PERGERAKAN PARTIKEL UNTUK EFEK AURA PADA KARAKTER BERBASIS ALGORITMA BOIDS

SKRIPSI

Oleh:
RASYIQAL FIKRI
NIM. 19650015

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

PERGERAKAN PARTIKEL UNTUK EFEK AURA PADA KARAKTER BERBASIS ALGORITMA BOIDS

SKRIPSI

Oleh:
RASYIQAL FIKRI
NIM. 19650015

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

PERGERAKAN PARTIKEL UNTUK EFEK AURA PADA KARAKTER BERBASIS ALGORITMA BOIDS

SKRIPSI

Oleh :
RASYIQAL FIKRI
NIM. 19650015

Telah Diperiksa dan Disetujui untuk Diuji
Tanggal:

Dosen Pembimbing I	Dosen Pembimbing II
Juniardi Nur Fadila, M.T	Dr. Muhammad Faisal, M.T
NIP. 19920605 201903 1 015	NIP. 19740510 2005011 1 007

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

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

HALAMAN PENGESAHAN

PERGERAKAN PARTIKEL UNTUK EFEK AURA PADA KARAKTER BERBASIS ALGORITMA BOIDS

SKRIPSI

Oleh :
RASYIQAL FIKRI
NIM. 19650015

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	:	Puspa Miladin NSAB M.Kom NIP. 19930828 201903 2 018	( )
Anggota Penguji II	:	Juniardi Nur Fadila, M.T NIP. 19920605 201903 1 015	( )
Anggota Penguji III	:	Dr. Muhammad Faisal, M.T NIP. 19740510 2005011 1 007	( )

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	:	Rasyiqal Fikri
NIM	:	19650015
Fakultas	:	Sains dan Teknologi
Program Studi	:	Teknik Informatika
Judul Skripsi	:	Pergerakan Partikel Untuk Efek Aura Pada Karakter Berbasis Algoritma Boids

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, 21 Juni 2023
Yang membuat peryataan,

Rasyiqal Fikri
NIM. 19650015

MOTTO

"Hal memalukan bukanlah ketika kau jatuh, tetapi ketika kau tidak mau bangkit lagi."

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

Assalamu alaikum, Wr. Wb.

Alhamdulillah, dengan limpahan rasa syukur dan puji kepada Allah SWT atas kesehatan dan petunjuk-Nya, penulis berhasil menyelesaikan skripsi ini. Semoga rahmat dan salam senantiasa tercurah kepada Nabi Muhammad SAW, sebagai pemimpin kita, dan semoga kita termasuk dalam golongan orang-orang yang beriman.

Penulis ingin mengungkapkan rasa terima kasih yang tulus kepada semua pihak yang telah memberikan bantuan dan dukungan dalam penulisan skripsi ini. Oleh karena itu, melalui kesempatan ini, penulis ingin menyampaikan ucapan terima kasih yang sebesar-besarnya kepada :

1. Prof. Dr. M. Zainuddin, M.A., selaku Rektor Universitas Islam Negeri Maulana Malik Ibrahim Malang.
2. Dr. Sri Harini, M.Si, selaku Dekan Fakultas Sains dan Teknologi Universitas Islam Negeri (UIN) Maulana Malik Ibrahim Malang.
3. Bapak Dr. Fachrul Kurniawan, M.MT., selaku Ketua Jurusan dan Bapak Yunifa Miftachul Arif, M.T., selaku Sekretaris Jurusan Teknik Informatika Universitas Islam Negeri Maulana Malik Ibrahim Malang yang senantiasa memberikan dorongan.
4. Juniardi Nur Fadila, M.T, selaku dosen pembimbing I dan Dr. Muhammad Faisal, M.T selaku dosen pembimbing II atas arahan dan bimbingan yang telah diberikan, yang telah membantu penulis dalam menyelesaikan pengerjaan skripsi ini.

5. Segenap dosen Jurusan Teknik Informatika UIN Malang yang dengan penuh dedikasi memberikan arahan dan pengetahuan.
6. Terimakasih kepada Ayah dan Ummi yang sering memberikan semangat dan candaannya, tidak lupa kepada kaka dan adik yang juga membantu memberikan masukan dalam mengerjakan skripsi ini.
7. Terima kasih kepada semua responden penelitian yang telah berpartisipasi dan memberikan kontribusi pendapatnya dengan meluangkan waktunya.
8. Teman-teman Alliance Of Informatic Engineering (ALIEN) 2019 yang bahu membahu dan saling memberikan dukungan dalam perjalanan menghadapi tugas-tugas kuliah sejak awal masuk jurusan hingga menyelesaikan skripsi.
9. Teman-teman Tim Ahlusunnah Wal Jamaah dan *The jungle*. Bisyri, Thoriq, Zulfan, Dicky, Putri, Widya, Anam,Pejon, Sadad, Alfin, Alfin Cipmang, Faiz, Faiz Hilmi, Riduan, Pejon, Andi, Oshi, Nia, Avika, Ratih, Mace Jumilah yang selalu ada dalam mendengarkan keluh kesah kehidupan selama berkuliah, dan ifan anwar yang senantiasa membantu dalam mengerjakan skrpsi ini.
10. Semua pihak yang telah memberikan bantuan dalam menyelesaikan skripsi ini, meskipun tidak dapat disebutkan satu per satu, sangat dihargai dan diucapkan terima kasih.
11. Penulis sendiri dengan tekad dan upaya maksimal telah berusaha agar skripsi ini dapat selesai dan percaya dengan kemampuan sendiri

tanpa harus melihat orang lain.

Penulis menyadari bahwa skripsi ini masih memiliki kekurangan dan jauh dari kesempurnaan. Oleh karena itu, penulis sangat mengapresiasi dan menerima dengan senang hati setiap kritik dan saran yang diberikan. Semoga skripsi ini dapat memberikan manfaat yang berarti.

Wassalamu alaikum, Wr. Wb.

Malang, 21 Juni 2023

Penulis

DAFTAR ISI

DAFTAR GAMBAR

DAFTAR TABEL

ABSTRAK

Fikri, Rasyiqal. 2023. **Pergerakan Partikel Untuk Efek Aura Pada Karakter Berbasis Algoritma Boids.** Skripsi. Jurusan Teknik Informatika Fakultas Sains dan Teknologi Universitas Islam Negeri Maulana Malik Ibrahim Malang. Pembimbing: (I) Juniardi Nur Fadila, M.T, Dr. Muhammad Faisal, S.Kom., M.T.

Kata Kunci: *Boids, Aura, Game*

VFX adalah salah satu teknik yang digunakan untuk meningkatkan pengalaman bermain dalam *video game*. Untuk menciptakan simulasi *visual effect* yang menarik pada karakter dalam *game*, seringkali digunakan partikel sistem yang memiliki pergerakan dinamis. Algoritma boids menjadi pilihan yang tepat karena mampu memberikan perilaku yang fleksibel sesuai dengan lingkungan sekitarnya. Penelitian ini bertujuan untuk mengevaluasi potensi algoritma flocking boids dalam meningkatkan tingkat keimmersivean permainan melalui penerapan efek aura yang menarik pada karakter. Dalam penelitian ini, akan dilakukan dua pengujian berbeda yaitu algoritma boids akan dibandingkan dengan dua algoritma berbeda dan akan diuji tingkat collision, waktu untuk membentuk pola, dan pergerakan partikel. Pengujian yang kedua menggunakan *usability testing*. Didapatkan algoritma boids tidak terjadi collision antara partikel dengan karakter, membutuhkan waktu 1.47-4.66 detik untuk membentuk pola, dan pergerakannya dinamis. Pada *usability testing* didapatkan boids memiliki skor rata-rata 2,8, yang menunjukkan bahwa pengguna cukup puas dengan implementasi tersebut.

ABSTRACT

Fikri, Rasyiqal. 2023. **Particle Movement for Aura Effects on Characters Based on the Boids Algorithm.** Thesis. Department of Informatics Engineering, Faculty of Science and Technology, Maulana Malik Ibrahim State Islamic University Malang. Advisors: (I) Juniardi Nur Fadila, M.T., Dr. Muhammad Faisal, S.Kom., M.T.

VFX is one of the techniques used to enhance the gaming experience in video games. To create compelling visual effects for characters in games, particle systems with dynamic movements are often used. The boids algorithm is a suitable choice as it provides flexible behavior according to the surrounding environment. This study aims to evaluate the potential of the flocking boids algorithm in enhancing the level of game immersion through the application of captivating aura effects on characters. Two different tests will be conducted in this research. Firstly, the boids algorithm will be compared with two different algorithms, and the level of collision, pattern formation time, and particle movement will be examined. The second test involves usability testing. The results indicate that the boids algorithm does not result in any collision between particles and characters, takes 1.47-4.66 seconds to form patterns, and exhibits dynamic movement. In the usability testing, the boids algorithm received an average score of 2.8, indicating that users are sufficiently satisfied with its implementation.

Keywords: Boids, Aura, Game

مستخلص البحث

فكري، راسيقال. 2023. حركة الجسيمات لتأثير الهالة على الشخصية بناءً على خوارزمية بويدس. رسالة بحثية. قسم هندسة الحاسوب، كلية العلوم والت كنولوجيا، جامعة الإسلامية الحكومية مولانا مالك إبراهيم مالانغ. المشرفون(I) :جونياردي نور فاديل، الماجستير في التكنولوجيا، (II) الدكتور محمد فيصل، بكالوريوس في علوم الحاسوب والتكنولوجيا، الماجستير في التكنولوجيا.

الكلمات الرئيسية: **Boids، Aura، لعبة.**

تأثيرات بصرية هي تقنية واحدة تُستخدم لتعزيز تجربة اللعب في ألعاب الفيديو. من أجل إنشاء تأثيرات بصرية مثيرة للاهتمام على الشخصيات في اللعبة ، يتم استخدام نظام الجسيمات الذي يتمتع بحركة ديناميكية. تعتبر خوارزمية Boids خيارًا مناسبًا حيث تستطيع توفير سلوك مرن يتناسب مع البيئة المحيطة. يهدف هذا البحث إلى تقييم إمكانات خوارزمية Boids في تعزيز مستوى التفاعلية للعبة من خلال تطبيق تأثير الهالة الجذاب على الشخصيات. في هذا البحث ، سيتم إجراء اختبارين مختلفين ، حيث سيتم مقارنة خوارزمية Boids مع خوارزميتين مختلفتين وسيتم اختبار مستوى التصادم ، والوقت المستغرق لتشكيل الأنماط ، وحركة الجسيمات. الاختبار الثاني سيشمل اختبار قابلية الاستخدام. تم الحصول على نتائج أن خوارزمية Boids لا تواجه تصادمًا بين الجسيمات والشخصيات ، وتستغرق ما بين 1.47 إلى 4.66 ثانية لتشكيل الأنماط ، وتتمتع بحركة ديناميكية. في اختبار قابلية الاستخدام ، تم الحصول على متوسط درجة 2.8 لخوارزمية Boids ، مما يشير إلى رضا المستخدمين عن التنفيذ.

BAB 1

PENDAHULUAN

1.1 Latar Belakang

Pengembangan teknologi *game* saat ini menuntut permainan yang lebih *immersive*. Salah satu cara untuk mencapai *game* yang *immersive* adalah dengan menambahkan efek visual atau VFX yang menarik dan memukau. *Visual effects* (VFX) sering digunakan dalam *video game*. Keberadaan efek visual (VFX) dalam permainan video sangat penting. Efek tersebut bertujuan untuk mendukung gameplay dan membuat dunia dalam permainan menjadi hidup. Efek tersebut dapat berupa ledakan, daun yang jatuh dari pohon, atau bentuk yang berkedip pada antarmuka pengguna yang memberitahu pemain apa yang harus dilakukan (Nordberg, 2020).

Game bertipe RPG adalah salah satu genre *game* online yang paling menarik dan berkembang pesat, di mana pemain mengambil peran karakter dalam pengaturan fiksi dan bertanggung jawab untuk memainkan peran-peran tersebut dalam sebuah narasi (Zhao et al., 2021). VFX adalah salah satu teknik yang digunakan untuk meningkatkan pengalaman bermain *game* RPG. Teknik yang digunakan dalam membuat suatu efek visual dalam game yaitu dengan menggunakan partikel sistem. Contoh *visual effects* (VFX) pada *game* RPG yaitu efek aura pada karakter.

Efek aura adalah salah satu contoh *visual effects* (VFX) yang dapat digunakan untuk meningkatkan kualitas visual permainan. Aura dapat menjadi unsur daya tarik atau faktor yang meningkatkan imersi dalam *game* bagi pemain

(Triartanto et al., 2019). Istilah imersi diambil dari bahasa Inggris *'to immerse'* yang berarti mencelupkan, menyerap atau melibatkan secara mendalam (Mukodi, 2009). Imersi dalam *game* terjadi ketika pemain benar-benar terlibat dalam pengalaman bermain *game*.

Visual effects (VFX) merupakan suatu proses pengolahan citra visual yang ditingkatkan dengan menggunakan teknologi komputer grafis. Hal ini digunakan ketika animator ingin menggambarkan objek yang sulit ditemukan dalam lingkungan nyata dan harus disimulasikan dalam lingkungan maya (Faqih, 2016). Efek aura, sebagai salah satu contoh dari penggunaan VFX dalam video *game*, memberikan kesan terhadap karakter dalam permainan tampil lebih hidup dan membuat pemain merasa lebih terlibat dalam permainan. Selain itu, efek aura dapat digunakan untuk menambahkan kesan dramatis pada permainan, seperti menunjukkan karakter yang sedang marah atau kesakitan. Dalam beberapa kasus, efek aura juga dapat menambahkan kesan misteri pada karakter, seperti menunjukkan karakter yang memiliki kekuatan supernatural. Penambahan efek aura pada karakter juga dapat meningkatkan kualitas visual permainan secara signifikan dan membuat permainan lebih menyenangkan. Dalam industri video *game*, VFX memiliki peran yang sangat penting dalam menciptakan pengalaman bermain yang lebih menarik dan memuaskan bagi para pemain.

