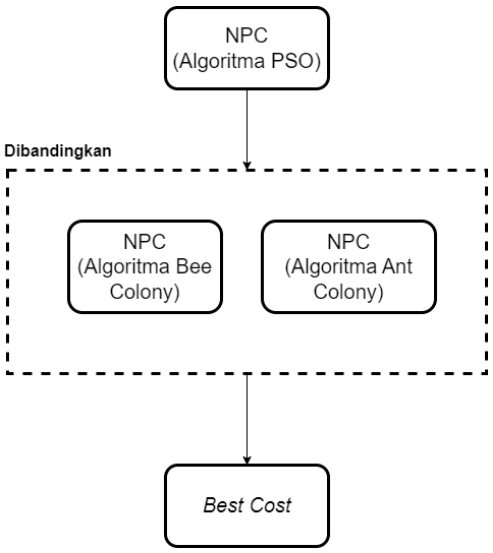
tersebut lebih menarik bagi semut berikutnya. Namun, feromon juga akan menguap seiring berjalannya waktu.

c. Penambahan variasi dalam pemilihan jalur: Semut juga memiliki kecenderungan untuk melakukan eksplorasi dengan memilih jalur-jalur yang belum banyak dilalui oleh semut lain. Hal ini akan menambah variasi dalam pemilihan jalur dan membantu semut menemukan jalur-jalur yang lebih baik.
d. Penggunaan *heuristic*: Selain menggunakan jejak feromon, semut juga menggunakan *heuristic* untuk memilih jalur. *Heuristic* dapat membantu semut dalam menentukan jalur yang lebih optimal.
e. Komunikasi antara semut Semut juga dapat berkomunikasi satu sama lain melalui jejak feromon. Hal ini dapat membantu semut menemukan jalur yang lebih baik dan menghindari jalur yang kurang optimal.

Dengan karakteristik-karakteristik tersebut, algoritma ACO dapat diaplikasikan pada berbagai masalah optimasi seperti penjadwalan, rute jaringan, dan desain parameter.

3.5 Desain Pengujian

Desain pengujian adalah proses perencanaan dan pengaturan suatu eksperimen untuk mencapai tujuan yang diinginkan. Termasuk pemilihan pengaturan kondisi percobaan, dan pemilihan sampel.



Gambar 3.4 Desain Pengujian

Desain dari pengujian dari algoritma *particle swarm optimation* akan melewati 2 proses pengujian. Proses pertama akan dibandingkan dengan 2 sample, yaitu menggunakan algoritma *bee colony optimation* dan algoritma *ant colony optimation* dimana nantinya akan dilakukan perbandingan dengan 3 kondisi yang berbeda, diantaranya:

a. Waktu tempuh NPC menemukan rute terhadap *player*
b. Waktu tempuh NPC menemukan rute yang ber-*obstacle* terhadap *player*.
c. Pemilihan rute NPC yang statis (sama) atau dinamis (tidak sama).

Setelah didapatkan hasil dari proses pertama, akan dilanjutkan dengan proses yang kedua yaitu mementukan *best cost.* Penentuan *best cost* disini akan dihitung dari perbandingan hasil proses pertama setiap algortima. Dari hasil tersebut diharapkan dapat memberikan hasil yang representatif.

BAB IV

IMPLEMENTASI DAN PEMBAHASAN

4.1 Implementasi

Bab ini akan membahas tentang implementasi dari rencana yang telah diajukan. Selain itu, dilakukan pengujian pada permainan untuk mengevaluasi sejauh mana permainan tersebut mencapai tujuan penelitian yang telah ditetapkan.

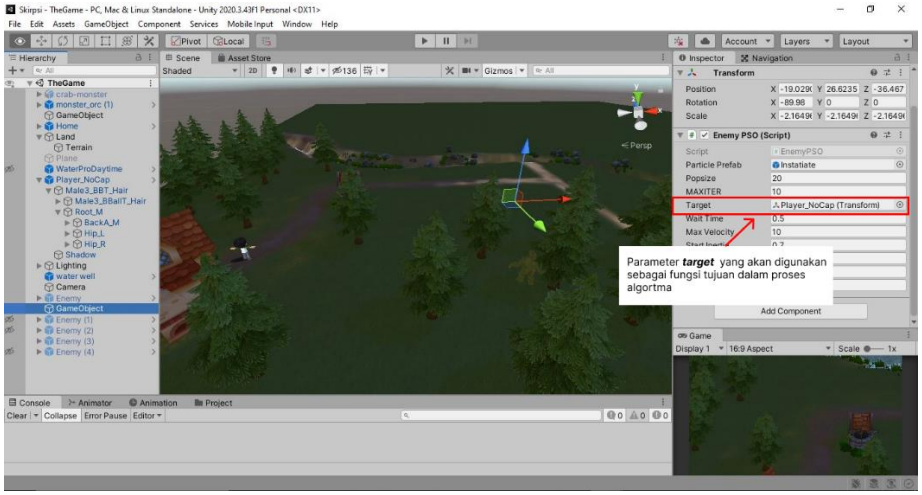
4.1.1 Implementasi Algoritma *Single Objective PSO* pada Game

Pada proses implementasi algoritma *Single Objective Particle Swarm Optimation*, dilakukan pembangunan sistem berdasarkan desain yang telah diusulkan. Pada tahap ini, dilakukan implementasi kecerdasan buatan pada karakter NPC dengan menggunakan algoritma *Single Objective Particle Swarm Optimization* (PSO). Implementasi ini melibatkan tiga proses utama yang akan dijelaskan melalui diagram blok yang terdapat dalam gambar 4.1.

