

**MEKANISME FORMASI GERAK BERKELOMPOK
(*FLOCKING*) PADA PARTIKEL YANG BERGERAK SENDIRI**

SKRIPSI

**Disusun sebagai Salah Satu Syarat untuk Memperoleh Gelar Sarjana Sains pada
Jurusen Fisika**



Oleh
ARIQ DHIA IRFANUDIN
1157030004

**JURUSAN FISIKA
FAKULTAS SAINS DAN TEKNOLOGI
UNIVERSITAS ISLAM NEGERI SUNAN GUNUNG DJATI
BANDUNG
2019**

a.n Kementrian Agama Universitas Islam Negeri (UIN) Sunan Gunung Djati Bandung Fakultas Sains dan Teknologi Jl. A. H. Nasution 105 Bandung	FORM (FR)	Nomor Dokumen	FST-TU-AKM-FR-D.03
		Tanggal Terbit	1 September 2016
		Nomor Revisi	02
		Halaman	1/1

SURAT PERNYATAAN KEASLIAN SKRIPSI

Saya yang bertanda tangan dibawah ini

Nama : ARIQ DHIA IRFANUDIN
NIM : 1157030004
Jurusan : JURUSAN FISIKA

Dengan ini menyatakan sebagai berikut :

1. Skripsi yang berjudul MEKANISME FORMASI GERAK BERKELOMPOK (*Flocking*) PADA PARTIKEL YANG BERGERAK SENDIRI ini adalah asli dan belum pernah diajukan untuk mendapatkan gelar akademik, baik di UIN Sunan Gunung Djati Bandung maupun di Perguruan Tinggi lain.
2. Karya tulis ini adalah murni gagasan, rumusan, dan penelitian saya, tanpa bantuan pihak lain, kecuali arahan Tim Pembimbing, diskusi teman, dan masukan dari Tim Penelaah.
3. Dalam karya tulis ini tidak terdapat karya atau pendapat yang telah ditulis atau dipublikasikan orang lain, kecuali secara tertulis dengan jelas dicantumkan dalam daftar pustaka sebagai acuan dalam naskah dengan menyebutkan nama pengarangnya.
4. Pernyataan ini saya buat dengan sesungguhnya, apabila di kemudian hari terdapat penyimpangan dan ketidakbenaran dalam pernyataan ini, maka saya bersedia menerima sanksi akademik berupa pencabutan gelar yang telah diperoleh karena karya ini, serta sanksi lainnya sesuai dengan norma yang berlaku di Perguruan Tinggi ini.
5. Lembar pernyataan ini saya buat sebenar-benarnya tanpa ada paksaan dan tekanan dari pihak manapun.

Bandung,26 Agustus 2019
Yang Membuat Pernyataan

ARIQ DHIA IRFANUDIN
1157030004

LEMBAR PERSETUJUAN

**MEKANISME FORMASI GERAK BERKELOMPOK (*FLOCKING*) PADA
PARTIKEL YANG BERGERAK SENDIRI**

ARIQ DHIA IRFANUDIN

1157030004

Menyetujui,

Pembimbing 1,

Pembimbing 2,

Dr. Yudha Satya Perkasa, M.Si
NIP. 197911172011011005

Dr. rer. nat. Sparisoma Viridi
NIP. 198510172007070105

Mengetahui,

Dekan
Fakultas Sains dan Teknologi,

Ketua
Jurusan Fisika,

Dr. Hasniah Aliah
NIP. 197911172011011005

Dr. Yudha Satya Perkasa, M.Si
NIP. 197911172011011005

LEMBAR PENGESAHAN

Skripsi dengan judul : "**MEKANISME FORMASI GERAK BERKELOMPOK (*Flocking*) PADA PARTIKEL YANG BERGERAK SENDIRI**" telah dipertanggung jawabkan dalam sidang Munaqasyah Jurusan Fisika Fakultas Sains dan Teknologi UIN Sunan Gunung Djati Bandung pada 26 Agustus 2019, dengan majelis yang terdiri dari:

Menyetujui ,

Penguji I

Penguji II,

Ridwan Ramdani, M.Si
NIP. 198904162019031016

Dr. rer. nat. Imamal Muttaqien
NIP. 198310062009121009

Mengetahui ,

Ketua Majelis,

Sekretaris,

Dr. Yudha Satya Perkasa, M.Si
NIP. 197911172011011005

Dr. rer. nat. Sparisoma Viridi
NIP. 198510172007070105

LEMBAR PERSEMBAHAN

"Sesungguhnya dalam penciptaan langit dan bumi, dan silih bergantinya malam dan siang terdapat tanda-tanda bagi Ulil Albab. (Yaitu) orang-orang yang mengingat Allah sambil berdiri atau duduk atau dalam keadaan berbaring dan mereka memikirkan tentang penciptaan langit dan bumi (seraya berkata): "Ya Tuhan kami, tiadalah Engkau menciptakan ini dengan sia-sia, Maha Suci Engkau, maka peliharalah kami dari siksa neraka". (Ali Imran: 190-191)