Visual effect Aura dalam *game* dapat membantu meningkatkan kualitas grafis dalam permainan. Aura bisa memberikan kesan kekuatan dan keunikan, serta dapat meningkatkan kualitas visual permainan. Namun, harus diingat bahwa segala sesuatu yang diciptakan oleh Allah SWT bertujuan untuk menjadikannya terasa

indah bagi orang-orang yang memandang. Seperti yang tertulis dalam Surah Al-Hijr ayat ke-16:

وَلَقَدْ جَعَلْنَا فِي ٱلسَّمَآءِ بُرُوجًا وَزَيَّنَّٰهَا لِلنَّٰظِرِينَ ١٦

"Dan sesungguhnya Kami telah menciptakan gugusan bintang-bintang (di langit) dan Kami telah menghiasi langit itu bagi orang-orang yang memandang(nya), " (QS. al-Hijr :16).

Dalam tafsir jalalain menafsirkan (Dan sesungguhnya Kami menciptakan gugusan bintang-bintang di langit) yang berjumlah dua belas, yaitu: Aries, Taurus, Gemini, Cancer, Leo, Virgo, Libra, Scorpio, Sagitarius, Capricorn, Aquarius dan Pisces. Bintang-bintang tersebut merupakan garis-garis peredaran daripada tujuh bintang yang beredar, yaitu Mars mempunyai garis edar pada bintang Aries dan bintang Scorpio; Venus mempunyai garis edar pada bintang Taurus dan bintang Libra; Utarid mempunyai garis edar pada bintang Gemini dan bintang Virgo; bulan mempunyai garis edar pada bintang Cancer; matahari mempunyai garis edar pada bintang Leo; Jupiter mempunyai garis edar pada bintang Sagitarius dan Pisces; Saturnus mempunyai garis edar pada bintang Capricorn dan Aquarius (dan Kami telah menghiasi langit itu) dengan bintang-bintang yang gemerlapan (bagi orang-orang yang memandang.) (Imam Jalaludin Muhammad bin Ahmad Mahalli dan Syaikh Jalaluddin Abdurahman bin Abi Bakar Suyuti, 2010).

Dalam dunia digital modern, VFX merujuk kepada penggunaan CGI yang dibuat menggunakan perpaduan, 2D, 3D, dan *particle system* ataupun dengan efek dinamis yang diaplikasikan dengan objek keras maupun lunak serta partikel (Okun et al., 2020). Dalam beberapa kasus, penerapan *visual effect* dalam *game* terkadang kurang *immersive*.

Particle system sering digunakan dalam membuat suatu efek visual dalam *video game. Particle system* bukanlah sistem yang statis. Seiring berjalannya waktu, partikel dalam sistem ini terus mengubah bentuknya, bergerak secara kontinu, dan partikel baru ditambahkan ke dalam sistem, sementara partikel lama akan menghilang. Setiap partikel dalam sistem memiliki siklus hidup tertentu, termasuk tahap "*produce*", "*activity*" dan "*death*". Setiap parameter yang terkait dengan partikel dikendalikan melalui proses acak. Karena karakteristik dari sistem partikel ini, sistem ini memiliki efek yang baik dalam pembentukan objek "*fuzzy*"(Zhang & Hu, 2017).

Oleh karena itu, algoritma yang tepat harus digunakan untuk menciptakan simulasi *visual effect* yang menarik. Dalam hal ini, diperlukan algoritma yang dinamis untuk membentuk pergerakan sistem partikel yang bergerak secara dinamis. Algoritma *boids* memiliki keunggulan dalam pergerakannya, yaitu berperilaku dinamis yakni mempunyai perilaku yang fleksibel sesuai dengan keadaan lingkungan (Djamaludin, 2016).

Beberapa algoritma yang dapat berperilaku dinamis pada objek dalam sebuah *video game* antara lain algoritma algoritma boids dan *Particle Swarm Optimization* (PSO). Implementasi algorithm *boids* merupakan algoritma yang dapat digunakan dalam simulasi *flocking* dalam sebuah *game*. Algoritma ini dibuat berdasarkan pergerakan kawanan burung, kawanan darat, atau kawanan ikan (Dewi et al., 2020).

Dalam algoritma *boids*, terdapat objek yang disebut *boid* yang berada dalam kerumunan (*flock*). Tiap *boid* dalam kerumunan harus mengetahui posisi dan

kecepatan *boid* di sekitarnya yang disebut tetangga. Setiap kali kerumunan itu bergerak, posisi *boid* akan berubah secara konstan sehingga setiap *boids* harus memperbarui informasi tetangganya selama *game* berlangsung. Algoritma *flocking boid* adalah sebuah algoritma yang digunakan untuk mensimulasikan perilaku kelompok hewan atau serangga.

Terdapat tiga aturan yang harus ada untuk membuat gerakan kelompok (*flocking)* yaitu, *collision avoidance* aturan untuk menghindari benturan dengan tetangga terdekat dalam kerumunan. Kedua, *velocity matching*, yaitu upaya boid untuk menyamakan kecepatan dengan tetangga (*neighbour*) lainnya dalam kerumunan. Ketiga, *flock centering*, yaitu usaha tiap individu *boid* dalam kelompok untuk tetap berdekatan dengan kerumunan terdekat (Reynolds, 1987). Ketiga aturan tersebut harus dipenuhi secara bersama-sama agar gerakan *flocking* bisa terjadi dengan lancar. Hal ini terjadi karena ketiga aturan tersebut membantu individu dalam kelompok untuk saling terhubung dan bekerja sama dalam mencapai tujuan bersama.

Algoritma *boids* adalah salah satu algoritma yang paling sering digunakan dalam simulasi perilaku kerumunan, dan telah digunakan sebagai dasar untuk sebagian besar algoritma yang digunakan dalam simulasi kerumunan (Sajwan et al., 2014). Algoritma ini memungkinkan untuk menciptakan simulasi yang realistis dari perilaku kerumunan, seperti burung yang terbang, ikan yang berenang, atau manusia yang berjalan. Hal ini membuat algoritma *boids* sangat berguna dalam berbagai aplikasi yang memerlukan simulasi perilaku kerumunan yang realistis.

1.2 Pernyataan Masalah

Berdasarkan uraian di atas, rumusan masalah yang didapat adalah seberapa efektif algoritma flocking *boids* untuk menciptakan efek aura pada karakter *game* dalam meningkatkan kualitas visual dalam permainan yang *immersive*?

1.3 Tujuan Penelitian

Berdasarkan rumusan masalah diatas, tujuan dari penelitian ini adalah untuk mengukur tingkat keefektifan algoritma *flocking boids* dalam menciptakan efek aura pada karakter yang *immersive*.

1.4 Batasan Masalah

Untuk mencapai tujuan penelitian yang ditentukan, perlu ditetapkan batasan masalah agar penelitian ini lebih terstruktur dan fokus. Adapun batasan masalah yang akan digunakan adalah sebagai berikut:

a Implementasi *flocking boids algorithm* ini berfokus pada pergerakan partikel aura pada karakter.

b Simulasi aura yang akan dibangun menggunakan *platform* Unity3D.

BAB 2

STUDI PUSTAKA

Dalam studi pustaka akan dibahas penelitian yang telah dilakukan sebelumnya yang akan digunakan sebagai referensi untuk mempermudah pemahaman terkait topik yang terkait dengan penelitian dan membandingkannya dengan penelitian yang sedang dilakukan.

2.1 Penelitian Terkait

Dalam Penelitian dengan judul "*Modified Flocking Algorithm for Optimizing Non-Player Character Movements*". Tujuan dari penelitian ini adalah untuk meningkatkan pergerakan karakter non-player (NPC) dalam game 3D dengan mengimplementasikan modifikasi pada algoritma flocking. Peneliti mengumpulkan data melalui studi pustaka dan observasi, dan kemudian melakukan pengembangan menggunakan metodologi pengembangan multimedia. Pengujian dilakukan menggunakan metode blackbox testing dan menghasilkan hasil yang memuaskan, dengan hanya terjadi 6,6% benturan antara 100 agen dan tingkat frame maksimal 10 fps (Dewi et al., 2020).

Dalam Penelitian dengan judul "Perancangan 3D *Modeling* dan VFX *Water Simulation* Dalam Animasi 3d Berjudul "*Blue & Flash*" oleh. Dalam penelitian ini, dibahas tentang tantangan yang dihadapi dalam pembuatan efek visual (VFX) air seperti air terjun, gelombang, dan percikan pada game dengan genre aksi fantasi RPG menggunakan perangkat lunak Autodesk Maya 2017. Tujuan dari penelitian ini adalah untuk memahami dan mengatasi kendala yang mungkin timbul dalam

proses pembuatan efek visual tersebut, sehingga menghasilkan tampilan visual yang sesuai dengan konsep dan gaya animasi 3D yang diinginkan. Metode pengujian yang digunakan adalah kuesioner, yang digunakan untuk mengukur kinerja model 3D dan simulasi efek air VFX dalam menciptakan lingkungan yang realistis. Hasil dari kuesioner menunjukkan bahwa rata-rata 93,3% responden memberikan penilaian positif terhadap kinerja model 3D dan simulasi VFX air dalam menciptakan lingkungan yang realistis dan tampilan visual yang meyakinkan (Simamora et al., 2019).

Dalam Penelitian dengan judul "Implementasi Algoritma *Boids* dan *Collision Avoidance* Pada Pergerakan NPC Dalam *Game* 2D Berbasis Web". Dalam penelitian ini, tujuannya adalah untuk menguji waktu yang dibutuhkan oleh gerombolan NPC ketika menggunakan algoritma Boids dan Collision Avoidance, serta untuk mengetahui apakah jumlah NPC dapat mempengaruhi performa komputer. Penelitian ini dilakukan dengan membandingkan tiga skenario, yaitu tanpa menggunakan algoritma Boids, hanya menggunakan algoritma Boids, dan menggunakan algoritma Boids dan Collision Avoidance. Hasil penelitian menunjukkan bahwa penggunaan algoritma Boids dapat mengurangi jumlah NPC yang bertabrakan dari 20 menjadi hanya dua dalam beberapa eksperimen. Penggunaan algoritma Boids dan Collision Avoidance juga menghasilkan waktu yang lebih singkat, dengan waktu terendah mencapai 5.390 milidetik. Namun, ketika variabel Separation, Alignment, dan Cohesion ditingkatkan, waktu yang diperlukan meningkat menjadi 12.717 milidetik pada iterasi ke-11. Selain itu, semakin banyak jumlah NPC Boids, performa komputer menjadi lebih rendah,

dengan rata-rata Frame Rate turun dari 58.78 FPS dengan 10 boids menjadi 19.49 FPS ketika jumlah boids ditingkatkan menjadi 150 (Maxmilano, 2020).

Dalam penelitian dengan judul "Penerapan Algoritma *Hybrid Pathfinding* A* dan *Boids* untuk *Game* Pesawat Tempur". Dalam penelitian ini, digunakan kombinasi algoritma pathfinding A* dan Boids dengan tujuan untuk menemukan jalur terbaik dan akurat bagi pesawat yang terbang secara berkelompok. Algoritma A* digunakan untuk mencari jalur terpendek dan menghindari halangan statis, sedangkan algoritma Boids digunakan untuk mencari arah terbang pesawat secara berkelompok dan menghindari halangan dinamis. Hasil penelitian menunjukkan bahwa penggunaan algoritma A* hanya saat grid bersentuhan dengan halangan statis dapat mengurangi waktu eksekusi dan penurunan frame rate (FPS). Sementara itu, penggunaan algoritma Boids secara terus menerus dapat mencegah tabrakan antara pesawat dengan kompleksitas waktu yang rendah. Rata-rata frame rate pada semua peta adalah 75.91 dan rata-rata waktu eksekusi dari semua peta adalah 102.79 milidetik (Prasetyo et al., 2017).

Dalam penelitian dengan judul "*Visual Effects for Mobile Games*". Penelitian ini memiliki dua tujuan utama, yaitu menciptakan efek visual yang bersih dan jelas serta meningkatkan keterampilan dalam menciptakan efek visual dan memahami berbagai teknik penciptaan yang berbeda. Efek visual tersebut dirancang untuk memvisualisasikan kemampuan utama karakter dalam permainan dan menciptakan dasar untuk memahami cara kerja dan penampilan variasi elemen dari kemampuan utama yang sama pada karakter yang berbeda. Proses penciptaan efek visual dilakukan menggunakan mesin permainan Unity dengan memanfaatkan

sistem partikel bawaan Unity, perangkat lunak 3D Blender, dan perangkat lunak manipulasi gambar GIMP. Hasilnya adalah efek visual yang jelas dan mendukung pengalaman bermain permainan (Nordberg, 2020).

Dalam penelitian dengan judul "Implementasi Algoritma RVO sebagai Sistem Kendali Gerombolan NPC pada Permainan Action RPG". Penelitian ini bertujuan untuk menciptakan simulasi pergerakan kelompok NPC pada game bertipe Action RPG yang mirip dengan kondisi aslinya. Dalam penelitian ini menggunakan algoritma RVO (*Reciprocal Velocity Obstacles*), untuk mengatur pergerakan aset agar terlihat lebih alami. Berdasarkan hasil pengujian yang dilakukan, RVO telah terbukti efektif dalam mengoordinasikan pergerakan NPC menjadi lebih baik (Fadila & Arif, 2020).

2.2 Landasan Teori

2.2.1 *Flocking*

Flocking Behavior merupakan sebuah perilaku berkelompok yang dimiliki oleh sekelompok burung atau kawanan burung yang sedang mencari makan ataupun sedang terbang. Sifat atau perilaku ini juga dimiliki oleh kawanan ikan yang disebut *Shoaling* dan pada kawanan hewan darat yang disebut *Swarming* (Brilyanto et al., 2021).