Gambar 4.1 Diagram Blok Implementasi PSO

Secara umum implementasi algoritma *single objective particle swarm optimation,* menerapkan 3 langkah utama penentuan target, inisialisasi partikel, dan *update* posisi partikel.

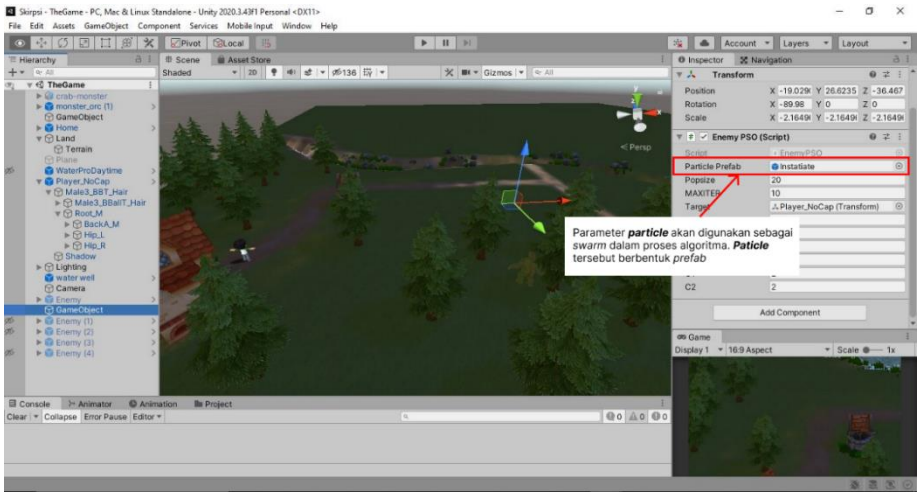
4.1.1.1 Penentuan Target



Gambar 4.2 Penentuan Target

Pada proses pertama ini akan dilakukan penentuan target dari proses algoritma *single objective particle swarm optimation.* Hal ini diperlukan agar partikel memiliki tujuan yang jelas yaitu mencari *game object player.*

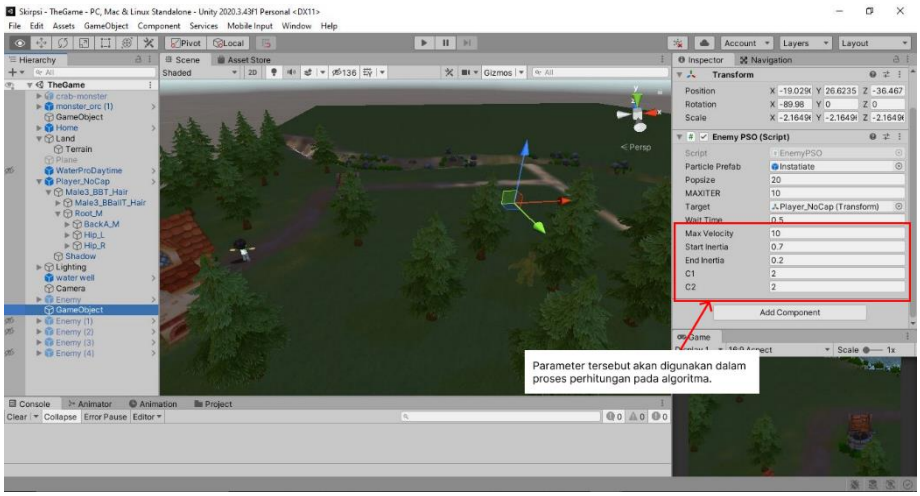
4.1.1.2 Inisialisasi Partikel



Gambar 4.3 Inisilasisasi Partikel

Algoritma *single objective particle swarm optimation* identik dengan sebuah partikel yang berkelompok, maka dari itu pada proses ini akan dilakukan proses penginisialisasian partikel tersebut. Dalam implementasi pada game yang diteliti, partikel akan disimpan dalam bentuk *prefab* yang selanjutnya akan dipanggil menggunakan fungsi *instantiate.*

4.1.1.3 *Update* Posisi Partikel

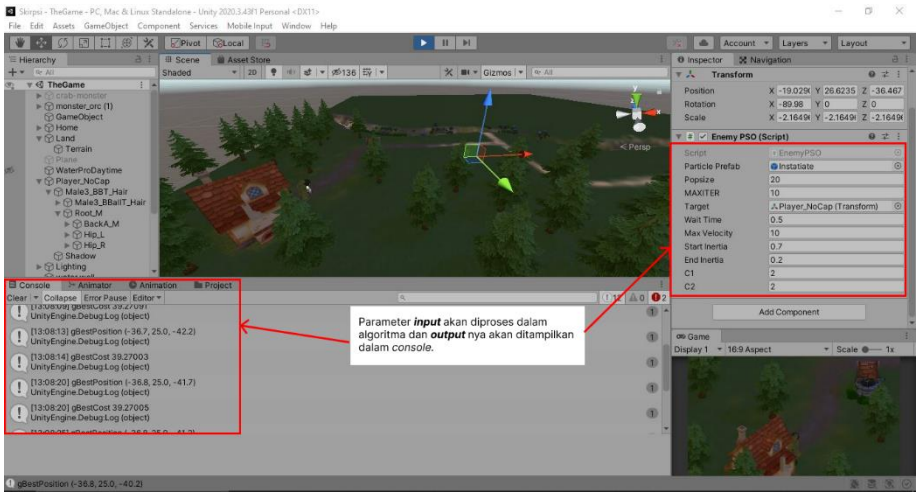


Gambar 4.4 Update Posisi Partikel

Pada proses ini akan dilakukan perpindahan posisi dari partikel berdasarkan hasil perhitungan di setiap *variable*. Setelah didapat hasil dari *Pbest* dan *Gbest* dari masing masing partikel, partikel tersebut akan berpindah mengikuti hasil dari *pbest* dan *gbest* tersebut.