Ku persembahkan Skripsi ini untuk:

Alim Amyanto

Nuryati

Nida Syaqilla

ABSTRAK

Nama : ARIQ DHIA IRFANUDIN
Program Studi : Fisika
Judul : MEKANISME FORMASI GERAK BERKELOMPOK
(Flocking) PADA PARTIKEL YANG BERGERAK SENDIRI

Terdapat beberapa gerak berkelompok yang dilakukan oleh kelompok hewan, seperti burung, ikan dan serangga. Di alam, orde terstruktur ditunjukan oleh pergerakan dan arah setiap individual yang membuat mereka lebih aman dan effisien. Keinginan untuk berkelompok bergantung pada jarak tetangga terdekat meskipun masih ada beberapa parameter yang dapat menyebabkan keinginan tersebut. Untuk mendapatkan pemahaman lebih terhadap sistem ini, kita memanupulasi interaksi setiap partikel ke dalam dua situasi. Pada situasi statis, kita mengatur setiap partikel untuk berinteraksi hanya dengan kelompoknya sendiri, yang berarti partikel tidak akan merespon entitas yang lain. Pada situasi dinamis, kita mengatur setiap partikel dapat berinteraksi dengan seekor predator sepanjang partikel lain berada di dalam zona atraksi dan repulse partikel. Pendekatan Newtonian digunakan untuk menangani parameter fisika seperti posisi dan kecepatan partikel yang Kemudian dihitung menggunakan integrase Euler. Terdapat beberapa pola yang muncul di dalam simulasi kita seperti formasi garis dan formasi kelompok parallel.

Kata Kunci: Gerak berkelompok, Integrasi Euler, Interaksi tetangga.

ABSTRACT

Name : ARIQ DHIA IRFANUDIN
Studies Program : Physics
Title : *Flocking Mechanism in Self-Propelled Particle*

There is some collective motion that does by animal groups, such as bird flocks, fish school, insect swarms. In nature, Structural order showed in movement and direction every single individual make them safer and more efficient. The desire to flock is dependent on the nearest-neighbor distance even though there are other parameters that can affect the flocks. In order to get more insight into this system, we manipulate interaction with each vehicle in two situations. In a static situation, we set each vehicle to interact with their group only, which means the vehicle will not respond to other entities that not selected. In a dynamic situation, we set each vehicle can interact with a predator as long as the neighbor is in the orientation and attraction zones of the vehicle. The Newtonian approach will be used to handling physical parameters such as position, velocity and, acceleration of the vehicles then these physical parameters are calculated and updated by the Euler integration. There are some patterns that appear in our simulation such as line formation and dynamic parallel group.

Keywords: *Collective motion, Euler integration, Neighbor interaction.*

KATA PENGANTAR

Segala puji bagi Allah SWT, Skripsi yang berjudul MEKANISME FORMASI GERAK BERKELOMPOK (*Flocking*) PADA PARTIKEL YANG BERGERAK SENDIRI dapat diselesaikan. Ini diajukan untuk memenuhi syarat kelulusan program Strata-1 (S1) jurusan Fisika, Fakultas Sains dan Teknologi, UIN Sunan Gunung Djati Bandung.

Skripsi ini selesai dengan adanya bantuan dari berbagai pihak. Oleh karena itu penulis ucapkan terima kasih yang kepada:

1. Allah SWT yang selalu berada di dekat penulis memberikan rahmat dan hidayah untuk menjadi manusia yang berguna.
2. Nabi Muhammad SAW yang mana beliau adalah panutan bagi penulis.
3. Bapak Dr. Yudha Satya Perkasa dan Bapak Dr. rer. nat Sparisoma Viridi Selaku Dosen Pembimbing Tugas Akhir, karena atas Bimbingan dan kepercayaan yang beliau berikan kepada penulis untuk menyelesaikan skripsi ini.
4. Bapak Muhammad Ridwan, M.Si Selaku Dosen Pengaji 1.
5. Bapak Dr.rar.nat. Imamal Mutaqqien.M.Si, selaku Dosen Pengaji 2.
6. Seluruh dosen fisika UIN Bandung yang telah banyak meluangkan waktunya untuk memberikan pengetahuan, arahan, masukan, dan dukungan yang berarti bagi penulis. Dan juga senantiasa membimbing penulis mempelajari keilmuan dalam menyelesaikan skripsi ini.
7. Rekan seperjuangan laboratorium sistem modeling Dinda, Ikhsan, Arum dan Indri.
8. Rekan-rekan Fisika angkatan 2015 yang memotivasi agar penulis lulus sebagai sarjana.