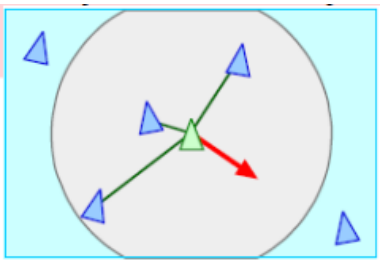
2.2.2 *Algoritma Boids*

Algoritma *Boids* merupakan salah satu metode yang dapat menggambarkan gerak dan perilaku dalam sebuah kerumunan (*flock*). Algoritma ini di kembangkan oleh Craig Reynolds (1987) yang akhirnnya di kembangkan di dalam *game*

komputer. Pada penelitian (Dewi et al., 2020) menyebutkan *rules of boids algorithm* Craig Reynolds dijelaskan sebagai *separation, alignement,* dan *cohesion*.

a. Separation

Aturan *separation* yaitu individu dalam *flock* akan menghindari benturan dengan tetangga terdekat saat berada dalam kerumunan.



Gambar 2.1 *Separation*

Pada gambar 2.1 menjelaskan pergerakan *separation* yaitu kemampuan *boid* untuk mengontrol perilaku dalam menjaga jarak antar *boid* yang berdekatan agar tidak berbenturan. Untuk menghitung *separation* dapat menggunakan perkondisian berikut.

$$d(Px, Pb) \leq d2 \cap d(Px, Pb) \geq d1$$

Pada perkondisian $d(Px, Pb) \leq d2 \cap d(Px, Pb) \geq d$, jika jarak antara boid i (Px) dan tetangga terdekat (Pb) kurang dari atau sama dengan d2 (jarak maksimum) dan jika jarak antara boid i dan tetangga terdekat lebih besar dari atau sama dengan d1 (jarak *minimum*), *Separation* dapat dihitung dengan rumus 2.1 sebagai berikut (Dewi et al., 2020):

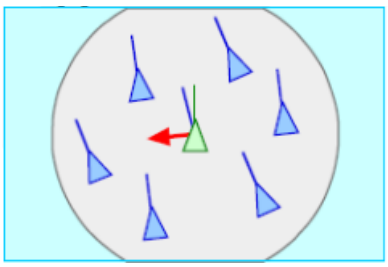
$$\overrightarrow{Vsr} = \sum_{x}^{n} \frac{\overrightarrow{\overrightarrow{Vx} + \overrightarrow{Vb}}}{d(Px, Pb)} \quad (2.1)$$

Keterangan:

$\overrightarrow{Vsr}$	: kecepatan hasil perhitungan *Separation* Px
$\overrightarrow{Vx}$	: kecepatan *boid* yang sedang diproses
Px	: posisi boid yang sedang diproses
Pb	: tetangga terdekat dari boid Px
$\overrightarrow{Vb}$	: kecepatan *boid* lainnya (*boid* tetangga)
n	: jumlah tetangga boid Px dalam radius

b. Alignment

Mengkoordinasikan perilaku boid agar dapat bergerak pada kecepatan yang sama dengan boid yang ada di sekitarnya.



Gambar 2.2 *Alignment*

Pada gambar 2.2 menjelaskan pergerakan *alignment*, yaitu kemampuan untuk mengontrol perilaku individu *boid* untuk menyesuaikan kecepatannya dengan *boids* lain dalam kawanan. Untuk menghitung *alignment* dapat menggunakan perkondisian berikut.

$$d(Px, Pb) \leq d2 \ \cap d(Px, Pb) \geq d1$$

$d(Px, Pb) \leq d2 \ \cap d(Px, Pb) \geq d1$: Jika jarak antara boid i (Px) dan tetangga terdekat (Pb) kurang dari atau sama dengan d2 (jarak maksimum) dan jika jarak antara boid i dan tetangga terdekat lebih besar dari atau sama

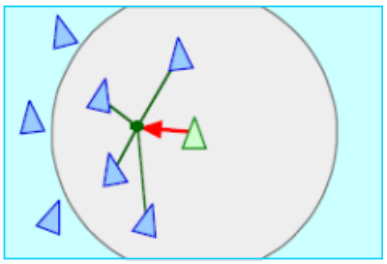
dengan d1 (jarak minimum), *Alignment* dapat dihitung dengan rumus 2.2 sebagai berikut (Dewi et al., 2020):

$$\overrightarrow{Var} = \frac{1}{n}\sum_{x}^{n} \overrightarrow{Vx} \qquad (2.2)$$

Keterangan:
$\overrightarrow{Var}$: nilai rata-rata dari perbedaan kecepatan antara boid.
$\overrightarrow{Vx}$: kecepatan *boid* yang sedang diproses.
n : jumlah tetangga boid dalam radius.

c. *Cohesion*

Cohesion adalah kemampuan untuk mengatur perilaku *boids* agar bergerak menuju posisi rata-rata kawanan terdekat.



Gambar 2.3 *Cohesion*

Pada gambar 2.3 menjelaskan pergerakan *cohesion*, yaitu kemampuan untuk mengontrol perilaku individu *boid* dengan bergerak ke arah atau posisi rata-rata dengan *boids* tetangga. Untuk menghitung *Cohesion* dapat menggunakan perkondisian berikut.

$$d(Px, Pb) \leq d2 \ \cap d(Px, Pb) \geq d1$$

$d(Px, Pb) \leq d2 \ \cap d(Px, Pb) \geq d1$: Jika jarak antara boid i (Px) dan tetangga terdekat (Pb) kurang dari atau sama dengan d2 (jarak maksimum) dan jika jarak antara boid i dan tetangga terdekat lebih besar dari

atau sama dengan d1 (jarak minimum), *Cohesion* dapat dihitung dengan rumus 2.3 sebagai berikut (Dewi et al., 2020):

$$Pavg = \sum_{x}^{n} \frac{Px}{n} \quad (2.3)$$

Keterangan:
Pavg : posisi rata-rata dari semua boid
Px : boid yang sedang diproses
n : jumlah tetangga boid dalam radius

2.2.3 *Particle System*

Sistem partikel adalah sebuah teknik yang digunakan dalam bidang fisika game, motion graphics, dan komputer grafis. Teknik ini melibatkan penggunaan sejumlah besar partikel kecil, seperti sprite, model 3D, atau objek grafis lainnya, untuk mensimulasikan fenomena "fuzzy" yang sulit diproduksi menggunakan teknik rendering konvensional. Fenomena-fenomena tersebut dapat mencakup fenomena alam atau proses yang dipengaruhi oleh reaksi kimia. Dengan menggunakan sistem partikel, kita dapat menciptakan efek yang lebih realistis dan kompleks dalam permainan, animasi, atau grafis komputer. (Reeves, 1983).

Menurut (Zhu et al., 2017) proses pembentukan *particle system* meliputi beberapa hipotesis dan mekanisme, seperti:

a. Hipotesis komposisi partikel: Partikel dalam sistem memiliki jalur yang kontinu atau diskrit, dan bergerak secara konsisten dengan distribusi yang ditentukan dalam ruang dan waktu.

b. Hipotesis hubungan antar partikel: Partikel dalam sistem tidak saling berpotongan dengan objek lain, dan tidak dapat ditembus oleh partikel lain.
c. Hipotesis sifat partikel: Setiap partikel memiliki sejumlah sifat yang tidak abstrak.
d. Mekanisme masa hidup partikel: Setiap partikel memiliki masa hidup.
e. Mekanisme gerak partikel: Partikel selalu bergerak sepanjang hidupnya.
f. Algoritma penggambaran partikel.

2.2.4 *Efek Visual*

VFX atau bisa juga disebut visual effect akan memberikan kesan atau impresi lebih baik kepada pemain (Sudrajat, 2022). Dengan adanya efek-efek visual, VFX dapat membantu meningkatkan kesan dan impresi kepada pemain dari dunia *game*. Baik itu melalui efek api, ledakan, atau partikel, VFX memberikan kesan yang nyata dan membantu membuat *game* terlihat hidup dan hidup. Oleh karena itu, VFX merupakan bagian integral dari dunia *game* dan sangat penting untuk diterapkan dengan baik untuk mencapai pengalaman bermain terbaik bagi pemain.

2.2.5 *Usability Testing*

Usability testing adalah metode evaluasi produk yang melibatkan pengujian langsung terhadap pengguna. Teknik ini digunakan untuk menilai seberapa mudah sebuah produk dapat digunakan oleh pengguna (Yumarlin MZ, 2016). Terdapat lima komponen penting yang perlu diperhatikan.

a *Learnability* (kemudahan pembelajaran), mengukur seberapa mudah pengguna dapat menyelesaikan tugas dasar ketika mereka pertama kali berinteraksi dengan desain.

b *Efficiency* (efisiensi) mengacu pada kecepatan pengguna dalam menyelesaikan tugas setelah mereka menguasai desain tersebut.

c *Memorability* (daya ingat) mengukur sejauh mana pengguna dapat mengingat dan menemukan kembali tindakan dan fitur dari sebuah aplikasi setelah jangka waktu tertentu.

d *Errors* (kesalahan), yang mencakup jumlah kesalahan yang dilakukan oleh pengguna, seberapa serius kesalahan tersebut, dan bagaimana pengguna memperbaikinya.

e *Satisfaction* (kepuasan) melibatkan pencarian desain yang memberikan kepuasan dan kegembiraan bagi pengguna.

2.2.6 *System Usability Scale*

System Usability Scale (SUS) adalah skala usability yang cepat dan sederhana yang dirancang untuk memberikan indikasi keseluruhan tentang usability suatu produk oleh penggunanya. Skala SUS dinilai dalam rentang 0 hingga 100, dengan skor yang lebih tinggi menunjukkan *usability* yang lebih baik (Brooke, 2020).

BAB 3

METODOLOGI PENELITIAN

3.1 Deskripsi *Game*

Visual effect (VFX) aura pada *video game* memang sangat umum digunakan pada *game-game* bertipe *action* RPG. Hal ini dikarenakan efek aura yang memantulkan cahaya dan gerakan yang intens membuat *player* merasa terlibat dalam aksi dan pertarungan yang terjadi pada layar. Efek aura juga menambah estetika dari tampilan *game*, menciptakan suasana yang lebih intens dan dramatis. Selain itu, efek aura juga bisa menunjukkan status karakter, misalnya bila karakter sedang memakai item khusus atau memiliki *special skills*. Aura biasanya digunakan sebagai indikator kuat atau kondisi spesial dari karakter pada video *game*. Aura sering diperlihatkan sebagai efek visual di sekitar karakter, menunjukkan bahwa karakter tersebut memiliki kekuatan atau kondisi spesial tertentu, seperti *godmode, berserkermode,* atau *healingmode*.

Dalam *game action* RPG, aura sering digunakan sebagai mekanisme untuk menunjukkan perbedaan antara karakter dan untuk memberikan kesan dramatis pada pertempuran. Dalam hal ini aura akan membantu *player* dalam memahami status karakter yang digunakan dan membantu *player* untuk mengambil keputusan taktis selama bermain. Dalam *game* ini, desain partikel pada *visual effects* (VFX) Aura digunakan sebagai pemberitahuan bahwa karakter sedang menggunakan *skill* tertentu. Selain itu, Aura juga memberikan penambahan kesan visual yang memperkaya pengalaman bermain *game*. *Particle system* yang digunakan memiliki

bentuk bulat dan dilengkapi dengan efek trailing yang memperkuat efek visual yang dihasilkan.

3.2 Skenario *Game*

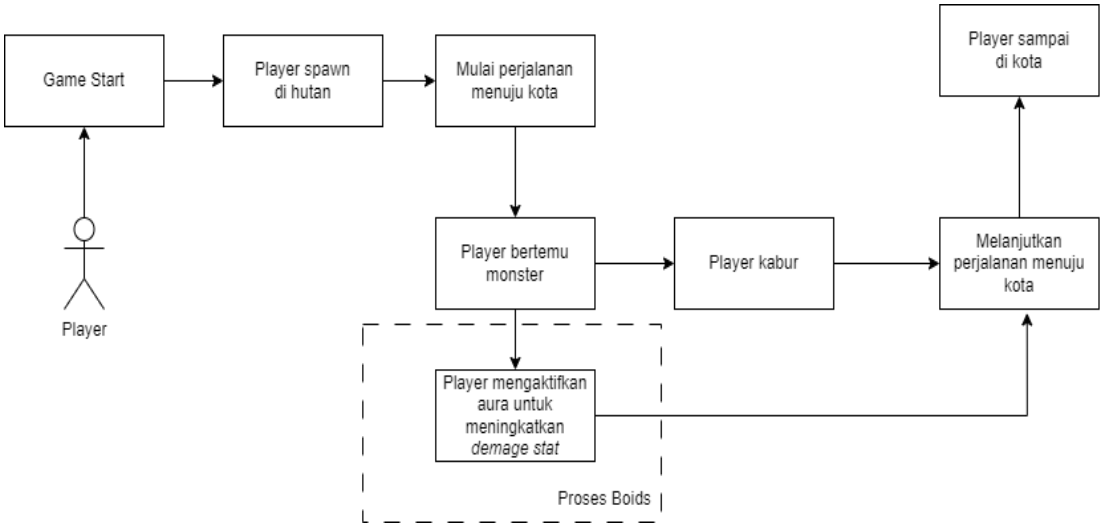
Game yang diteliti berlatar belakang *Game open world* RPG. Pada *game* RPG ini memungkinkan pemain untuk menjelajahi dunia dalam *game* yang luas dan memainkan berbagai karakter yang memiliki kemampuan atau skill yang berbeda-beda. Setiap karakter di dalam *game* memiliki karakteristik dan kemampuan yang unik, seperti elemen, senjata, dan *skill* yang dapat digunakan dalam pertarungan. Setiap karakter dalam *game* ini memiliki kemampuan khusus yang dapat dikembangkan dan digunakan dalam pertempuran. *Player* dapat memilih untuk bekerja sama dengan karakter lain dalam *game*, yang memiliki aura yang berbeda-beda.