4.1.2 *Output* masing – masing *variable*

Variable yang telah diinisialisasikan sebelumnya akan memperoleh hasil dari proses perhitungan yang ada dalam algoritma *single objective particle swarm optimation*. Hasil dari perhitungan tersebut akan tampil pada *console unity*.



Gambar 4.5 Letak Log Ketika Run

4.1.3 Pengujian Game

Pengujian ini berdasarkan desain pengujian yang telah dijelaskan pada bab 3. Pengujian akan dilakukan menggunakan perbandingan 3 NPC dengan masing masing NPC memiliki 3 algoritma yang berbeda, diantaranya adalah algoritma *single objective particle swarm optimation*, *bee colony optimation*, dan *ant colony optimation.* 3 algoritma tersebut akan melaui 3 jenis pengujian dengan 3 parameter pengukuran yang berbeda yaitu waktu tempuh NPC menemukan *player*, waktu tempuh NPC menemukan *player* ketika ada *obstacle*, dan yang terakhir pencarian rutenya apakah statis atau dinamis. Setiap pengujian pengukuran akan dilakukan 10 kali percobaan.

4.1.3.1 Waktu tempuh NPC menemukan rute terhadap *player*

Gambar 4.6 Layouting Awal Posisi NPC dan Player Parameter 1

Pada gambar 4.6 terdapat posisi awal dari NPC dan *player*. Posisi tersebut akan menjadi posisi yang akan digunakan disetiap algortima pada pengujian parameter pengukuran yang pertama. Namun pada proses pengujian, *player* akan berpatroli secara random atau bergerak secara otomatis ke arah ruang gerak yang diberikan.

Pada parameter pengukuran yang pertama akan dilakukan pengujian pada waktu tempuh NPC menemukan rute terhadap *player*. Nantinya akan dilakukan 10 kali percobaan pada masing-masing algoritma. Pada tabel 4.1 akan dijabarkan perbandingan waktu tempuh dari hasil pengukuran parameter yang pertama dari setiap algoritma di masing masing NPC.

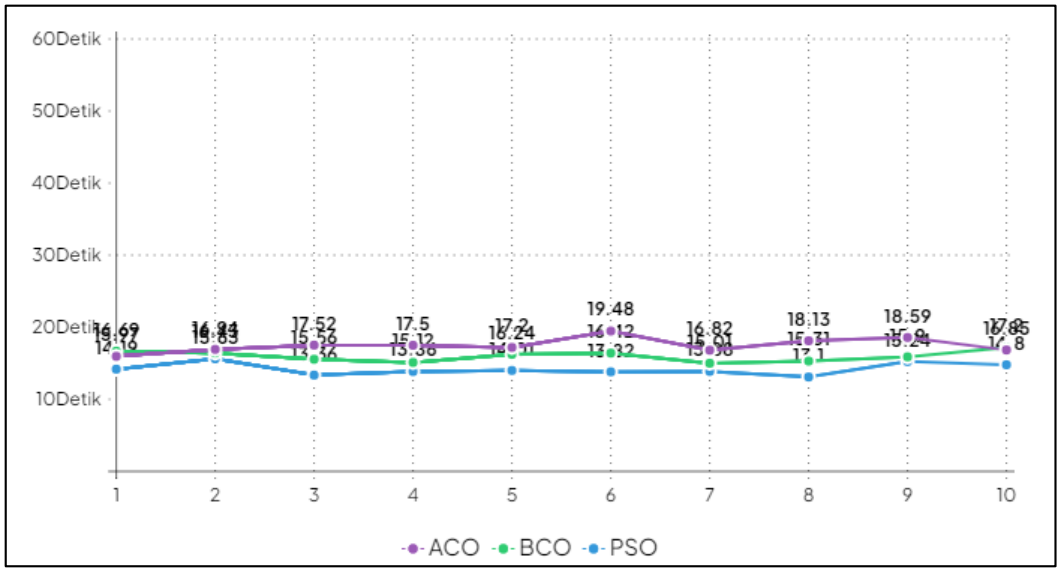
Tabel 4.1 Perbandingan Hasil Waktu Tempuh

Pengujian	***Particle swarm optimation***	***Bee colony optimation***	***Ant Colony optimation***
Pengujian ke 1	14,19 detik	16,69 detik	**15,97 detik**
Pengujian ke 2	15,63 detik	16,43 detik	16,94 detik
Pengujian ke 3	13,36 detik	15,56 detik	17,52 detik
Pengujian ke 4	13,86 detik	15,12 detik	17,50 detik
Pengujian ke 5	14,01 detik	16,24 detik	17,20 detik
Pengujian ke 6	13,82 detik	16,42 detik	19,48 detik

Tabel 4.1 Lanjutan

Pengujian	*Particle swarm optimation*	*Bee colony optimation*	*Ant Colony optimation*
Pengujian ke 7	13,88 detik	**15,01 detik**	16,82 detik
Pengujian ke 8	**13,10 detik**	15,31 detik	18,13 detik
Pengujian ke 9	15,24 detik	15,90 detik	18,59 detik
Pengujian ke 10	14,80 detik	17,20 detik	16,85 detik
Rata-rata	**14,18 detik**	**15,98 detik**	**17,5 detik**

Tabel 4.1 menujukkan hasil rata-rata perbandingan waktu tempuh dari algoritma *single objective particle swarm optimation, bee colony optimation,* dan *ant colony optimation,* yang dimana *single objective particle swarm optimation* memiliki waktu tempuh dengan rata rata 14,18 detik pada pengukuran menggunakan parameter yang pertama ini.