Akhir kata, penulis menyadari karya tulis ini masih banyak kekurangan oleh karena itu kritik dan saran yang membangun sangat diharapkan demi kesempurnaan skripsi ini. dengan harapan semoga karya tulis ini bermanfaat bagi pembaca.

Bandung, 26 Agustus 2019

Penulis

DAFTAR ISI

SURAT PERNYATAAN KEASLIAN SKRIPSI	i
LEMBAR PERSETUJUAN	ii
LEMBAR PENGESAHAN	iii
LEMBAR PERSEMBAHAN	iv
ABSTRAK	v
ABSTRACT	vi
KATA PENGANTAR	vii
DAFTAR ISI	xi
DAFTAR GAMBAR	xii
DAFTAR TABEL	xiii
DAFTAR SINGKATAN	xiv
DAFTAR SIMBOL DAN OPERATOR	xv
1 PENDAHULUAN	1
1.1 Latar Belakang	1
1.2 Tujuan	2
1.3 Rumusan Masalah	2
1.4 Batasan Masalah	3
1.5 Metode Pengumpulan Data	3
1.6 Sistematika Penulisan	3

2 TEORI DASAR	5
2.1 Model <i>Flocking</i>	5
2.1.1 Dasar Mekanika <i>Flocking</i>	5
2.2 Model Interaksi <i>Self-Propelled Particles</i>	6
2.2.1 Asumsi dan persamaan	6
2.3 Aturan Dasar <i>Flocking</i>	7
2.3.1 <i>Alignment</i>	8
2.3.2 <i>Cohesion</i>	9
2.3.3 <i>Separation</i>	9
2.3.4 <i>Escaping</i>	10
2.4 Model Partikel	10
2.4.1 <i>Peripheral View</i>	11
2.4.2 <i>Perception Zone</i>	12
2.4.3 <i>Steering Behavior</i>	13
2.5 Metode Euler	14
3 METODOLOGI PENELITIAN	15
3.1 Alat yang Digunakan	15
3.1.1 Perangkat Keras	15
3.1.2 Perangkat Lunak	15
3.2 Penerapan Sebagian Aturan	16
3.2.1 Satu Aturan Berlaku	16
3.2.2 Dua Aturan Berlaku	16
3.3 Kombinasi Empat Aturan	17
3.3.1 Interaksi Statis	18
3.3.2 Interaksi Dinamis	18
3.4 Simulasi Komputer	18
3.5 Penentuan Parameter Gerak	20
4 HASIL DAN PEMBAHASAN	22
4.1 Sebagian Aturan	22
4.2 Kombinasi Empat Aturan	24
5 PENUTUP	28
5.1 Kesimpulan	28
5.2 Saran	30

DAFTAR PUSTAKA	31
INDEX	34
LAMPIRAN	36
A Kode Program	36
A.1 Kode Program index.html	36
A.2 Kode Program sketch.js	40
A.3 Kode Program loopedThings.js	44
A.4 Kode Program boid.js	48
B Karya Tulis	54
B.1 Simposium Nasional Inovasi dan Pembelajaran Sains 2018	54
B.2 Simulasi Peracunan Produk Fisi pada Teras Reaktor Nuklir Berbasis JavaScript	60
B.3 <i>Static and Dynamic Neighbor Interaction of Flocking Model in New-</i> <i>tonian Approach</i>	67
C Riwayat Hidup	77

DAFTAR GAMBAR

2.1	Tiga aturan dasar interaksi <i>flocking</i>	8
2.2	Model partikel dengan <i>peripheral view</i> (ϑ), <i>separation zone</i> (z_s), <i>alignment zone</i> (z_a), <i>cohesion zone</i> (z_c) dan <i>escaping zone</i> (z_e).	11
2.3	Skema <i>steering behavior</i>	13
3.1	Skema <i>perception zone</i> , (a) partikel dengan jari-jari z_a , z_c dan z_s yang sama. (b) partikel dengan jari-jari z_s dan $z_a = z_c$. (c) partikel dengan jari-jari $z_a = z_c$, $z_a > z_s$ dan $z_a < z_e$	17
3.2	Algoritma pemrograman.	19
3.3	Model partikel dengan ID.	21
4.1	Hanya satu aturan <i>flocking</i> yang diterapkan dengan $N = 30$, $z_a = 0.8$, $z_c = 0.8$, $z_s = 0.5$, dan $t = 300$. (a) hanya <i>alignment</i> , (b) hanya <i>cohesion</i> , dan (c) hanya <i>separation</i>	22
4.2	Hanya dua aturan <i>flocking</i> yang diterapkan dengan $N = 30$, $z_a = 0.8$, $z_c = 0.8$, $z_s = 0.5$, dan $t = 300$. (a) <i>alignment</i> dan <i>cohesion</i> , (b) <i>alignment</i> dan <i>separation</i> , (c) <i>cohesion</i> dan <i>separation</i>	23
4.3	Interaksi tanpa pedator dengan $N = 30$. (a) $z_a = 0.5$, $z_c = 0.5$, dan $z_s = 0.5$ (b) $z_a = 0.8$, $z_c = 0.8$, dan $z_s = 0.5$	24
4.4	Interaksi dengan pedator. $N = 30$. (a) $z_a = 1$, $z_c = 1$ $z_s = 0.5$, dan $z_e = 0.2$. (b) $z_a = 1$, $z_c = 1$, $z_s = 0.5$, dan $z_e = 0.8$. (c) $z_a = 1$, $z_c = 1$, $z_s = 0.5$, dan $z_e = 1.4$	25
4.5	Hasil simulasi yang telah dilakukan oleh Lebar Bajec.	27