Game open world RPG ini memiliki konsep bahwa aura menjadi kekuatan utama karakter, dan player yang memainkan karakter tersebut mampu mengontrolnya. Selama petualangan, karakter akan berinteraksi dengan berbagai jenis musuh dan menyelesaikan berbagai *quest*. Kemampuan karakter dalam mengontrol aura akan membantunya mengatasi berbagai rintangan dan musuh. *Player* dapat memilih untuk meningkatkan kemampuan aura dalam dua bidang utama, yaitu serangan dan pertahanan.

3.3 Desain Sistem

Desain sistem adalah proses mendefinisikan komponen dan tata letak sistem, menentukan bagaimana komponen ini berinteraksi satu sama lain dan

melakukan tugas tertentu, dan menentukan bagaimana sistem dapat diimplementasikan dan dioperasikan. Berikut merupakan diagram blok untuk desain sistem *game*.



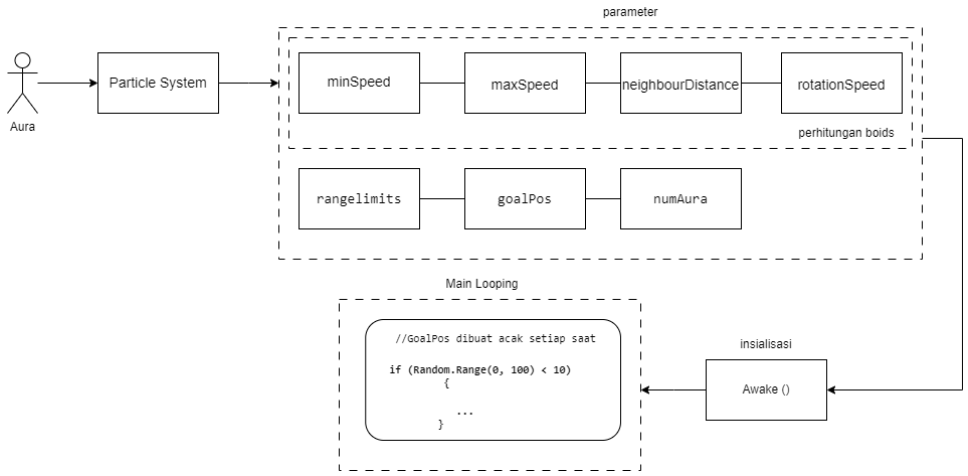
Gambar 3.1 Blok Diagram Desain Sistem Game

Berdasarkan gambar desain sistem pada gambar 3.1, rancangan dari sistem *game* ini dimulai dari *player* memilih start menu, *player* akan langsung diarahkan ke dalam *game*. *Player* akan spawn di dalam hutan dan akan di arahkan untuk menuju sebuah kota yang nantinya *player* akan memulai petualangan yang sesungguhnya. Selama perjalanan menuju kota, *player* akan bertemu dengan monster hutan sebagai NPC yang memiliki tugas untuk menghalangi jalur *player*. NPC ini memiliki kemampuan untuk mengejar *player*. Dalam situasi ini, karakter *player* memiliki skill berupa sistem partikel visual effect aura yang dapat aktif ketika *player* menekan tombol tertentu.

Aura yang dimiliki *player* dapat memberikan dampak besar dalam pertarungan melawan monster. Ini karena Aura tersebut berfungsi untuk

meningkatkan damage status yang diterima oleh monster, sehingga membuat *player* lebih mudah mengalahkan musuh yang ada. Penggunaan aura ini dapat membuat perbedaan besar dalam hasil pertarungan dan membuat pemain lebih unggul dalam menghadapi musuh.

Implementasi *boids* memiliki tujuan untuk menentukan perilaku dan interaksi individu *boids* dengan menggunakan aturan-aturan tertentu. Aturan-aturan ini mencakup konsep alami seperti *alignment, separation,* dan *cohesion*, yang digunakan untuk membuat perilaku burung terlihat alami. Setiap *boid* memiliki kecepatan dan posisi unik dan akan bergerak dengan mengikuti aturan-aturan ini. Kecepatan dan posisi *boid* diperbarui berdasarkan interaksi dengan *boid* lain dan aturan-aturan tertentu. Dengan mengikuti aturan-aturan tersebut, *boid* dapat mencapai keseragaman dalam perilakunya dan membuat tindakan *boid* terlihat alami.

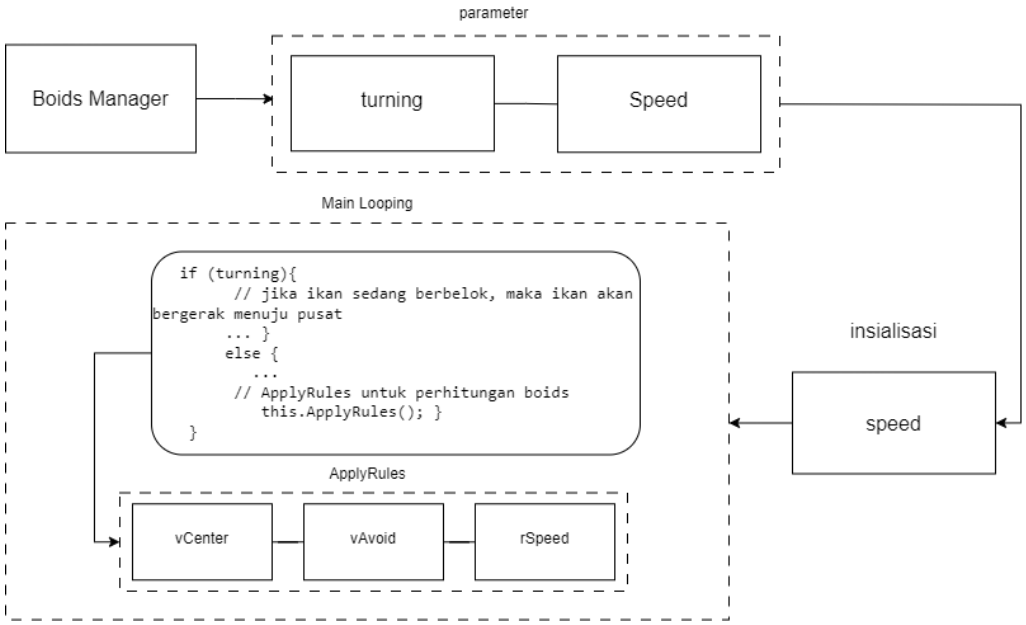


Gambar 3.2 Block Diagram Proses Boids

Dalam implementasi aura dalam game ini, menggunakan sistem partikel yang melibatkan dua kelas berbeda, yaitu kelas "BoidsManager" dan kelas "Boids".

Gambar 3.2 merupakan proses pada "BoidsManager". Kelas "BoidsManager" memiliki tanggung jawab utama dalam mengatur dan mengendalikan perilaku partikel aura secara global. Dengan kelas ini dapat diatur jumlah dan posisi partikel yang dihasilkan Selain itu, kelas ini juga bertanggung jawab dalam menentukan batas gerak partikel dengan mempertimbangkan nilai *rangelimits* yang telah ditentukan sebelumnya.

Selain itu, kelas ini juga mengacak posisi *goalPos* yang akan mengubah tujuan partikel secara acak, memberikan variasi dan keacakan dalam pergerakan partikel dalam simulasi. Dengan demikian, kelas "BoidsManager" memainkan peran penting dalam mengatur dan mengontrol aspek-aspek fundamental dalam perilaku partikel dalam simulasi tersebut.



Gambar 3.3 Blok Diagram Perhitungan Boids

Pada gambar 3.3 merupakan proses dari kelas ‘Boids’, dalam kelas ini akan diatur bagaimana individu partikel aura berinteraksi terhadap tetangganya. Dalam

prosesnya. *speed* diinisialisasi untuk menghitung kecepatan objek pada individu *boids* yang akan diupdate setiap *frame*. Pada proses *main looping*, akan dilakukan pengecekan kondisi apakah objek (*boids*) berada di luar batas. Jika iya, maka objek akan dirotasi menuju titik tengah. Namun, jika objek berada dalam batas lingkup kelompok partikel, maka method *ApplyRules()* akan dipanggil. Method ini bertanggung jawab mencari objek-objek yang menjadi tetangga dari objek yang sedang diproses, kemudian menghitung jarak antara objek tersebut dengan objek tetangganya.

Objek yang sedang diproses akan mengumpulkan informasi tentang kelompok tersebut, seperti posisi tengah kelompok (*vCenter*), kecepatan rata-rata kelompok (*rSpeed*), dan menghindari tabrakan (*vAvoid*). Setelah itu, arah dan kecepatan objek yang sedang diproses akan diubah agar terintegrasi dengan kelompok tersebut. Method ini dijalankan secara periodik untuk mengupdate perilaku objek-objek pada simulasi

3.4 Desain Parameter

Desain parameter digunakan dalam menentukan dan mengatur nilai parameter dalam suatu percobaan atau eksperimen. Hal ini dapat berupa kondisi lingkungan, jumlah sampel, atau hal lain yang digunakan dalam menentukan dan mengatur nilai parameter dalam suatu percobaan atau eksperimen.

Tabel 3. 1 Desain Parameter

Nama Variabel	Keterangan
Speed	*Speed* digunakan untuk menentukan kecepatan *boid* yang berada dalam kelompok.
Turning	Variabel *boolean* yang akan menentukan apakah *boid* sedang berbelok atau tidak.
minSpeed dan *maxSpeed*	Mendefinisikan kecepatan minimum dan maksimum *boid*.

Tabel 3.1 Lanjutan

Nama Variabel	Keterangan
rangelimits	Variabel batas yang menentukan batasan dari titik tengah.
direction	Menentukan arah gerak *boids (Vector 3)*
bounds	Untuk menentukan batas area objek yang berada dalam *flock*.
rotationSpeed	Kecepatan rotasi *boids* yang digunakan untuk menentukan arah *boids* bergerak menuju titik tengah *Boids Manager* ketika objek berada diluar batas.
rSpeed	Kecepatan rata-rata dari sekumpulan *boids*.
mDistance	Variabel yang menentukan jarak antara *boids*.
neighbourDistance	Jarak antar tetangga yang ditentukan dalam class *Flock*Manager
goalPos	Variabel *Vector3* yang menentukan posisi tujuan *boids*.

3.5 Perhitungan manual

Perhitungan manual pada algoritma boids melibatkan langkah-langkah untuk mengatur perilaku pergerakan partikel secara dinamis. Untuk setiap partikel (individu boid), akan dilakukan perhitungan akselerasi berdasarkan tiga prinsip utama: menghindari benturan (*separation*), menyelaraskan arah gerak (*alignment*), dan posisi rata-rata (*cohesion*).

3.5.1 Separation

Tiap individu *boids* akan akan menghindari benturan dengan *boids* lain. Pada algoritma *boids* 3D, dilakukan penentuan arah boid dengan mengurangi jarak antara boid dengan *boid* lain di lingkungannya. Misalkan terdapat tiga *Boids* A, B, dan C. Dengan jarak minimum antar *boids* yaitu dua dan jarak maksimum antar *boids* adalah tiga.

Tabel 3. 2 Inisialisasi Posisi *Boids*

Nama *Boids*	Posisi (x, y, z)	*Velocity*
Boids A	(2,3,4)	(2,5,2)
Boids B	(1,3,2)	(1,3,2)

Tabel 3.2 Lanjutan

Nama *Boids*	**Posisi (x, y, z)**	***Velocity***
Boids C	(4,5,2)	(2,3,4)

Untuk mengetahui separation *Boids*, langkah pertama yaitu menghitung semua jarak antar *Boids* menggunakan rumus Euclidean Distance.

Rumus *euclidean distance*:

$$d\ (p, q) = \sqrt{\sum_{i=1}^{n}(q_i - p_i)^2}$$

Menghitung jarak antar *boids:*

$$d\ (PA, PB) = \sqrt{((2-1)^2 + (3-3)^2 + (4-2)^2)}$$

$$= \sqrt{(1 + 0 + 4)}$$

$$= \sqrt{5} = 2.23$$

$$d\ (PA, PC) = \sqrt{((2-4)^2 + (3-5)^2 + (4-2)^2)}$$

$$= \sqrt{(4 + 4 + 4)}$$

$$= \sqrt{12} = 3{,}46$$

$$d\ (PB, PC) = \sqrt{((1-4)^2 + (3-5)^2 + (2-2)^2)}$$

$$= \sqrt{(9 + 4 + 0)}$$

$$= \sqrt{13} = 3.60$$

Boids akan melakukan perpindahan jika:

$$d(Px, Pb) \leq d2 \cap d(Px, Pb) \geq d1 \rightarrow Vsr$$

Dari perkondisian tersebut jarak antar *boids* yang memenuhi adalah d (PA, PB), yaitu:

$$d\ (PA, PB): 2.23 \leq 3 \cap 2.23 \geq 2$$

Selanjutnya akan dihitung nilai *separation* untuk masing-masing tetangga yang memenuhi perkondisian tersebut. Nilai *separation* dapat dihitung dengan rumus:

$$\overrightarrow{Vsr} = \sum_{x}^{n} \frac{\overrightarrow{\overrightarrow{Vx} + \overrightarrow{Vb}}}{d(Px, Pb)}$$

Menghitung rata-rata kecepatan *boid* $\overrightarrow{\overrightarrow{Va} + \overrightarrow{Vb}}$:

$$\overline{v} = \frac{\overrightarrow{Vx} + \overrightarrow{Vb}}{2} = \frac{\left((2 + 1) \ , (5 + 3) \ , (2 + 2) \ \right)}{2}$$

$$= (1.5, 4, 2)$$

Menghitung kecepatan *Separation* yang baru:

$$\overrightarrow{Vsr} = \sum_{x}^{n} \frac{\overline{v}}{d(Px, Pb)}$$

$$\overrightarrow{Vsr} = \frac{(1.5, \ 4, \ 2)}{\sqrt{5}}$$

$$\overrightarrow{Vsr} = \frac{(1.5, \ 4, \ 2)}{2.23}$$

$$\overrightarrow{Vsr} = (0.67, 1.79, 0.89)$$

Jadi, vektor perpindahan gaya *separation* untuk masing-masing tetangga yang memenuhi syarat yaitu *d* (PA, PB) dengan gaya *separation* $(0.67, 1.79, 0.89)$. Gaya *separation* antara *boids* digunakan untuk untuk mengubah arah *boids*. Gaya *separation* akan mendorong *boids* agar menjauhi *boids* lain yang terlalu dekat dengannya, sehingga *boids* akan menghindari benturan.