Gambar 4.7 Grafik Perbandingan Waktu Tempuh Parameter 1

Pada gambar 4.7 terdapat grafik perbandingan dari algoritma yang digunakan. Pada algoritma *single objective particle swarm optimation* (grafik berwarna biru) terdapat 5 kenaikan waktu tempuh dengan rataan kenaikaan 0,97 detik. Hal tersebut terjadi dikarenakan posisi dari *player* atau target dari algoritma

yang tidak berdiam di tempat yang mengakibatkan terdapat peningkatan proses iterasi dari algoritma. Pada algoritma *single objective particle swarm optimation* didapat bahwa percobaan ke-8 memiliki waktu tempuh tersingkat yaitu 13,10 detik.

Pada algoritma *bee colony optimation* (grafik warna hijau) terdapat 5 kenaikan waktu tempuh dengan rataan kenaikan 0,70 detik. Hal tersebut terjadi dikarenakan posisi dari *player* atau target dari algoritma yang tidak berdiam di tempat yang mengakibatkan terdapat peningkatan proses iterasi dari algoritma. Pada algoritma *bee colony optimation* didapat bahwa percobaan ke-7 memiliki waktu tempuh tersingkat yaitu 15,01 detik.

Pada algoritma *ant colony optimation* (grafik berwarna ungu) terdapat 5 kenaikan waktu tempuh dengan rataan kenaikan 1,12 detik. Hal tersebut terjadi dikarenakan posisi dari *player* atau target dari algoritma yang tidak berdiam di tempat yang mengakibatkan terdapat peningkatan proses iterasi dari algoritma. Pada algoritma *ant colony optimation* didapat bahwa percobaan ke-1 memiliki waktu tempuh tersingkat yaitu 15,97 detik.

Penggunaan parameter waktu tempuh dalam pengujian penelitian ini memiliki tujuan agar lebih mudah membandingkan tingkta keefektifan antara algoritma yang digunakan dalam penelitian ini.

4.1.3.2 Waktu tempuh NPC menemukan rute terhadap *player* (*Obstacle)*

Gambar 4.8 Layouting Awal Posisi NPC dan Player Parameter 2

Pada gambar 4.8 terdapat posisi awal dari NPC dan *player*. Posisi tersebut akan menjadi posisi yang akan digunakan disetiap algortima pada pengujian parameter pengukuran yang kedua. Namun pada proses pengujian, *player* akan berpatroli secara random atau bergerak secara otomatis ke arah ruang gerak yang diberikan.

Pada parameter pengukuran yang kedua akan dilakukan pengujian pada waktu tempuh NPC menemukan rute terhadap *player* dengan penambahan *game object obstacle*. Nantinya akan dilakukan 10 kali percobaan pada masing-masing algoritma. Pada tabel 4.2 akan dijabarkan perbandingan waktu tempuh dari hasil pengukuran parameter yang kedua dari setiap algoritma di masing masing NPC.

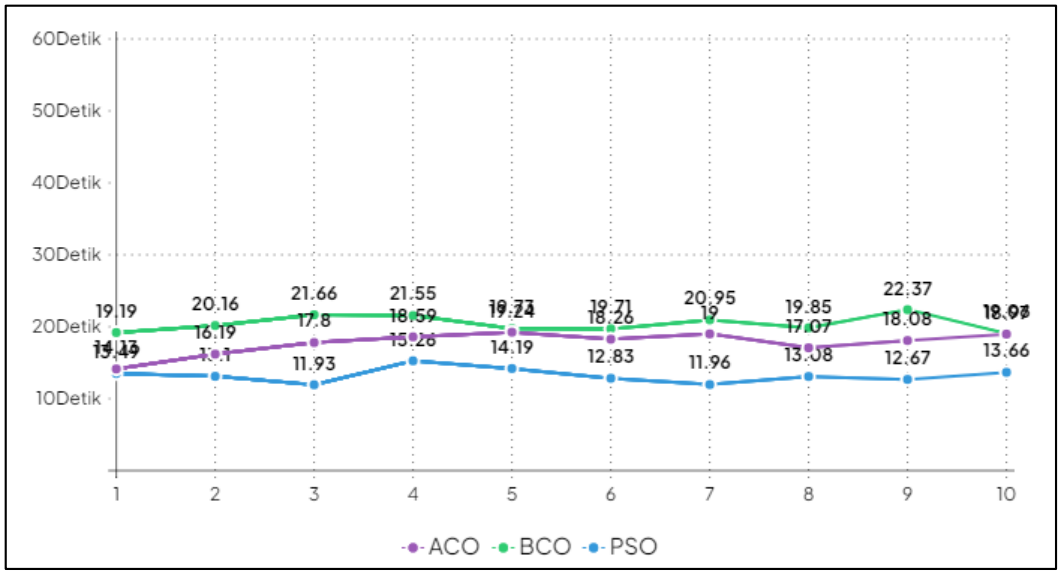
Tabel 4.2 Perbandingan Hasil Waktu Ber-*Obstale*