DAFTAR TABEL

3.1	Spesifikasi Perangkat Keras.	15
3.2	Spesifikasi Perangkat Lunak.	15
3.3	Perbandingan penulisan <i>code</i> menggunakan dan tanpa <i>library P5js.</i> .	19

DAFTAR SINGKATAN

SPPs	: <i>Self Propelled Particle systems</i>
OOP	: <i>Object Oriented Programming</i>
SPP	: <i>Self Propelled Particle</i>
HTML	: <i>Hyper Text Markup Language</i>
CSS	: <i>Cascading Style Sheet</i>

DAFTAR SIMBOL DAN OPERATOR

v	:	Kecepatan
p	:	Posisi
m	:	Massa
Δt	:	<i>Step Size</i>
t	:	Waktu
ω	:	<i>Weighting Factor</i>
P	:	<i>Function of Perception Zone</i>
ϑ	:	<i>Peripheral View</i>
d	:	<i>distance</i>
D	:	<i>Flocking Function</i>
z	:	<i>Perception Zone</i>
x	:	<i>Input State</i>
λ	:	<i>Output state</i>
q	:	<i>Current State Properties</i>

BAB 1

PENDAHULUAN

1.1 Latar Belakang

Terdapat fenomena alam yang begitu menakjubkan, ketika sekelompok hewan pada spesies yang sama mampu bergerak secara sinkron. Setiap individu dalam kelompok memiliki arah dan kelajuan yang relatif sama tanpa saling bertabrakan satu sama lain. Fenomena ini disebut *flocking*, istilah ini sering digunakan untuk menjelaskan gerak berkelompok pada kawanan burung. Adapun *school* dan *herd* adalah gerak berkelompok yang merujuk pada hewan ikan dan binatang darat. Sistem berkelompok ini sangat bermanfaat bagi hewan-hewan tersebut karena dapat membantu dalam perlindungan diri dari pemangsa, untuk menghemat energi saat burung sedang bermigrasi (formasi V) maupun membantu saat sedang mencari makanan.

Para ahli biologi sudah banyak yang mengkaji fenomena gerak berkelompok (Niwa, 1996) (Toner, 1998) (Huth & Wissel, 1992)(Huth & Wissel, 1994). Secara konsep, gerak berkelompok cukup mudah untuk dijelaskan. Ketika sekelompok ikan sedang diincar oleh pemangsa, semua individu bergerak menuju lokasi yang paling aman. Namun bagaimana caranya setiap ikan dalam kelompok tersebut mendapatkan informasi yang mereka butuhkan untuk bergerak secara sinkron dan menghindari pemangsa?

Sekitar tahun 1987, Craig Reynolds membuat model untuk memvisualisasikan sekelompok hewan yang mampu merespon pergerakan individu di dalam kelompok. Reynolds memberikan istilah **boids** untuk mensimulasikan gerak *flocking* pada burung dan membuat setiap boids dalam kelompok menjadi sebuah agen independen

yang mengikuti beberapa aturan sederhana agar mampu melakukan beberapa hal secara mandiri (Reynolds, 1987).

Dari simulasi yang telah dibuat oleh Craig Reynolds, cukup sulit untuk menakar keberhasilan dan validitasnya secara obyektif. Maka dari itu terdapat beberapa tinjauan lain secara kuantitatif untuk mendapatkan pemahaman lebih tentang fenomena gerak berkelompok. Diantaranya dipaparkan oleh Vicsek yang mengenalkan *Self-Driven Particles System* (Chaté *et al.*, 2008b) yang kemudian menjadi *Self-Propelled Particles System* (SPPs) (Vicsek & Zafeiris, 2012). Model interaksi pada partikel yang bergerak sendiri atau SPP dapat digunakan untuk menjelaskan perubahan gerak setiap partikel yang pada awalnya tidak beraturan menjadi terarah dan dapat membentuk formasi yang unik.