3.5.2 Alignment

Alignment adalah salah satu perilaku atau aturan yang digunakan dalam algoritma *boids*. Pada dasarnya, perilaku ini mengatur agar setiap individu *boid* atau agen dalam kelompok untuk mengikuti menyelaraskan arah gerak dengan *boid* disekitarnya. Contoh perhitungan *alignment* pada *boid* 3D, yaitu sebagai berikut:

Diketahui terdapat lima *boids* dalam sebuah kawanan yaitu A, B, C, D, dan E, serta jarak minimum (d1) adalah 2 dan jarak maksimum (d2) adalah 3. Maka tentukan *velocity matching* untuk *boid* A dengan kecepatan awal masing-masing *boid* adalah sebagai berikut:

Tabel 3. 3 Inisialisasi Kecepatan *Boids*

Nama *Boids*	**Posisi**	***Velocity***
Boids A	(2, 3, 4)	(1, 4, 2)
Boids B	(1, 3, 2)	(1, 3, 4)
Boids C	(4, 5, 2)	(1, 1, 2)
Boids D	(1, 3, 4)	(1, 3, 3)
Boids E	(1, 4, 2)	(2, 3, 4)

Untuk menentukan *alignment* pada *Boids* A, perlu menghitung rata-rata jarak dari seluruh *boids* yang ada dalam kawanan tersebut menggunakan rumus *euclidean distance*.

$$d(PA, PB) = \sqrt{((2-1)^2 + (3-3)^2 + (4-2)^2)}$$

$$= \sqrt{(1 + 0 + 4)}$$

$$= \sqrt{5}$$

$$= 2.23$$

$$d(PA, PC) = \sqrt{((2-4)^2 + (3-5)^2 + (4-2)^2)}$$

$$= \sqrt{(4 + 4 + 4)}$$

$$= \sqrt{12} = 3.46$$

$$d\,(PA, PD) = \sqrt{((2-1)^2 + (3-3)^2 + (4-4)^2)}$$

$$= \sqrt{(1 + 0 + 0)}$$

$$= \sqrt{1}$$

$$= 1$$

$$d\,(PA, PE) = \sqrt{((2-1)^2 + (3-4)^2 + (4-2)^2)}$$

$$= \sqrt{(1 + 1 + 4)}$$

$$= \sqrt{6}$$

$$= 2.45$$

Perhitungan *alignment* akan dilakukan dengan syarat jarak antar *boids* $d(Px, Pb) \leq d2 \cap d(Px, Pb) \geq d1$. Dari syarat tersebut jarak antar *boids* yang memenuhi adalah d (PA, PB) dan d (PA, PE), yaitu:

Boid A terhadap boid B: $2.43 \leq 3 \cap 2.43 \geq 2$

Boid A terhadap boid C: $2.45 \leq 3 \cap 2.45 \geq 2$

Langkah selanjutnya yaitu melakukan perhitungan *alignment*, dengan n sebanyak dua karen *boid* A memiliki jarak lebih dekat dengan boid B dan E.

$$\overrightarrow{Var} = \frac{1}{n}\sum_{x}^{n} \overrightarrow{Vx}$$

$$\overrightarrow{Var} = \frac{1}{2}\,(Va + Vb + Vc\,)$$

$$\overrightarrow{Var} = \frac{((1,4,2) + (1,3,4) + (1,1,2))}{2}$$

$$\overrightarrow{Var} = \frac{(3,8,8)}{2} = (1.5, 4, 4)$$

Jadi, nilai *alignment* untuk *boid* B dan C terhadap *boid* A adalah $(1.5, 4, 4)$.

3.5.3 Cohesion

Dalam perilaku *cohesion*, *boids* akan berfokus pada pergerakan menuju posisi rata-rata dari seluruh *boids*. Berikut adalah contoh perhitungan cohesion pada beberapa *boids* dengan posisi yang berbeda. Misalnya, jika ditentukan lima *boids* A, B, C, D, dan E dengan posisi yang berbeda pada masing-masing *boids*, dan jarak minimum antar *boid* (d1) adalah 2 dan jarak maksimum antar boid (d2) adalah 3.

Tabel 3. 4 Inisialisasi Posisi *Boids*

Nama *Boids*	**Posisi (x, y, z)**
Boids A	(2, 3, 4)
Boids B	(1, 3, 2)
Boids C	(4, 5, 2)
Boids D	(1, 3, 4)
Boids E	(1, 4, 2)

Untuk menentukan *cohesion* pada *Boids* D, pertama harus ditentukan jarak boid yang saling berdekatan dengan syarat perkondisian $d(Px, Pb) \leq d2 \cap d(Px, Pb) \geq d1$ menggunakan rumus *Euclidean Distance*.

d (PA, PB) = $\sqrt{((2-1)^2 + (3-3)^2 + (4-2)^2)}$

= $\sqrt{(1 + 0 + 4)}$

= $\sqrt{5}$

= 2.23

d (PA, PC) = $\sqrt{((2-4)^2 + (3-5)^2 + (4-2)^2)}$

= $\sqrt{(4 + 4 + 4)}$

= $\sqrt{12}$

= 3.46

$$d(PA, PD) = \sqrt{((2-1)^2 + (3-3)^2 + (4-4)^2)}$$

$$= \sqrt{(1 + 0 + 0)}$$

$$= \sqrt{1}$$

$$d(PA, PE) = \sqrt{((2-1)^2 + (3-4)^2 + (4-2)^2)}$$

$$= \sqrt{(1 + 1 + 4)}$$

$$= \sqrt{6}$$

$$= 2.45$$

$$d(PB, PC) = \sqrt{((1-4)^2 + (3-5)^2 + (2-2)^2)}$$

$$= \sqrt{(9 + 4 + 0)}$$

$$= \sqrt{13}$$

$$= 3.6$$

$$d(PB, PD) = \sqrt{((1-1)^2 + (3-3)^2 + (2-4)^2)}$$

$$= \sqrt{(0 + 0 + 4)}$$

$$= \sqrt{4}.$$

$$d(PB, PE) = \sqrt{((1-1)^2 + (3-4)^2 + (2-2)^2)}$$

$$= \sqrt{(0 + 1 + 0)}$$

$$= \sqrt{1}$$

$$d(PC, PD) = \sqrt{((4-1)^2 + (5-3)^2 + (2-4)^2)}$$

$$= \sqrt{(9 + 4 + 4)}$$

$$= \sqrt{17}$$

$$= 4.12$$

d (PC, PE) = $\sqrt{((4-1)^2 + (5-4)^2 + (2-2)^2)}$

$= \sqrt{(9 + 1 + 0)}$

$= \sqrt{10}$

$= 3.16$

d (PD, PE) = $\sqrt{((1-1)^2 + (3-4)^2 + (4-2)^2)}$

$= \sqrt{(0 + 1 + 4)}$

$= \sqrt{5}$

$= 2.23$

Perhitungan *cohesion* akan dilakukan dengan syarat perkondisian jarak antar *boids* $d(Px, Pb) \leq d2 \cap d(Px, Pb) \geq d1$. Dari syarat tersebut jarak antar *boids* yang memenuhi adalah d (PA, PB) dan d (PA, PE), d (PD, PE) yaitu:

d (PA, PB) = $2.23 \leq 3 \cap 2.23 \geq 2$

d (PA, PE) = $2.45 \leq 3 \cap 2.45 \geq 2$

d (PD, PE) = $2.23 \leq 3 \cap 2.23 \geq 2$

Terdapat empat *boid* yang saling berdekatan. Untuk menghitung nilai *cohesion* pada *boid* akan menggunakan rumus:

$$Pavg = \sum_{x}^{n} \frac{Px}{n}$$

$$Pavg = \frac{(2,3,4) + (1,3,2) + (1,3,4) + (1,4,2)}{4}$$

$$Pavg = \frac{(5,13,12)}{4}$$

$$Pavg = (1.25, 3.25, 3)$$

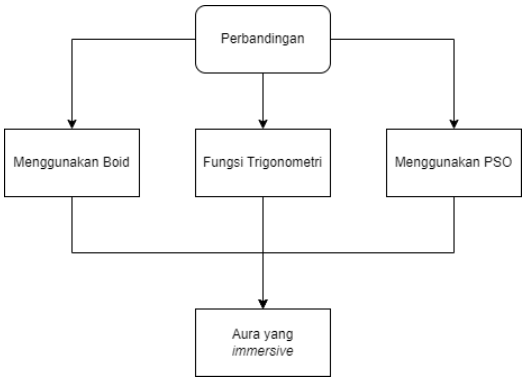
Jadi, posisi rata-rata atau *cohesion* dari *boid* adalah $(1.25, 3.25, 3)$. Dalam implementasi aturan *cohesion*, setiap boid akan bergerak menuju posisi rata-rata (*Pavg*) dari *boids* yang berdekatan dengan kecepatan yang sesuai, sehingga terjadi pengelompokan antara *boids* tersebut.

3.6 Pengujian

Dalam penelitian ini, akan dilakukan dua pengujian. Pengujian pertama akan membandingkan algoritma boids dengan dua metode lain, yaitu fungsi trigonometri dan PSO (*Particle Swarm Optimization*). Pengujian kedua akan menggunakan metode *usability testing* untuk mengevaluasi pengalaman pengguna dalam menggunakan implementasi efek aura dengan menggunakan algoritma boids.

3.6.1 Perbandingan Metode

Untuk mencapai hasil yang valid dan dapat dipertanggungjawabkan, penting untuk menyusun desain percobaan yang teratur dan terstruktur. Desain percobaan melibatkan perencanaan dari suatu eksperimen, termasuk pemilihan pengaturan kondisi yang sesuai dan pemilihan sampel yang akan diamati.



Gambar 3. 4 Desain Pengujian Perbandingan Metode

Pengujian ini bertujuan untuk mengevaluasi hasil implementasi penggunaan algoritma *boids,* fungsi trigonometri, *dan* PSO terhadap efek aura pada karakter. Pengujian ini akan melibatkan perbandingan tingkat collision, waktu pembentukan pola, dan pergerakan partikel antara ketiga metode.

3.6.2 *Usability Testing*

Pada tahap ini, dilakukan pengujian *usability testing* dengan memberikan sejumlah skenario tugas (*taks scenario*) kepada 21 responden dengan kriteria pernah memainkan game bertipe RPG/MMORPG/MOBA. Seluruh responden diharapkan menyelesaikan setiap tugas tersebut. Terdapat 5 macam skenario tugas (*task*) yang telah dijelaskan secara rinci dalam Tabel 3.5.

Tabel 3. 5 Skenario Tugas

No	Tugas *(task)*
1.	Memulai permainan
2.	Menekan tombol '1' untuk mengeluarkan aura *boids* dalam keadaan diam dan bergerak
3	Menekan tombol '2' untuk mengeluarkan aura fungsi trigonometri dalam keadaan diam dan bergerak
4	Menekan tombol '3' untuk mengeluarkan aura PSO dalam keadaan diam dan bergerak
5	Serang monster dengan kombinasi aura dan tombol '*space*' untuk menyerang
6.	Hentikan permainan

Berikut adalah penjelasan dari *taks* skenario yang terdapat pada tabel 3.1:

Task 1. Memulai permainan

User diminta untuk mengklik tombol *play* pada tampilan unity3D, setelah tombol *play* di klik maka *user* dapat memulai permainan.

Task 2. Menekan tombol '1' untuk mengeluarkan aura *boids*

User diminta untuk menekan tombol 1 pada *keyboard* untuk mengeluarkan suatu kemampuan khusus (aura berwarna merah) yang sudah di implementasi algoritma *boids* dalam permainan.

Task 3. Menekan tombol ‘2’ untuk mengeluarkan aura fungsi trigonometri

User diminta untuk menekan tombol 2 pada *keyboard* untuk mengeluarkan suatu kemampuan khusus (aura berwarna ungu) yang sudah di implementasi fungsi trigonometri dalam permainan.

Task 4. Menekan tombol ‘3’ untuk mengeluarkan aura PSO

User diminta untuk menekan tombol 2 pada *keyboard* untuk mengeluarkan suatu kemampuan khusus (aura berwarna biru) yang sudah di implementasi algoritma PSO dalam permainan.

Task 5. Serang monster dengan kombinasi aura dan tombol ‘*space*’ untuk menyerang

User diminta untuk menggunakan aura dan mengkombinasikannya dengan serangan *magic attack* melalui tombol *space*. Tujuannya adalah agar serangan yang dihasilkan menjadi lebih kuat dan efektif terhadap monster.

Task 6. Hentikan permainan

User diminta untuk mengklik menu tombol *esc* pada *keyboard* untuk mengakhiri permainan. *Task* akan dianggap selesai ketika *user* telah keluar dari permainan.

Setelah melakukan *taks* skenario, *user* akan diminta menjawab 10 pertanyaan bertipe *System Usability Scale* (SUS) yang terdapat pada Tabel 3.2.