Pengujian	***Particle swarm optimation***	***Bee colony optimation***	***Ant Colony optimation***
Pengujian ke 1	13,49 detik	19,19 detik	**14,13 detik**
Pengujian ke 2	13,10 detik	20,16 detik	16,19 detik
Pengujian ke 3	**11,93 detik**	21,66 detik	17,80 detik
Pengujian ke 4	15,26 detik	21,55 detik	18,59 detik
Pengujian ke 5	14,19 detik	19,73 detik	19,24 detik

Tabel 4.2 Lanjutan

Pengujian	*Particle swarm optimation*	*Bee colony optimation*	*Ant Colony optimation*
Pengujian ke 6	12,83 detik	19,71 detik	18,26 detik
Pengujian ke 7	11,96 detik	20,95 detik	19,00 detik
Pengujian ke 8	13,08 detik	19,85 detik	17,07 detik
Pengujian ke 9	12,67 detik	22,37 detik	18,08 detik
Pengujian ke 10	13,66 detik	**19,06 detik**	18,97 detik
Rata-rata	**13,21 detik**	**20,42 detik**	**17,73 detik**

Tabel 4.2 menujukkan hasil rata-rata perbandingan waktu tempuh dari algoritma *single objective particle swarm optimation*, *bee colony optimation*, dan *ant colony optimation*, yang dimana *single objective particle swarm optimation* memiliki waktu tempuh dengan rata-rata 13,21 detik pada pengukuran menggunakan parameter yang kedua ini.



Gambar 4.9 Grafik Perbandingan Waktu Tempuh Parameter 2

Pada gambar 4.9 terdapat grafik perbandingan dari algoritma *single objective particle swarm optimation* (grafik berwarna biru), *bee colony optimation* (grafik warna hijau) dan *ant colony optimation* (grafik berwarna ungu). Terlihar beberapa perbedaan dari grafik pada gambar 4.8 yang dihasilkan Sehingga Pada

algoritma *single objective particle swarm optimation* (grafik berwarna biru) terdapat 3 kenaikan waktu tempuh dengan rataan kenaikan 1,81 detik. Hal tersebut terjadi dikarenakan posisi dari *player* atau target dari algoritma yang tidak berdiam di tempat, ditambah dengan adanya *obstacle* pada ruang NPC dengan *player* menambah proses iterasi dari algoritma harus ditingkatkan. Pada algoritma *single objective particle swarm optimation* didapat bahwa percobaan ke-3 memiliki waktu tempuh tersingkat yaitu 11,93 detik.

Pada algoritma *bee colony optimation* (grafik warna hijau) terdapat 4 kenaikan waktu tempuh dengan rataan kenaikan 1,55 detik. Hal tersebut terjadi dikarenakan posisi dari *player* atau target dari algoritma yang tidak berdiam di tempat, ditambah dengan adanya *obstacle* pada ruang NPC dengan *player* menambah proses iterasi dari algoritma harus ditingkatkan. Pada algoritma *bee colony optimation* didapat bahwa percobaan ke-10 memiliki waktu tempuh tersingkat yaitu 19,06 detik.

Pada algoritma *ant colony optimation* (grafik berwarna ungu) terdapat 7 kenaikan waktu tempuh dengan rataan kenaikan 1,10 detik. Hal tersebut terjadi dikarenakan posisi dari *player* atau target dari algoritma yang tidak berdiam di tempat, ditambah dengan adanya *obstacle* pada ruang NPC dengan *player* menambah proses iterasi dari algoritma harus ditingkatkan. Pada algoritma *ant colony optimation* didapat bahwa percobaan ke-1 memiliki waktu tempuh tersingkat yaitu 14,13 detik.

4.1.3.3 Pemilihan rute NPC yang statis (sama) atau dinamis (tidak sama)

Pada parameter pengukuran yang ketiga memiliki tujuan untuk mengidentifikasi perbedaan dari algoritma *optimation* dengan algoritma *pathfinding* dalam menentukan atau memilih rute optimal NPC terhadap *player*. Pada parameter pengukuran yang ketiga ini akan dilakukan pengujian pada rute yang digunakan NPC untuk menemukan *player* apakah statis (sama) atau dinamus (tidak sama). Nantinya akan dilakukan 5 kali percobaan pada masing-masing algoritma.

Tabel 4.3 Perbandingan Rute yang Digunakan *Single Objective* PSO

Pengujian	***Particle swarm optimation***	***Contoh Gambaran***
Pengujian ke 1	-	
Pengujian ke 2	Rute tidak sama	
Pengujian ke 3	Rute tidak sama	

Tabel 4.3 Lanjutan

Pengujian	***Particle swarm optimation***	***Contoh Gambaran***
Pengujian ke 4	Rute tidak sama	
Pengujian ke 5	Rute tidak sama	

Tabel 4.3 merupakan hasil *visual* dari pengujian algoritma *single objective particle swarm optimation.* Didapat dari pengujian ke-1 dan pengujian ke-5 pada algoritma *single objective particle swarm optimation* berdasarkan tabel 4.3 bahwasanya rute yang digunakan/dipilih adalah dinamis (**tidak sama**).

Tabel 4.4 Perbandingan Rute yang Digunakan *Bee Colony Optimation*

Pengujian	***Bee Colony optimation***	***Contoh Gambaran***
Pengujian ke 1	Rute tidak sama	

Tabel 4.4 Lanjutan

Pengujian	***Bee Colony optimation***	***Contoh Gambaran***
Pengujian ke 2	Rute tidak sama	
Pengujian ke 3	Rute tidak sama	
Pengujian ke 4	Rute tidak sama	
Pengujian ke 5	Rute tidak sama	

Tabel 4.4 dan merupakan hasil *visual* dari pengujian algoritma *bee colony optimation.* Didapat dari pengujian ke-1 dan pengujian ke-5 pada algoritma *bee*

colony optimation berdasarkan tabel 4.4 bahwasanya rute yang digunakan/dipilih adalah dinamis (**tidak sama**).