Telah diketahui dalam membuat simulasi gerak berkelompok, terdapat tiga aturan dasar yang harus dipenuhi. Adapun tinjauan secara kuantitatif turut menyumbangan agar ditemukannya simulasi gerak berkelompok yang lebih natural.

Namun apa mekanisme minimal yang harus dipenuhi agar gerak berkelompok dapat terjadi dan tetap stabil? *Flocking* dapat dikatakan stabil ketika partikel dapat berkelompok dengan jumlah anggota yang banyak walaupun disaat yan sama harus menghindar dari kejaran partikel pemangsa/predator.

1.2 Tujuan

Tujuan dari penelitian ini adalah menganalisis dan mensimulasikan mekanisme minimal apa yang harus dipenuhi agar *flocking* dapat terjadi dan tetap stabil.

1.3 Rumusan Masalah

Rumusan masalah dalam penelitian ini adalah

- Apa mekanisme minimal terjadinya *flocking*?
- Bagaimana setiap parameter terasosiasi pada setiap individu dengan gaya-gaya interaksi yang mempengaruhi eksistensi flocking, geometri, kecepatan, arah gerak dan stabilitas dari struktur *flocking*.

1.4 Batasan Masalah

Dalam penelitian ini digunakan tiga metode pengumpulan data yaitu:

- Semua partikel dianggap sama dan mengikuti aturan yang sama
- Setiap partikel mampu merasakan partikel lainnya jika partikel tersebut berada di dalam radius persepsi.
- Gaya yang diberikan oleh partikel tetangga yang berbeda turut dipertimbangkan.
- Besar gaya diantara partikel berpasangan bergantung pada jarak diantara pasangan dan kecepatan relatif keduanya.

1.5 Metode Pengumpulan Data

Dalam penelitian ini digunakan tiga metode pengumpulan data yaitu:

- Studi literatur. Sebelum melakukan analisis dan simulasi, terlebih dahulu dilakukan studi literatur yang bersumber dari jurnal ilmiah dan buku agar mendapatkan informasi yang dapat dijadikan sebagai acuan selama penelitian.
- Metode numerik. Besaran-besaran pada simulasi akan diupdate menggunakan metode interasi Euler.
- *Object Oriented Programming* (OOP). Metode OOP digunakan untuk membangun simulasi yang menggunakan bahasa pemrograman *JavaScript*.

1.6 Sistematika Penulisan

Pembahasan pokok dari penelitian ini dibagi menjadi beberapa bab yang diuraikan secara singkat sebagai berikut:

BAB I PENDAHULUAN

Bab ini menguraikan mengenai latar belakang, perumusan masalah, batasan masalah, tujuan penelitian, metode pengumpulan data, dan sistematika penulisan.

BAB II TEORI DASAR

Bab ini berisi teori-teori yang dapat membangun penelitian diantaranya, School, Vicsek Model, Model Interaksi Self-Propelled Particles.

BAB III METODOLOGI PENELITIAN

Pada bab ini diuraikan tahap-tahap dalam penelitian. Tahapan tersebut meliputi: analisis persamaan analitik dan mensimulasikan program secara numerik.

BAB 2

TEORI DASAR

2.1 Model *Flocking*

2.1.1 Dasar Mekanika *Flocking*

Gerak berkelompok suatu organisme erat kaitannya dengan disiplin ilmu biologi. Namun apa yang membedakan fenomena gerak berkelompok dari tinjauan fisika dan biologi? *Collision Rule* antar individu yang secara prinsip membedakan dua jenis sistem tersebut. Istilah tersebut digunakan untuk menjelaskan bagaimana nilai kecepatan tiap individu berubah selama berinteraksi. Karena partikel yang bergerak dengan kecepatan yang tetap, arah gerak tiap partikel bergantung pada besar kecepatan rata-rata semua partikel dalam kelompok.

Proses perubahan fase dari kelompok yang tidak teratur menjadi memiliki kecepatan yang sama dan saling terkoordinasi dapat diakibatkan oleh perubahan satu atau lebih variabel yang spesifik pada sistem. Hal ini disebut dengan *order parameter*. *Order parameter* pada kasus gerak berkelompok adalah kecepatan ternormalisasi rata-rata.

Dengan *order parameter* ini, model Vicsek mampu menjelaskan bagaimana tiap individu berkoordinasi dalam sistem agar *flocking* dapat terjadi. Adapun salah satu model yang ada adalah model interaksi *Self-Propelled Particles* (SPP) atau model untuk partikel yang dapat bergerak sendiri. Bergerak sendiri dalam artian partikel mampu bergerak secara independen tanpa harus bergantung kepada partikel lain. Dalam model dasar ini, tiap partikel diasumsikan akan bergerak dengan kecepatan yang tetap v_0 dengan posisi dan kecepatan awal yang acak.