Tabel 3. 6 Rancangan Pertanyaan Kuesioner

No	Pertanyaan	Kategori
1	Saya rasa saya akan sering menggunakan game ini	*Satisfaction*
2	Saya menemukan bagian dari game yang komplek dan tidak diperlukan.	*Efficiency*
3	Saya pikir game ini mudah digunakan	*Learnability*
4	Saya rasa saya memerlukan bantuan teknisi untuk dapat memainkan game ini	*Error*
5	Saya rasa fungsi dalam game ini saling terintegrasi	*Efficiency*

Tabel 3.6 Lanjutan

No	Pertanyaan	Kategori
6	Saya rasa terlalu banyak ketidakkonsistenan di dalam game ini.	*Error*
7	Saya merasa percaya diri saat memainkan game ini.	*Satisfaction*
8	Saya rasa aura berwarna ungu tidak dapat meningkatkan kejelasan dan keterlibatan saya dalam permainan ini.	*Satisfaction*
9	Saya rasa aura berwarna merah dapat meningkatkan kejelasan dan keterlibatan saya dalam permainan ini.	*Satisfaction*
10	Saya rasa aura berwarna biru tidak dapat meningkatkan kejelasan dan keterlibatan saya dalam permainan ini.	*Satisfaction*

Terdapat lima opsi nilai dalam setiap pertanyaan pada sepuluh pertanyaan di atas, dengan skala angka 1 hingga 5. Angka 1 menunjukkan bahwa responden sangat tidak setuju, sementara angka 5 menunjukkan bahwa responden sangat setuju, seperti yang terlihat pada Tabel 3.8.

Tabel 3. 7 Rentang Skala Penilaian SUS

Nilai	Keterangan
5	Sangat Setuju
4	Setuju
3	Netral
2	Tidak Setuju
1	Sangat Tidak Setuju

Adapun perhitungan SUS sebagai berikut:

a Pertanyaan ganjil: dikurang 1 dari nilai yang diberikan responden.

b Pertanyaan genap: 5 dikurang nilai yang diberikan responden.

c Jumlah nilai yang didapat setelah semua pertanyaan terjawab kemudian dikalikan dengan 2,5 sehingga mendapatkan nilai 0 – 100.

Setelah mendapatkan nilai, hasilnya direpresentasikan dalam bentuk huruf dengan beberapa kategori berikut (Lewis & Sauro, 2018):

Tabel 3. 8 Nilai SUS

Nilai	Skor
≥ 51	F
52-67	D
68-73	C
74-80	B
≤ 80.3	A

BAB 4

IMPLEMENTASI DAN PEMBAHASAN

4.1 Implementasi

Pada bab ini, akan dibahas mengenai implementasi dari rencana yang telah diusulkan. Selain itu, dilakukan pengujian pada aura dari karakter untuk menentukan apakah aura tersebut sudah berjalan sesuai dengan tujuan penelitian yang ingin dicapai.

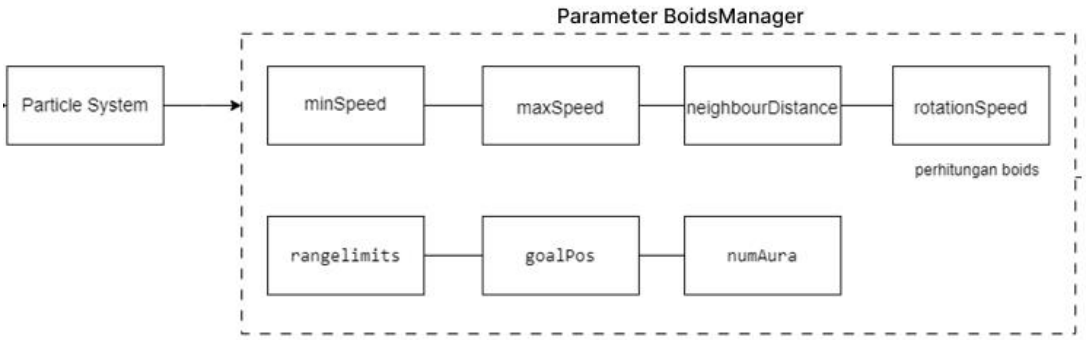
4.1.1 Implementasi *Boids Algorithm* pada Aura

Pada proses implementasi *boids algorithm*, dilakukan pembangunan sistem berdasarkan desain yang telah diusulkan. Tahap ini melibatkan implementasi algoritma *boids* pada *particle system* untuk karakter. Dalam implementasi ini terdapat tiga proses utama yang akan dijabarkan.

Gambar 4. 1 Diagram blok implementasi algoritma boids

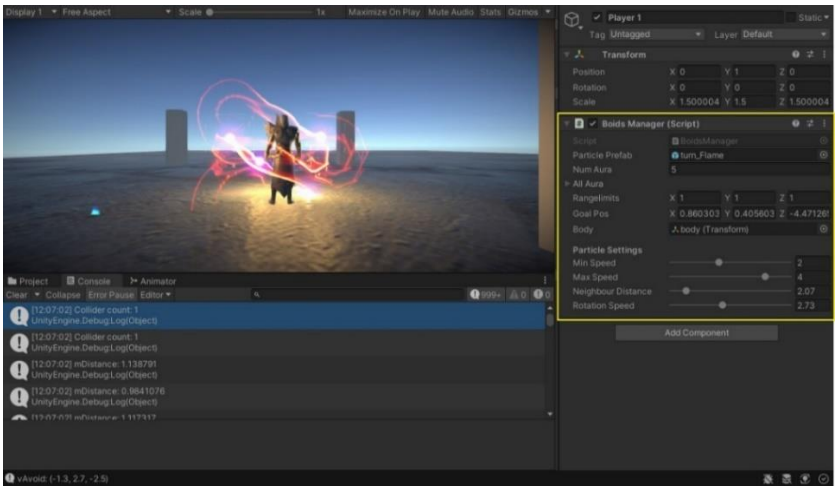
Secara umum implementasi algoritma *boids*, menerapkan tiga langkah utama yaitu *particle system* yang akan digunakan sebagai aura pada karakter diberikan script dengan nama *class* "*BoidsManager*". *Class* "*BoidsManager*" bertanggung jawab mengendalikan dan mengkoordinasikan pergerakan partikel dalam grup atau kelompok. Pada proses ini akan men-*set* beberapa variabel yang

terkait dengan parameter partikel seperti *prefab* objek yang akan digunakan sebagai partikel aura, *rangeLimit* (batas lingkup), kecepatan *minimum* dan *maximum*, jarak antar tetangga, dan kecepatan rotasi. Inisialisasi nilai parameter pada *boidsManager* ini digunakan untuk mengatur dan mengontrol perilaku partikel aura secara berkelompok pada *game* yang akan dibuat seperti pada gambar 4.2.



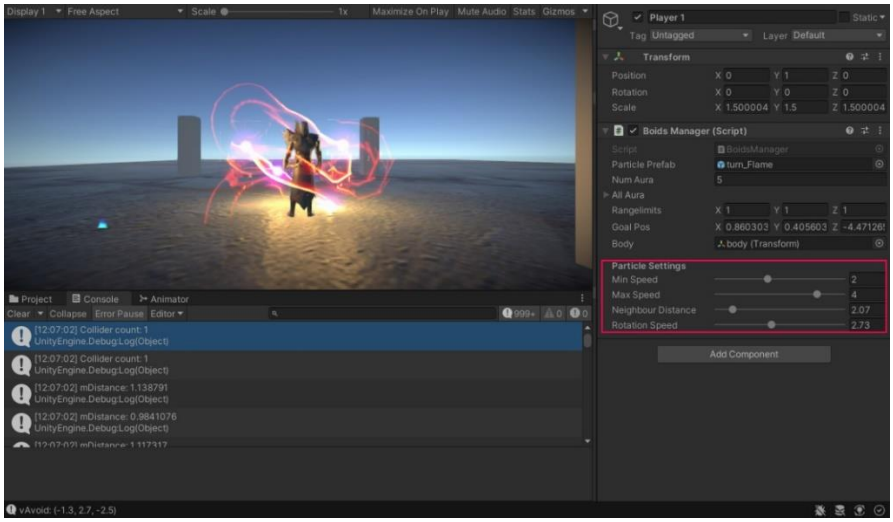
Gambar 4. 2 Proses *BoidsManager*

Variabel-variabel yang sudah diberi nilai dalam *boidsManager* pada gambar 4.3 akan digunakan pada *class boid* untuk mengatur pergerakan partikel aura.



Gambar 4. 3 Inisialisasi *Boids Manager*

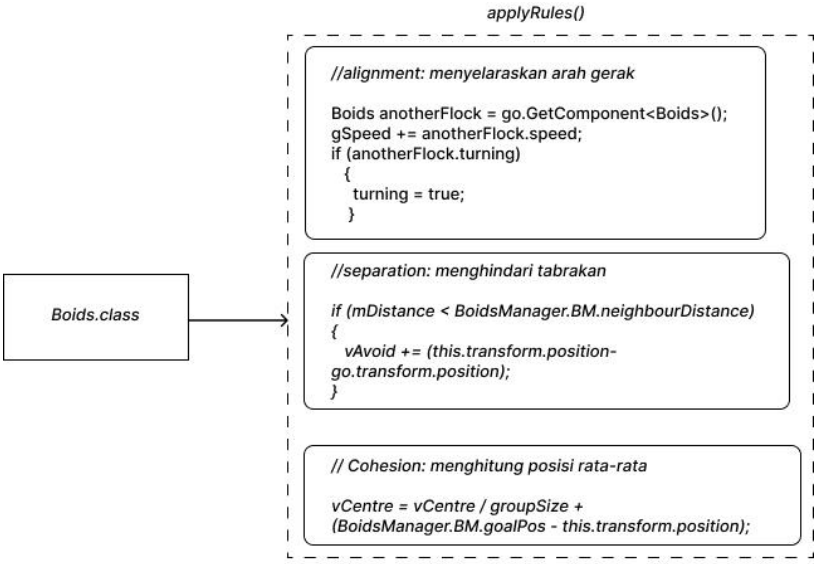
Class Boids bertanggung jawab untuk mengimplementasikan algoritma *Boids* yang mengatur perilaku individu partikel.



Gambar 4. 4 Inisialisasi nilai parameter

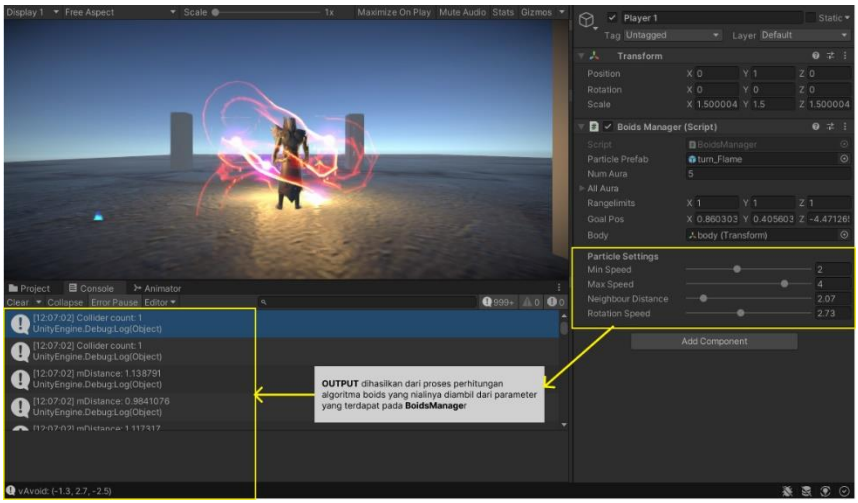
Nilai-nilai seperti *minSpeed*, *maxSpeed*, jarak dan jarak antar tetangga yang telah diinisialisasi dengan nilai acak dalam *BoidsManager* pada gambar 4.4. Kecepatan partikel akan dibatasi oleh nilai minimum dan maksimum yang telah ditentukan, sehingga partikel tidak akan bergerak terlalu lambat atau terlalu cepat. Selain itu, jika jarak antar partikel kurang dari nilai *neighbourDistance,* partikel tersebut akan dianggap sebagai tetangga dan partikel tersebut akan berinteraksi sesuai dengan aturan *boids*, dan akan dilakukan perhitungan *separation*, *alignment*, dan *cohesion*.

Nilai-nilai *separation*, *alignment*, dan *cohesion* didapat dari insialisasi nilai parameter pada *class BoidsManager* yang akan di-inputkan ke dalam perhitungan algoritma boids yang dilakukan dengan memanggil *method applyRules* seperti pada gambar 4.5.



Gambar 4. 5 Proses *Class Boids*

4.1.2 *Variable Output*



Gambar 4. 6 Output perhitungan algoritma boids pada Unity

Hasil dari perhitungan dalam algoritma *boids* akan disimpan pada variabel yang telah diinisialisasi sebelumnya. Kemudian, hasil tersebut akan ditampilkan pada *console unity* agar dapat diamati.

4.1.3 Pengujian Aura

Pengujian ini dilakukan berdasarkan desain pengujian yang telah dijelaskan pada bab tiga. Pengujian akan dilakukan menggunakan perbandingan tiga desain aura berbeda dengan masing masing aura memiliki tiga algoritma yang berbeda, diantaranya yaitu tanpa menggunakan *boids*, algoritma *boids,* dan algoritma *particle swarm optimization* (PSO). Ketiga algoritma tersebut akan melalui tiga jenis pengujian dengan tiga parameter pengukuran yang berbeda yaitu waktu yang dibutuhkan untuk membentuk pola tertentu, kestabilan pola, dan *collision* yang terjadi antara aura dengan objek sekitar. Setiap pengujian pengukuran akan dilakukan dengan keadaan berbeda, yaitu saat karakter sedang dalam posisi diam (*idle*), dan bergerak (*move*).

4.1.3.1 Waktu Yang Dibutuhkan Untuk Membentuk Pola

Pada parameter pengujian yang pertama ini memiliki tujuan untuk mengetahui perbedaan waktu yang dibutuhkan partikel aura untuk membentuk suatu pola yang memutari karakter.

Tabel 4. 1 Waktu Untuk Membentuk Pola Memutar

Pengujian	Algoritma Boids (detik)	Trigonometri (detik)	PSO (detik)
Idle (diam)	01.47	0.13	02.09
Move (bergerak)	04.66	0.23	05.14

Tabel 4.1 menunjukkan hasil perbandingan dari parameter pertama perbedaan waktu yang dibutuhkan partikel aura untuk membentuk suatu pola yang memutari karakter menggunakan algoritma *boids*, fungsi trigonometri, dan *particle swarm optimization* (PSO). Dalam keadaan diam partikel aura yang menggunakan algoritma boids membutuhkan waktu 01.47s untuk membentuk suatu pola

memutar, penggunaan algoritma PSO membutuhkan waktu 02.09s, dan partikel aura yang menggunakan fungsi trigonometri membutuhkan waktu 0.13 detik saat diam dan 0.23 detik saat bergerak. Pada penggunaan algoritma *boids* dan PSO partikel akan di-*spawn* dalam posisi acak sehingga butuh waktu untuk melakukan pergerakan atau pola yang memutari karakter.