Tabel 4.5 Perbandingan Rute yang Digunakan *Ant Colony Optimation*

Pengujian	***Ant Colony optimation***	***Contoh Gambaran***
Pengujian ke 1	Rute tidak sama	
Pengujian ke 2	Rute tidak sama	
Pengujian ke 3	Rute tidak sama	
Pengujian ke 4	Rute tidak sama	

Tabel 4.5 Lanjutan

Pengujian	***Ant Colony optimation***	***Contoh Gambaran***
Pengujian ke 5	Rute tidak sama	

Tabel 4.5 merupakan hasil *visual* dari pengujian algoritma *ant colony optimation.* Didapat dari pengujian ke-1 dan pengujian ke-5 pada algoritma *ant colony optimation* berdasarkan tabel 4.9 dan tabel 4.10 bahwasanya rute yang digunakan/dipilih adalah dinamis (**tidak sama**).

Tabel 4.3 dan 4.5 menujukkan hasil perbandingan parameter ketiga pengujian tentang pemilihan rute NPC yang statis (sama) atau dinamis (tidak sama) dari algoritma *single objective particle swarm optimation*, *bee colony optimation,* dan *ant colony optimation.* Didapat bahwa pada percobaan yang pertama tidak dapat menjabarkan hasil dari hasil parameter 3 dikarenakan tidak ada perbandingan dengan percobaan sebelumnya tentang rute yang ditempuh oleh 3 algortima yang digunakan, namun pada percobaan kedua sampai dengan percobaan ke sepuluh, didapat hasil dari 3 algoritma yang digunakan yaitu rute yang digunakan/dipilih adalah dinamis (**tidak sama).** Hal ini dikarenakan bahwa pada algoritma *optimation* terdapat banyak parameter yang harus dipertimbangkan, berbeda halnya dengan algoritma *pathfinding* yang fokus pada penentuan rute tercepat dan terpendek dari titik A ke titik B tanpa memiliki banyak parameter yang harus dipertimbangkan.

4.2 Integrasi dengan Islam

Penentuan rute optimal dalam penelitian ini bertujuan agar pengguna dapat menikmati game yang lebih menantang dan kompetitif. Sehingga NPC dapat bergerak dan perpindah posisi untuk mencapai tujuan yang lebih efektif. Hal ini juga sesuai dengan ayat Al-Quran mengajarkan pentingnya usaha untuk mencari jalan yang benar atau rute yang optimal dalam kehidupan, seperti yang tertera dalam Surat Al-Isra ayat 57:

اُولٰۤىِٕكَ الَّذِيْنَ يَدْعُوْنَ يَبْتَغُوْنَ اِلٰى رَبِّهِمُ الْوَسِيْلَةَ اَيُّهُمْ اَقْرَبُ وَيَرْجُوْنَ رَحْمَتَهٗ وَيَخَافُوْنَ عَذَابَهٗ ۗ اِنَّ عَذَابَ رَبِّكَ كَانَ مَحْذُوْرًا

"Orang-orang yang mereka seru itu, mereka sendiri mencari jalan kepada Tuhan mereka siapa di antara mereka yang lebih dekat (kepada Allah) dan mengharapkan rahmat-Nya dan takut akan azab-Nya; sesungguhnya azab Tuhanmu adalah suatu yang (harus) ditakuti" (Q.S. Al-Isra [17]: 57).

Menurut tafsir aljalalain oleh Jalaluddin al-Mahalli dan Jalaluddin as-Suyuthi, menjelaskan bahwa individu yang mereka seru sebagai tuhan-tuhan sesembahan mereka sendiri mencari jalan untuk mendekat kepada Rabb mereka melalui ketaatan kepada-Nya. Pertanyaannya, siapakah di antara mereka yang mencari jalan yang lebih dekat kepada Allah? Mengapa mereka mencari jalan tersebut kepada selain-Nya? Sementara itu, mereka juga mengharapkan rahmat-Nya dan merasa takut akan azab-Nya, persis seperti orang-orang selain mereka. Oleh karena itu, mengapa kalian menganggap mereka sebagai tuhan-tuhan? Harus diingat bahwa azab dari Rabbmu adalah sesuatu yang harus ditakuti.

Menurut Tafsir Al-Wajiz oleh Syaikh Prof. Dr. Wahbah az-Zuhaili, Seorang ahli fiqih dan tafsir yang berasal dari Suriah menjelaskan bahwa orang-orang yang disembah oleh orang-orang musyrik dan dianggap sebagai tuhan selain Allah,

seperti malaikat dan Isa, sebenarnya mencari hal-hal yang mendekatkan mereka kepada Allah melalui perbuatan ketaatan dan ibadah. Mereka juga berupaya mencari cara agar dapat semakin dekat dengan Allah. Pertanyaannya, bagaimana mungkin mereka bisa berada dalam keadaan yang jauh dari-Nya? Mereka juga mengharapkan rahmat dari Tuhannya dan merasa takut akan azab-Nya, sama seperti hamba-hamba lainnya. Jadi, bagaimana mungkin kalian menganggap mereka sebagai Tuhan? Sejatinya, azab dari Tuhanmu telah diperingatkan bagi setiap individu (Al-Wajiz, 2023).