Untuk menginisiasi *flocking* pada partikel, terdapat *input state* $x = \langle \lambda_1, \dots, \lambda_n \rangle$ dan *output state* $\lambda(q) = \langle p, v \rangle$ yang akan digunakan untuk memproses properti dari model partikel. Setiap partikel memiliki propertinya masing-masing yaitu posisi, kecepatan, massa dan *perception zone* yang akan ditinjau dalam simulasi. Properti ini disebut dengan *current state properties* (q) (Bajec *et al.*, 2007). Dengan kata lain, properti tersebut adalah karakteristik yang dimiliki setiap partikel.

$$q = \langle \mathbf{p}, \mathbf{v}, z, m, \vartheta \rangle \quad (2.1)$$

2.2 Model Interaksi *Self-Propelled Particles*

Model interaksi SPP telah dikenalkan oleh (Chaté *et al.*, 2008b) sebagai kasus spesial dari *Boids model* yang dikenalkan oleh Reynolds (Reynolds, 1987). Pada pendekatan untuk membuat partikel yang bergerak sendiri. Setiap partikel bergerak dengan kecepatan absolut konstan yang kemudian dalam pergerakannya akan mendapat pengaruh dari lingkungan. Hal ini telah dilakukan oleh (Czirók & Vicsek, 2000) (Levine *et al.*, 2001) (Czirók *et al.*, 1999). Namun disisi lain terdapat perbedaan pada sistem SPP yang dibuat oleh (Li Chuang *et al.*, 2007) (Li *et al.*, 2008) yaitu partikel dapat mengalami perlambatan ataupun percepatan. Perubahan kecepatan ini diakibatkan dimasukannya gaya interaksi antar partikel saat bergerak.

2.2.1 Asumsi dan persamaan

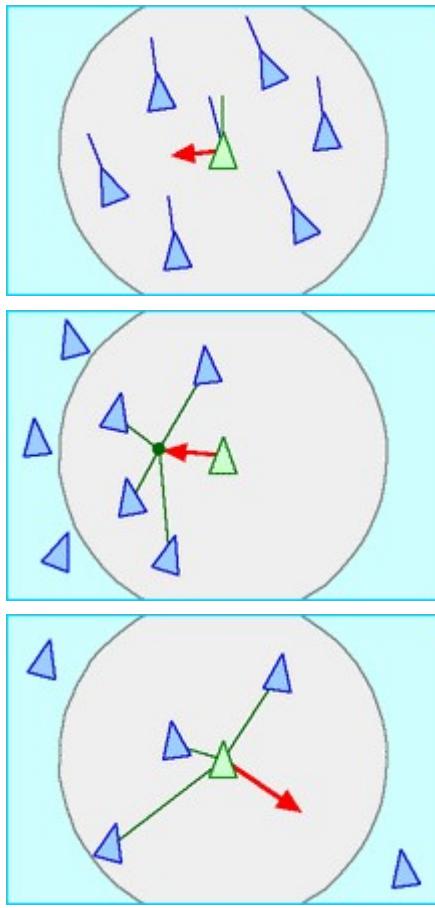
Dalam simulasi, jumlah partikel akan diwakili oleh N partikel. setiap partikel memiliki kecepatan yang sama dan diidentifikasi dengan $i = 1, 2, \dots, n$, bermassa $m = 1$ dengan posisi dan kecepatan dilambangkan oleh p_i dan v_i yang nilainya akan selalu diperbarui dengan pertambahan nilai $\Delta t = 1$. Adapun nilai p_0 dan v_0 akan bernilai acak.

$$\vec{F}_i = \sum_{i \neq j} \left(\sum_{n=1}^{\infty} \alpha_n \frac{\vec{p}_{ij}}{|\vec{p}_{ji}|^n} + \sum_{n=1}^{\infty} \beta_n \frac{\dot{\vec{p}}_{ji}}{|\dot{\vec{p}}_{ji}|^n} \right). \quad (2.2)$$

Persamaan 2.2 adalah persamaan umum untuk menentukan gaya-gaya yang terjadi selama partikel bergerak. Nilai α dan β adalah *force factor* sebagai pengali untuk *Drive Function* (D) yang akan dipaparkan di bab selanjutnya. Adapun dalam penelitian ini untuk ruas posisi nilai n hanya akan sampai dua. Sedangkan untuk ruas kecepatan nilai n hanya akan bernilai satu.