Pada aura yang tanpa menggunakan fungsi trigonometri membutuhkan waktu yang lebih cepat dalam membentuk suatu pola. Hal ini terjadi karena setiap partikel sudah diatur posisinya, partikel yang di-*spawn* diposisikan pada posisi yang ditentukan oleh persamaan trigonometri dalam lingkaran dengan radius di sekitar *centerObject.position*. Berbeda dengan algoritma boids dan PSO, dimana posisi partikel akan di-*spawn* secara acak menggunakan fungsi *Random.Range()* pada sumbu X, Y, dan Z berdasarkan *range limits* yang sudah ditentukan.

4.1.3.2 *Collision* Antara Partikel Aura Dengan Karakter

Pada parameter pengujian yang kedua ini memiliki tujuan untuk mengetahui perbandingan banyaknya atau seringnya *collision* yang terjadi antara partikel aura dengan karakter.

Tabel 4. 2 *Collision* yang terjadi antara partikel aura dengan karakter

Pengujian	Algoritma Boids *(fps)*	Trigonometri *(fps)*	PSO *(fps)*
Idle (diam)	0 *collisions*	0 *collisions*	4590 *collisions*
Move (bergerak)	0 *collisions*	0 *collisions*	3387 *collisions*

Tabel 4.2 menunjukkan hasil perbandingan dari parameter kedua yaitu perbandingan banyaknya atau seringnya *collision* yang terjadi antara partikel aura dengan karakter menggunakan algoritma *boids*, fungsi trigonometri, dan *particle*

swarm optimization (PSO). Dari pengujian tersebut dihasilkan penggunaan algoritma *particle swarm optimization* (PSO), menghasilkan 4590 *collisions* dalam keadaan diam dan 3387 *collisions* dalam keadaan bergerak. Hal ini terjadi karena pada algoritma PSO tidak dilakukan perhitungan untuk menghindari tabrakan dengan 'target' yaitu karakter, sehingga partikel dapat bergerak ke arah target tanpa mempertimbangkan jarak antara partikel dengan target.

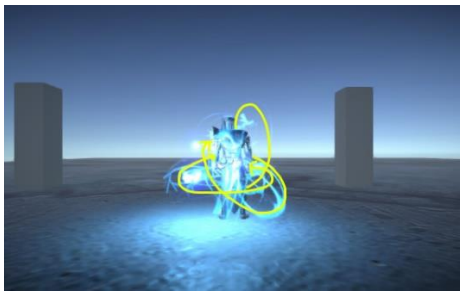
4.1.3.3 Pergerakan Partikel Aura yang Statis atau Dinamis

Pada parameter pengujian yang ketiga memiliki tujuan untuk mengidentifikasi perbedaan pergerakan partikel aura antara algoritma boids, PSO, dan *non-boids.* Pada parameter pengukuran ketiga, akan dilakukan pengujian terhadap gerakan partikel aura apakah bersifat statis atau dinamis. Pada pengujian ini akan dilakukan dalam dua kondisi, yaitu saat *player* dalam keadaan diam (*idle*) dan bergerak (*move*).

Tabel 4. 3 Pergerakan Partikel Aura

Kondisi	Algoritma	Gambar	Pergerakan
Idle (diam)	Boids		Dinamis
Move (bergerak)	Boids		Dinamis

Tabel 4.3 Lanjutan

Kondisi	Algoritma	Gambar	Pergerakan
Idle (diam)	Fungsi Trigonometri		Statis
Move (bergerak)	Fungsi Trigonometri		Statis
Idle (diam)	PSO		Dinamis
Move (bergerak)	PSO		Dinamis

Tabel 4.3, menunjukkan hasil perbandingan dari parameter ketiga yaitu pengujian mengenai pergerakan partikel aura menggunakan algoritma boids, Fungsi Trigonometri, dan *particle swarm optimization* (PSO). Pada percobaan tersebut, ditemukan bahwa saat karakter berada dalam keadaan diam (*idle*), kedua algoritma, yaitu *Boids* dan PSO menunjukkan pergerakan partikel aura yang dinamis, sedangkan aura yang menerapkan fungsi trigonometri bergerak secara

statis. Begitu juga pada saat karakter dalam keadaan bergerak (*move*), algoritma *Boids* dan PSO menunjukkan pergerakan partikel aura yang dinamis, sedangkan aura yang menerapkan fungsi trigonometri bergerak secara statis.

Algoritma *Particle Swarm Optimization* (PSO) pada dasarnya digunakan untuk mencari solusi optimal pada suatu masalah. Pergerakan aura menggunakan PSO bergerak dinamis karena setiap partikel dianggap sebagai agen yang bergerak dalam ruang pencarian (*search space*) dengan tujuan untuk menemukan solusi yang optimal. Oleh karena itu, setiap partikel harus bergerak secara dinamis untuk dapat mengeksplorasi seluruh ruang pencarian dan menemukan solusi yang optimal. Sedangkan pada pergerakan partikel aura yang menggunakan algoritma boids bergerak dinamis dikarenakan, posisi dan orientasi partikel dilakukan perhitungan mengenai pergerakan dan arah gerakan partikel aura yang terdapat pada *boids rules* yaitu *separation*, *cohesion*, dan *alignment*. Partikel diposisikan pada posisi yang berbeda secara acak di dalam batas lingkup yang telah ditentukan, yang membuat gerakan partikel aura terlihat lebih dinamis.

4.2 *Usability Testing*

Hasil dari usability testing diperoleh melalui kuesioner yang diisi oleh 21 responden. Berdasarkan kuesioner tersebut, akan dilakukan analisis dan perhitungan skor implementasi dari ketiga metode, yaitu algoritma boids, fungsi trigonometri, dan PSO.

4.2.1 Analisis Data Hasil *Usability Testing*

Penilaian dari pengujian aplikasi pada pengguna akhir diperoleh melalui proses penelitian yang menggunakan metode *usability testing*. Pada tahap awal, responden akan diberikan serangkaian skenario tugas (*task scenario*) untuk menyelesaikan semua tugas yang diberikan. Setelah tahap tersebut, 21 responden dengan kriteria pernah memainkan game bertipe RPG/MMORPG/MOBA akan diberikan 10 pertanyaan dan dijawab melalui kuesioner.

Setelah itu, responden akan memberikan nilai untuk setiap pertanyaan yang ada, dan kemudian nilai mentah akan diolah menggunakan sistem perhitungan SUS. Nilai yang dihasilkan memiliki variasi yang sangat besar, sehingga menghasilkan sebaran yang luas. Informasi lebih lanjut dapat ditemukan dalam Tabel 4.4 berikut ini.

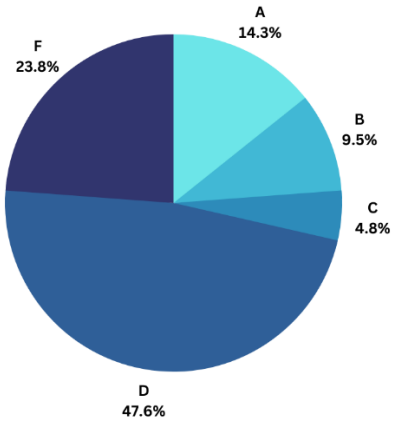
Tabel 4. 4 Hasil survey yang telah diolah ke dalam SUS

No	**SUS Score**										**Jumlah**	**Nilai (Jumlah x 2.5)**	**Nilai (Huruf)**
	Q1	Q2	Q3	Q4	Q5	Q6	Q7	Q8	Q9	Q10			
1	3	0	4	2	3	2	3	1	4	1	23	50	F
2	3	0	4	3	3	3	4	1	3	1	25	52.5	D
3	4	3	4	1	3	3	4	3	3	1	29	50	F
4	3	4	4	3	4	3	4	3	4	4	36	52.5	D
5	3	3	3	3	3	4	4	1	1	3	28	50	F
6	4	4	3	4	3	3	4	3	3	3	34	50	F
7	3	3	4	3	3	3	4	3	4	4	34	35	F
8	4	4	4	4	3	4	4	4	4	4	39	60	D
9	3	3	4	4	3	1	4	3	3	3	31	57.5	D
10	2	1	3	2	3	1	4	1	2	1	20	55	D
11	3	2	3	3	2	3	3	1	1	3	24	60	D
12	3	1	4	3	3	3	3	2	1	1	24	60	D
13	3	3	4	4	3	3	4	3	3	4	34	85	A
14	3	3	4	4	3	3	3	4	4	3	34	85	A
15	3	3	3	3	3	3	3	3	3	3	30	75	B
16	3	2	3	2	3	2	3	1	1	1	21	52.5	D
17	2	3	3	2	3	3	3	1	1	4	25	62.5	D
18	3	3	3	2	3	3	3	2	4	2	28	70	C
19	3	2	4	2	3	3	3	2	3	2	27	67.5	D
20	3	3	3	4	4	4	3	3	4	4	35	87.5	A
21	3	3	4	0	3	3	4	3	4	3	30	75	B

Tabel 4.4 Lanjutan

No	SUS Score										Jumlah	Nilai (Jumlah x 2.5)	Nilai (Huruf)
	Q1	Q2	Q3	Q4	Q5	Q6	Q7	Q8	Q9	Q10			
Nilai rata-rata perolehan SUS											611	72.73	C

Hasil survey yang bervariasi dapat diamati dari nilai (dalam bentuk huruf) yang diperoleh, responden memberikan hasil yang beragam mulai dari F, D, C, B hingga A.



Gambar 4. 7 Sebaran nilai SUS per responden

Gambar 4.7 menunjukkan bahwa sebaran nilai yang diberikan oleh setiap responden berbeda. Responden yang memberikan nilai D memiliki jumlah yang lebih banyak yaitu sekitar 47.6%. Selanjutnya diikuti nilai F 23.8%, A 14.3%, B 9.5% dan C 4.8%. Dari Tabel 4.6 juga telah diketahui hasil SUS secara keseluruhan mendapatkan nilai C dengan besaran 72.73. Hasil ini juga memperkuat kesimpulan bahwasannya usability dari game ini sudah cukup bagus.

4.2.2 Nilai Hasil Implementasi Metode

Dari ketiga metode yang diuji dalam kategori kepuasan kepada 21 responden, diperoleh skor rata-rata penilaian dari hasil *usability testing* sebagai berikut:

Tabel 4. 5 Tabel skor rata-rata penilaian metode

Responden	**Pertanyaan**		
	Q8 (Trigonometri)	**Q9 (Boids)**	**Q10 (PSO)**
Responden Ke- 1	1	4	1
Responden Ke-2	1	3	1
Responden Ke-3	3	3	1
Responden Ke-4	3	4	4
Responden Ke-5	1	1	3
Responden Ke-6	3	3	3
Responden Ke-7	3	4	4
Responden Ke-8	4	4	4
Responden Ke-9	3	3	3
Responden Ke-10	1	2	1
Responden Ke-11	1	1	3
Responden Ke-12	2	1	1
Responden Ke-13	3	3	4
Responden Ke-14	4	4	3
Responden Ke-15	3	3	3
Responden Ke-16	1	1	1
Responden Ke-17	1	1	4
Responden Ke-18	2	4	2
Responden Ke-19	2	3	2
Responden Ke-20	3	4	4
Responden Ke-21	3	4	3
Rata-rata perolehan skor	2.2	2.8	2.6

Pada tabel 4.5 didapatkan hasil dari ketiga metode yang diimplementasi dalam pergerakan aura pada karakter yaitu implementasi algoritma boids mendapatkan skor rata-rata 2.8, implementasi fungsi trigonometri mendapatkan skor rata-rata 2.2, dan implementasi dari PSO mendapatkan skor rata-rata 2.6. Hasil ini juga memperkuat kesimpulan bahwasannya implementasi boids pada aura sudah cukup baik.

4.3 Integrasi Dengan Islam

Tujuan dari menentukan efek visual yang *immersive* ini adalah untuk meningkatkan pengalaman dan keterlibatan pemain dalam permainan. Dengan menggunakan efek visual yang baik, pemain dapat dengan lebih baik memahami peristiwa yang terjadi dalam permainan tersebut. Hal ini akan membantu pemain merasa lebih terlibat dalam permainan dan meningkatkan kepuasan mereka saat bermain game. Hal ini juga sesuai dengan ayat Al-Quran pada surat Al-Hijr Ayat 16 dan Al-mulk ayat 4 untuk menghargai dan menyadari keindahan yang ada di sekitar kita, serta menunjukkan bahwa penglihatan yang digunakan dengan penuh kesadaran akan membawa kita pada pemahaman yang lebih dalam tentang keindahan itu sendiri:

وَلَقَدْ جَعَلْنَا فِي ٱلسَّمَآءِ بُرُوجًا وَزَيَّنَّٰهَا لِلنَّٰظِرِينَ ١٦

"Dan sesungguhnya Kami telah menciptakan gugusan bintang-bintang (di langit) dan Kami telah menghiasi langit itu bagi orang-orang yang memandang(nya), "
(QS. al-Hijr [15]:16).

Ayat tersebut dijelaskan sebagai "Dan sesungguhnya Kami menciptakan gugusan bintang-bintang di langit yang berjumlah dua belas. Gugusan bintang tersebut meliputi Aries, Taurus, Gemini, Cancer, Leo, Virgo, Libra, Scorpio, Sagitarius, Capricorn, Aquarius, dan Pisces. Bintang-bintang tersebut merupakan jalur peredaran dari tujuh bintang yang bergerak. Mars beredar di antara bintang Aries dan Scorpio, Venus beredar di antara bintang Taurus dan Libra, Utarid beredar di antara bintang Gemini dan Virgo, Bulan beredar di bintang Cancer, Matahari beredar di bintang Leo, Jupiter beredar di antara bintang Sagitarius dan Pisces, dan Saturnus beredar di antara bintang Capricorn dan Aquarius. Kami juga

telah menghiasi langit dengan bintang-bintang yang bersinar terang bagi orang-orang yang memandang."(Imam Jalaludin Muhammad bin Ahmad Mahalli dan Syaikh Jalaluddin Abdurahman bin Abi Bakar Suyuti, 2010).