Menurut tafsir Syaikh Dr. Shalih bin Abdullah bin Humaid (Imam Masjidil Haram) dijelaskan bahwa, Orang-orang yang mereka seru dan sembah, baik dari kalangan Malaikat maupun yang lainnya, secara aktif mencari cara untuk mendekatkan diri kepada Allah melalui pelaksanaan amal saleh. Mereka bahkan saling berkompetisi dalam upaya menjadi yang paling dekat dengan-Nya melalui berbagai perbuatan ketaatan. Mereka sangat mengharapkan rahmat-Nya dan merasa takut akan azab-Nya. Oleh karena itu, wahai Rasul, sungguh penting untuk menjauhi azab Tuhan (Al-Mukhtashar, 2023).

Menurut tafsir lengkap kemenag oleh kementrian agama RI menjelasan tersebut mengungkapkan bahwa orang-orang yang mereka seru, yang dianggap sebagai tuhan selain Allah, secara aktif mencari jalan menuju Tuhan mereka dan berkompetisi untuk menjadi yang lebih dekat dengan-Nya. Mereka selalu mengharapkan rahmat-Nya dan merasa takut akan azab-Nya. Sungguh, azab dari Tuhanmu merupakan sesuatu yang patut ditakuti oleh semua makhluk-Nya (RI, 2023).

Menurut tafsir almuyassar oleh kementrian agama Saudi Arabia dijelasan tersebut mengindikasikan bahwa orang-orang yang disembah bersama Allah oleh kaum musyrikin, seperti para nabi, orang-orang shalih, dan para malaikat, justru berlomba-lomba dalam upaya mendekatkan diri kepada tuhan mereka melalui amal-amal shalih yang mereka lakukan. Mereka dengan tulus mengharapkan rahmat-Nya dan merasa takut terhadap siksaan-Nya. Sesungguhnya, siksaan dari tuhanmu adalah sesuatu yang wajar untuk dikhawatirkan dan ditakuti oleh para hamba (Al-Muyassar, 2023).

BAB V

KESIMPULAN DAN SARAN

5.1 Kesimpulan

Berdasarkan implementasi dan hasil pengujian didapatkan bahwa:

a. Pada pengujian dengan parameter pertama tentang waktu tempuh NPC menemukan rute terhadap *player* didapatkan bahwa algoritma *single objective particle swarm optimation* memiliki waktu tempuh rata rata 14,18 detik lebih singkat dibanding algoritma *bee colony optimation* dan *ant colony optimation.*
b. Pada pengujian dengan parameter kedua tentang waktu tempuh NPC menemukan rute yang ber-*obstacle* terhadap *player* didapatkan bahwa algoritma *single objective particle swarm optimation* memiliki waktu tempuh rata rata 13,21 detik lebih singkat dibanding algoritma *bee colony optimation* dan *ant colony optimation.*
c. Pada pengujian dengan parameter kedua tentang pemilihan rute NPC yang statis (sama) atau dinamis (tidak sama) didapatkan bahwa algoritma *single objective particle swarm optimation*, *bee colony optimation* dan *ant colony optimation* mendapatkan hasil yang sama persis yaitu rute yang digunakan/dipilih adalah dinamis (**tidak sama).**

Dari hasil pengujian ketiga parameter dapat disimpulkan bahwa algoritma *single objective particle swarm optimation* mendapatkan hasil paling efektif dibandingkan dengan algoritma *bee colony optimation* dan *ant colony optimation* dalam penentuan rute optimal NPC terhadap *player*.

5.2 Saran

Berdasarkan hasil penelitian yang telah dilakukan, terdapat beberapa rekomendasi yang dapat diambil agar aplikasi serupa di masa depan dapat meningkatkan kualitasnya. Rekomendasi tersebut mencakup:

a. Mencoba untuk menggabungkan antara tipe algoritma *single objective particle swarm optimation* dan *multi objective particle swarm optimation*, agar dapat mempersingkat waktu pengolahan algoritma dalam menentukan fungsi heuristik.
b. Mengembangkan kasus menjadi *multi objective* agar dapat diketahui perbedaan dari algoritma *single objective* dalam penentuan rute NPC sehingga dapat dibandingkan penggunaan tipe algoritma yang mana yang lebih baik.
c. Menambah jumlah iterasi dari setiap algoritma yang diteliti dan pembanding agar proses penentuan fungsi heuristik dari setiap algoritma bisa lebih maksimal.

DAFTAR PUSTAKA

Agustina, C. and Wahyudi, T. (2015) 'Aplikasi Game Pendidikan Berbasis Android Untuk Memperkenalkan Pakaian Adat Indonesia', *Indonesian Journal on Software Engineering Aplikasi*, 1(1), pp. 33–39.

Al-Mukhtashar (2023) *TafsirWeb Surat Al-Isra Ayat 57, TafsirWeb*. Available at: https://tafsirweb.com/4661-surat-al-isra-ayat-57.html (Accessed: 2 February 2023).

Al-Muyassar (2023) *TafsirWeb Surat Al-Isra Ayat 57, TafsirWeb*. Available at: https://tafsirweb.com/4661-surat-al-isra-ayat-57.html (Accessed: 29 March 2023).

Al-Wajiz (2023) *TafsirWeb Surat Al-Isra Ayat 57, TafsirWeb*. Available at: https://tafsirweb.com/4661-surat-al-isra-ayat-57.html (Accessed: 29 March 2023).

Avedon, E. M. and Smith, B. S. (1971) 'The Study of Games', p. 545.

David (2021) 'Using Particle Swarm Optimization as Pathfinding Strategy in a Space with Obstacles'. Available at: http://arxiv.org/abs/2201.07212.