2.3 Aturan Dasar *Flocking*

Adalah Reynolds yang pertama kali memvisualisasikan obyek dalam jumlah banyak bergerak bersama. Terdapat tiga aturan dasar yang dikenalkannya agar flocking dapat terjadi. Diantaranya adalah kemampuan untuk menghindar dari tabrakan, bergerak menuju arah yang sama dengan tetangganya, dan mencoba untuk tetap dekat dengan pusat massa kelompok yang diilustrasikan pada **Gambar 2.1**.



(a). Pensejajaran: bergerak dengan kelajuan dan kecepatan yang sama dengan partikel tetangga

(b). Kohesi: bergerak menuju titik pusat massa kelompok.

(c). Pemisahan: menghindari partikel tetangga yang terlalu dekat.

Gambar 2.1: Tiga aturan dasar interaksi *flocking*.

Kelajuan dan arah setiap individu dibuat agar stokastik, namun arah dari kelompok bergantung pada lokasi dan arah individu yang lain (tetangga). Penelitian yang menginisiasi gerak berkelompok ini telah mampu membuat model gerak berkelompok tanpa adanya pemimpin.

2.3.1 *Alignment*

Aturan *alignment* membuat setiap partikel tetap bergerak bersama dalam kelompok. Dalam pandangan biologi, ada beberapa alasan mengapa sekelompok makhluk hidup melakukan hal ini. Salah satu contohnya adalah untuk bertahan hidup dari predator dan memudahkan untuk mencari makan. *Alignment* juga membuat tiap partikel tidak mudah untuk bertabrakan karena setiap partikel akan selalu berusaha untuk bergerak ke arah yang sama dengan kelajuan yang sama. Untuk menyesua-

ikan kecepatan, partikel akan memperhitungkan kecepatan rata-rata tetangga terdekatnya. Secara matematik dapat dituliskan sebagai berikut:

$$D_i^a(p, q) = \alpha^a \frac{1}{N^a} \sum_{i \neq j}^{N^a} \frac{\vec{v}_{ji}}{|\vec{v}_{ji}|}, \alpha^a = 1N, \quad (2.3)$$

dengan $v_{ji} = (v_j - v_i)$.

2.3.2 Cohesion

Aturan kohesi yang membuat partikel untuk selalu berusaha untuk berada di dekat tetangga yang lainnya. Hal ini dikarenakan partikel akan selalu mencari dimana posisi pusat massa dari kelompoknya dan bergerak ke titik tersebut. Untuk melakukan hal ini maka setiap partikel harus memperhitungkan posisi rata-rata dari setiap tetangga terdekatnya. Secara matematik dapat ditulis seperti

$$D_i^c(p, q) = \alpha^c \frac{1}{N^c} \sum_{i \neq j}^{N^c} \frac{\vec{p}_{ji}}{|\vec{p}_{ji}|}, \alpha^c = 1N, \quad (2.4)$$

dengan $p_{ji} = (p_j - p_i)$.

2.3.3 Separation

Aturan *separation* menjadi aturan yang paling penting diantara aturan yang lainnya dan tentunya tidak bisa dihiraukan oleh setiap partikel. Hal ini dikarenakan kasus *flocking* pada burung yang mana jika setiap burung mengabaikan aturan *separation* maka burung tersebut akan bertabrakan dengan burung yang lainnya. Oleh karena itu, aturan ini diperlukan agar partikel tetap bisa bergerak tanpa saling bertabrakan satu sama lain. Untuk melakukan hal tersebut maka setiap partikel harus memperhitungkan jarak ambang batas antara dirinya dengan tetangganya. Jika telah melewati ambang batas tersebut maka partikel harus bergerak menjauh atau ke arah yang berlawanan dari tetangganya. Seperti pada persamaan berikut:

$$D_i^s(p, q) = \alpha^s \sum_{i \neq j}^{N^s} \frac{\vec{p}_{ji}}{|\vec{p}_{ji}|^2}, \alpha^s = 1 Nm. \quad (2.5)$$

Pangkat dua pada **Persamaan** (2.5) memiliki kemiripan dengan Hukum Coloumb. Jika semakin dekat jarak antar partikel, maka semakin besar gaya tolak yang akan dihasilkan.