Dalam tafsir Al-Mukhtashar, ayat tersebut dijelaskan sebagai "Sesungguhnya Kami telah menciptakan bintang-bintang besar di langit yang manusia menggunakan sebagai petunjuk dalam perjalanan mereka saat berada dalam kegelapan malam, baik di daratan maupun di lautan. Kami menjadikan bintang-bintang ini sebagai hiasan yang menarik bagi orang-orang yang memperhatikan, agar mereka dapat mengambilnya sebagai bukti akan kekuasaan Allah Subhanahu Wa ta'ala." (Riyadh, 2022).

Dalam tafsir Ash-Shaghir menafsirkan sungguh Kami telah menciptakan gugusan bintang di langit (bintang-bintang yang besar) dan menjadikannya terasa indah bagi orang-orang yang memandang (As-Sariih, 2022).

Dalam tafsir as-Sa'di, ayat tersebut dijelaskan sebagai "Allah berfirman untuk menjelaskan kemuliaan kekuasaan-Nya dan rahmat-Nya kepada ciptaan-Nya, 'Sesungguhnya Kami telah menciptakan gugusan bintang-bintang di langit.' Bintang-bintang ini seperti menara dan simbol yang besar yang digunakan sebagai penunjuk arah dalam kegelapan, baik di daratan maupun di lautan. Dan Kami telah menghiasi langit itu untuk orang-orang yang melihatnya, karena tanpa adanya bintang-bintang, langit tidak akan memiliki pemandangan yang menarik dan bentuk yang mengagumkan. Fenomena ini mengundang orang-orang untuk merenungkan dan memperhatikan makna yang terkandung di dalamnya, serta mencari petunjuk menuju Sang Pencipta." (As-Sa’di, 2022).

Dalam tafsir Ash-Shaghir menafsirkan Sungguh Kami telah menciptakan gugusan bintang di langit (bintang-bintang yang besar) dan menjadikannya terasa indah bagi orang-orang yang memandang (As-Sariih, 2022).

ثُمَّ ارْجِعِ الْبَصَرَ كَرَّتَيْنِ يَنْقَلِبْ اِلَيْكَ الْبَصَرُ خَاسِئًا وَّهُوَ حَسِيْرٌ ٤

"Kemudian ulangi pandangan(mu) sekali lagi (dan) sekali lagi, niscaya pandanganmu akan kembali kepadamu tanpa menemukan cacat dan ia (pandanganmu) dalam keadaan letih. " (QS. Al-Mulk [67]:4).

Dalam tafsir Jalalain, ayat tersebut ditafsirkan sebagai "Kemudian pandanglah sekali lagi, ulangilah kembali penglihatanmu berkali-kali, niscaya penglihatanmu akan kembali kepadamu tanpa cacat atau kekurangan. Penglihatanmu akan kembali keadaan yang rendah, artinya tidak akan menemukan kekurangan atau kecacatan. Penglihatanmu akan kembali dalam keadaan yang buruk, yaitu tidak melihat adanya cacat sama sekali." (Imam Jalaludin Muhammad bin Ahmad Mahalli dan Syaikh Jalaluddin Abdurahman bin Abi Bakar Suyuti, 2010).

Dalam tafsir Al-Wajiz, ayat tersebut dijelaskan sebagai "Kemudian mata berulang kali melihatnya lagi. Mata tersebut menjadi lemah dan tidak mampu melihat kekurangan atau cacat pada ciptaan Allah. Mata tersebut menjadi lelah dan tidak mampu mengetahui apa yang dicari setelah melakukan pencarian yang berulang-ulang." (Az-Zuhaili, 2022).

Dalam tafsir Al-Madinah Al-Munawwarah menafsirkan Allah memerintahkan manusia untuk kembali melihat dan mengamati penciptaan langit dan betapa kokoh dan sempurnanya ia; dan Allah memerintahkan mereka untuk mencari kekurangan yang ada pada makhluk yang luar biasa ini (Ta'dzhim, 2022).

Dalam tafsir An-Nafahat Al-Makkiyah menafsirkan Allah memerintahkan manusia untuk melihatnya bukan hanya cukup melihat sekali saja; Bahkan Allah memerintahkan untuk melihat berkali-kali, agar ia dapat mencari kekurangannya, akan tetapi penglihatannya akan terulang untuk menatap ke langit kembali (setelah ia mencari-cari cacatnya), sambil merasa bosan dan hampa karena sebab gagal (mencari cacat); Karena tidak mungkin akan didapati pertentangan apapun atau celah (Asy-Syawi, 2022).

BAB 5

KESIMPULAN DAN SARAN

Pada bagian penutup ini, akan diuraikan ringkasan hasil penelitian ini beserta rekomendasi yang dapat diberikan kepada pembaca untuk pengembangan penelitian di masa depan.

5.1 Kesimpulan

Berdasarkan hasil dari implementasi dan pengujian yang dilakukan, dihasilkan bahwa:

1. Pada pengujian *collision testing* terhadap algoritma boids pada dua kondisi, yaitu saat karakter bergerak dan saat karakter diam. Hasilnya menunjukkan bahwa tidak ada partikel boid yang mengalami tabrakan dengan karakter dalam kedua kondisi tersebut.
2. Dalam penelitian ini, ditemukan bahwa penggunaan algoritma boids memberikan efek dinamis pada pergerakan aura. Selain itu, waktu yang dibutuhkan untuk melakukan *spawn* untuk membentuk pola aura saat dalam keadaan diam adalah 1.47 detik, sedangkan saat dalam keadaan bergerak membutuhkan waktu 4.66 detik.
3. Dalam penelitian ini, dilakukan uji coba *usability* dengan *usability scale system* (SUS), menghasilkan hasil SUS secara keseluruhan mendapatkan nilai C dengan besaran 72.73 yang artinya game ini sudah cukup baik.

4. Dari hasil implementasi ketiga metode didapatkan implementasi algoritma boids mendapatkan skor rata-rata tertinggi dari ketiga metode yaitu 2.8.

Dari dua pengujian yang telah dilakukan, dapat disimpulkan bahwa implementasi aura menggunakan algoritma boids dalam meningkatkan imersi dalam bermain video game mendapatkan hasil paling efektif.

5.2 Saran

Dalam penelitian yang telah dilakukan, terdapat beberapa kekurangan yang perlu diperbaiki. Untuk peningkatan ke depan, penerapan algoritma boids akan ditingkatkan lebih lanjut, termasuk peningkatan responsivitas partikel saat pemain bergerak dan pergerakan auranya bisa di variasi tidak hanya bergerak memutar dan tidak hanya berbentuk bulatan saja. Di samping itu, untuk pengembangan game ini ke depan, dapat dipertimbangkan untuk menambahkan kondisi partikel saat melakukan serangan. Sebagai contoh, saat menggunakan *skill* menyerang, jumlah partikel aura dapat berkurang.

DAFTAR PUSTAKA

As-Sa'di, S. A. bin N. (2022). *No Title*. Tafsirweb.Com. https://tafsirweb.com/4166-surat-al-hijr-ayat-16.html

As-Sariih, F. bin S. (2022a). *No Title*. Https://Tafsirweb.Com. https://tafsirweb.com/4166-surat-al-hijr-ayat-16.html

As-Sariih, F. bin S. (2022b). *No Title*. Tafsirweb.Com. https://tafsirweb.com/4166-surat-al-hijr-ayat-16.html

Asy-Syawi, S. M. bin S. (2022). *No Title*. Tafsirweb.Com. https://tafsirweb.com/11032-surat-al-mulk-ayat-4.html

Az-Zuhaili, S. P. D. W. (2022). *No Title*. Tafsirweb.Com. https://tafsirweb.com/11032-surat-al-mulk-ayat-4.html

Brilyanto, A., Hutagalung, B., Saputra, R. E., Siswo, A., & Ansori, R. (2021). *Penerapan Flocking Menggunakan Algoritma Boids Application of Flocking Using Boids Algorithm*.

Brooke, J. (2020). SUS: A "Quick and Dirty" Usability Scale. *Usability Evaluation In Industry*, *July*, 207–212. https://doi.org/10.1201/9781498710411-35

Dewi, A. S., Hidayat, E. W., & Sulastri, H. (2020). Modified Flocking Algorithm for Optimizing Non Player Character Movements. *International Journal of Engineering and Emerging Technology*, *5*.

Djamaludin, H. santoso. (2016). *ARTIFICIAL BEE COLONY PADA SIMULASI MENGELILINGI KA ' BAH (THAWAF) SKRIPSI Oleh : HERU SANTOSO DJAMALUDIN JURUSAN TEKNIK INFORMATIKA*.

Fadila, J. N., & Arif, Y. M. (2020). Implementasi Algoritma RVO sebagai Sistem Kendali Gerombolan NPC pada Permainan Action RPG. *Matics*, *12*(1), 87. https://doi.org/10.18860/mat.v12i1.8959

Faqih, A. H. (2016). *Analisis Kendala Penerapan Teknologi VFX pada Perfilman Indonesia*. 5.

Imam Jalaludin Muhammad bin Ahmad Mahalli dan Syaikh Jalaluddin Abdurahman bin Abi Bakar Suyuti. (2010). *Terjemah TafSIR Jalalain*. 1–402.

Lewis, J. R., & Sauro, J. (2018). Item Benchmarks for the System Usability Scale. *Journal of Usability Studies*, *13*(3), 158–167.

Maxmilano, R. (2020). *Implementasi Algoritma Boids dan Collision Avoidance Pada Pergerakan NPC Dalam Game 2D Berbasis Web*. *021*, 115.

Mukodi. (2009). MODEL PEMBELAJARAN KELAS IMERSI (Studi Kasus Implementasi Manajemen di MA Hasyim Asy'ari Jepara). *Jurnal Penelitian Pendidikan* , *1*(1), 77–100.

Nordberg, J. (2020). *VISUAL EFFECTS FOR MOBILE*.

Okun, J. A., Lavery, K., & Zwerman, S. (2020). The VES Handbook of Visual Effects. In J. A. Okun & S. Zwerman (Eds.), *The VES Handbook of Visual Effects* (3rd Editio). https://doi.org/10.4324/9781351009409

Prasetyo, F. R., Muh, E., Jonemaro, A., & Akbar, M. A. (2017). Penerapan Algoritma Hybrid Pathfinding A * dan Boids untuk Game Pesawat Tempur. *Jurnal Pengembangan Teknologi Informasi Dan Ilmu Komputer (J-PTIIK) Universitas Brawijaya*, *1*(12), 1616–1621.

Reeves, W. T. (1983). Particle Systems - Technique for Modeling a Class of Fuzzy Objects. *Computer Graphics (ACM)*, *17*(3), 359–376. https://doi.org/10.1145/964967.801167

Reynolds, C. W. (1987). Flocks-Hers-and-Schools. *Computer Graphics*, *21*(4), 25–34.

Riyadh, M. T. (2022). *No Title*. Https://Tafsirweb.Com. https://tafsirweb.com/4166-surat-al-hijr-ayat-16.html

Sajwan, M., Gosain, D., & Surani, S. (2014). *Flocking Behaviour Simulation : Explanation and Enhancements in Boid Algorithm* (Vol. 5, Issue 4). http://www.red3d.com/cwr/birds/

Simamora, P. R., Zega, S. A., & St, S. (2019). Perancangan 3D Modeling Dan Vfx Water Simulation Dalam Animasi 3D Berjudul "Blue & Flash." *Journal of Applied Multimedia and Networking (JAMN)*, *3*(2), 2548–6853. http://jurnal.polibatam.ac.id/index.php/JAMN

Sudrajat, M. A. (2022). Perancangan video game action RPG Fantasy Aious untuk perangkat komputer. *Journal of Digital Communication and Design (JDCODE)*, *1*(1), 22–29. http://ejurnal.ars.ac.id/index.php/jdcode/article/view/701

Ta'dzhim, M. A.-Q. (2022). *No Title*. Tafsirweb.Com. https://tafsirweb.com/11032-surat-al-mulk-ayat-4.html

Triartanto, A. Y., Martias, Suriyanto, A. D., & Fitriyanto. (2019). Anomalus sebagai

Brand Aura pada Karakter Superhero Film Avangers: Endagame. *Jurnal Mitra Pendidikan*, *3*(12), 1395–1408. http://www.e-jurnalmitrapendidikan.com/index.php/e-jmp/article/view/737/473

Yumarlin MZ. (2016). Evaluasi Penggunaan Website Universitas Janabadra Dengan Menggunakan Metode Usability Testing. *Informasi Interaktif*, *1*(1), 34–43. http://www.e-journal.janabadra.ac.id/index.php/informasiinteraktif/article/view/345

Zhang, B., & Hu, W. (2017). Game special effect simulation based on particle system of Unity3D. *Proceedings - 16th IEEE/ACIS International Conference on Computer and Information Science, ICIS 2017*, 595–598. https://doi.org/10.1109/ICIS.2017.7960062

Zhao, S., Xu, Y., Luo, Z., Tao, J., Li, S., Fan, C., & Pan, G. (2021). Player Behavior Modeling for Enhancing Role-Playing Game Engagement. *IEEE Transactions on Computational Social Systems*, *8*(2), 464–474. https://doi.org/10.1109/TCSS.2021.3052261

Zhu, G., Yao, S., Cui, X., Xu, S., IEEE Computer Society, International Association for Computer & Information Science, Wuhan da xue, & Institute of Electrical and Electronics Engineers. (2017). *16th IEEE/ACIS International Conference on Computer and Information Science (ICIS 2017) : proceedings : May 24-26, 2017, Wuhan, China*.