Dill, K. (2020) 'What Is Game AI?', *Game AI Pro*, pp. 32–39. doi: 10.1201/b16725-7.

Harsadi, P., Asmiatun, S. and Putri, A. N. (2021) 'Dynamic Pathfinding for Non-Player Character Follower on Game', *Jurnal Teknik Informatika C.I.T Medicom*, 13(2), pp. 55–63. doi: 10.35335/cit.vol13.2021.68.pp51-58.

Imamum Huda (2011) 'Pengembangan Aplikasi P3K Berbasis SmartPhone Android', (July), pp. 1–7.

Kencana, E. N., Harini, I. and Mayuliana, K. (2017) 'The Performance of Ant System in Solving Multi Traveling Salesmen Problem', *Procedia Computer Science*, 124, pp. 46–52. doi: 10.1016/j.procs.2017.12.128.

Moshinsky, M. (1959) *Ant Colony, Handbook of Approximation Algorithms and Metaheuristics, Nucl. Phys.*

Mu'min, S., Nugroho, M. H. and Susiki, S. M. (2015) 'Pergerakan Otonom Pasukan Berbasis Algoritma Boids Menggunakan Metode Particle Swarm Optimization', *Journal of Animation and Games Studies*, 1(1), pp. 1–16.

Novalia, D., Rahmidani, R. and Tasman, A. (2018) 'Pengaruh Brand Image Dan Brand Trust Terhadap Brand Loyalty Vaseline Hand & Body Lotion Pada Mahasiswa Universitas Negeri Padang', *Jurnal Ecogen*, 1(2), p. 316. doi:

10.24036/jmpe.v1i2.4752.

Nurahman (2020) ‘Evaluasi Performa Algoritma C4.5 dan C4.5 Berbasis PSO untuk Memprediksi Penyakit Diabetes’, *Jurnal E-Komtek (Elektro-Komputer-Teknik)*, 4(1), pp. 30–47. doi: 10.37339/e-komtek.v4i1.230.

Pyke, L. M. and Stark, C. R. (2021) ‘Dynamic Pathfinding for a Swarm Intelligence Based UAV Control Model Using Particle Swarm Optimisation’, *Frontiers in Applied Mathematics and Statistics*, 7(November), pp. 1–16. doi: 10.3389/fams.2021.744955.

RI, K. A. (2023) *TafsirWeb Surat Al-Isra Ayat 57*, *TafsirWeb*. Available at: https://tafsirweb.com/4661-surat-al-isra-ayat-57.html (Accessed: 29 March 2023).

Rizaldi, H. I., Jonemaro, E. M. A. and Akbar, M. A. (2018) ‘Penerapan Particle Swarm Optimization Untuk Balancing Ability Pada Team Battle Game RPG’, *Jurnal Pengembangan Teknologi Informasi dan Ilmu Komputer (J-PTIIK) Universitas Brawijaya*, 2(11), pp. 2548–964. Available at: http://j-ptiik.ub.ac.id.

Santoso, E., Budhi, G. S. and Intan, R. (2017) ‘Pembuatan Game Dengan Menerapkan Metode Decision Tree: UCB1, Untuk Menentukan Pemilihan Strategy Dalam AI.’, *Jurnal Infra*, 5(1), pp. 60–66.

Setiawan, A., Santoso, L. W. and Adipranata, R. (2019) ‘Penerapan Algoritma Particle Swarm Optimization (PSO) untuk Optimisasi Pembangunan Negara dalam Turn Based Strategy Game’, *Jurnal Infra*, 7(1), pp. 249–255.

Sharad Kumar and Mishra, D. P. M. (2019) ‘A Comparative Study of Genetic Algorithm and the Particle Swarm Optimization’, *International Journal of Advanced Scientific and Technical Research*, 8(5), pp. 98–103.

Shrivastava, K. and Kumar, S. (2018) ‘The effectiveness of parameter tuning on ant colony optimization for solving the travelling salesman problem’, *Proceedings - 2018 8th International Conference on Communication Systems and Network Technologies, CSNT 2018*, pp. 78–83. doi: 10.1109/CSNT.2018.8820263.

Siswanto, E. and Suni, A. F. (2021) ‘Aksi Penyerangan Non-Player Character (Npc) Menggunakan Metode Naïve Bayes Pada Shooter Game Attacking Behaviour of Non-Player Character (Npc) Using Naïve Bayes Method in Shooter Game’, 8(6). doi: 10.25126/jtiik.202183804.

Slowik, A. (2011) ‘Particle Swarm Optimization’, *The Industrial Electronics Handbook - Five Volume Set*, pp. 3–4. doi: 10.1007/978-3-319-46173-1_2.

Vanilia Cahya N (2019) ‘Solusi Optimasi Global Kontinu Menggunakan Algoritma Scout Particle Swarm’, *Carbohydrate Polymers*, 6(1), pp. 5–10.

Wu, I. (2023) ‘On Reinforcement Learning for the Game of 2048 Student : Advisor : Hung Guei Submitted to Institute of Computer Science and Engineering College of Computer Science National Yang Ming Chiao Tung University in partial Fulfillment of the Requirements for the’.

Yu, X., Chen, W. and Zhang, X. (2018) ‘An Artificial Bee Colony Algorithm for Solving Constrained Optimization Problems’, *Proceedings of 2018 2nd IEEE Advanced Information Management, Communicates, Electronic and Automation Control Conference, IMCEC 2018*, pp. 2663–2666. doi: 10.1109/IMCEC.2018.8469371.