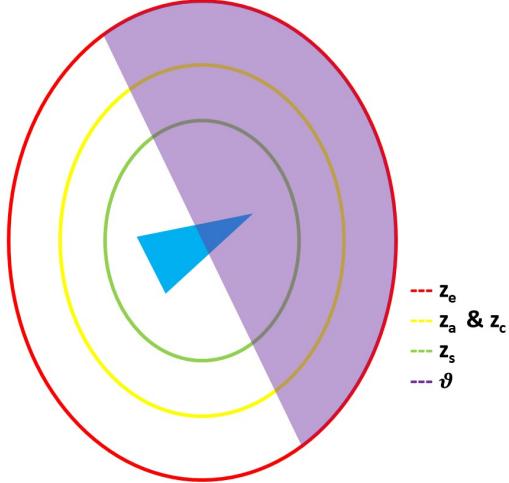
2.3.4 Escaping

Aturan *escaping* adalah aturan tambahan untuk partikel ketika berinteraksi dengan predator. Dengan aturan ini partikel akan mampu mengetahui posisi predator yang Kemudian informasi tersebut digunakan untuk partikel melarikan diri atau bergerak menjauh dari predator tersebut. Partikel harus bergerak menurut persamaan

$$D_i^e(p, q) = \alpha^e \sum_{i \neq j}^{N^e} \frac{\vec{p}_{ji}}{|\vec{p}_{ij}|}, \alpha^e = -1 N. \quad (2.6)$$

2.4 Model Partikel

Partikel memiliki dua komponen utama yang menunjangnya melakukan *flocking* yaitu *perception zone* yang memiliki jari-jari tertentu dan berfungsi sebagai sensor partikel untuk pengambilan sikap atau respon apa yang akan partikel lakukan terhadap partikel lainnya. Setiap *perception zone* memiliki fungsinya masing-masing, bergantung pada aturan *flocking* mana yang diterapkan. Sedangkan *peripheral view* (ϑ) berfungsi sebagai mata untuk mendeteksi posisi obyek lain di sekitarnya.



Gambar 2.2: Model partikel dengan *peripheral view* (ϑ), *separation zone* (z_s), *alignment zone* (z_a), *cohesion zone* (z_c) dan *escaping zone* (z_e).

2.4.1 Peripheral View

Peripheral view (ϑ) disini adalah pendekatan yang dilakukan agar model partikel lebih realistik. Pendekatan ini mirip seperti yang dilakukan (Frame & Flake, 2000) sebagai upaya membentuk formasi V seperti burung. *Peripheral view* berfungsi untuk membatasi penglihatan partikel. Sedangkan pada **Gambar 2.2** luas area di luar *peripheral view* dapat disebut titik buta partikel.

$$\varphi(\lambda_i, q_j) = \arccos \left(\frac{\mathbf{v}_j \cdot (\mathbf{p}_i - \mathbf{p}_j)}{\|\mathbf{v}_j\| \|\mathbf{p}_i - \mathbf{p}_j\|} \right) \quad (2.7)$$

Untuk menentukan apakah partikel ke- j berada di dalam *peripheral view*, sudut diantara kedua partikel harus dihitung terlebih dahulu yang kemudian besar sudut tersebut akan digunakan sebagai referensi untuk mengetahui posisi partikel ke- j .

2.4.2 Perception Zone

Dapat dilihat pada **Gambar 2.2** terdapat lingkaran merah dan lingkaran biru dan lingkaran hijau. Lingkaran merah menunjukkan *perception zone* yang akan menangani aturan *separation* (z_s). Sedangkan lingkaran biru menunjukkan *perception zone* yang akan menagani aturan *alignment* (z_a) dan *cohesion* (z_c). Adapun lingkaran hijau menunjukkan *perception zone* yang akan menangani aturan *escaping* (z_e). Jika ada partikel lain masuk/berada di dalam *perception zone*, maka partikel harus merespon dengan aturan dasar yang ada. Namun jika partikel tetangga berada di luar *perception zone* maka partikel tetangga tersebut tidak akan diperhitungkan.

$$d(\lambda_i, q_j) = \|p_i - p_j\| \quad (2.8)$$

$$P_i^a(x, q) = \{d(\lambda_i, q_j) < z_a \wedge \varphi(\lambda_i, q_j) < \vartheta\}, \quad (2.9)$$

$$P_i^c(x, q) = \{d(\lambda_i, q_j) < z_c \wedge \varphi(\lambda_i, q_j) < \vartheta\}, \quad (2.10)$$

$$P_i^s(x, q) = \{d(\lambda_i, q_j) < z_s \wedge \varphi(\lambda_i, q_j) < \vartheta\}, \quad (2.11)$$

$$P_i^e(x, q) = \{d(\lambda_i, q_j) < z_e \wedge \varphi(\lambda_i, q_j) < \vartheta\}. \quad (2.12)$$

Dalam penentuan apakah partikel ke- j berada di dalam *perception zone* partikel ke- i , jarak (d) diantara kedua partikel harus diketahui terlebih dahulu dengan menghitung jumlah magnitudo vektor posisinya. Nilai jarak tersebut digunakan sebagai referensi apakah partikel berada di dalam atau di luar *perception zone*. Kemudian dihitung pula apakah sudut partikel ke- j berada dalam batas *peripheral view*. Jika syarat tersebut dipenuhi, *function of perception zone* (P_i) pada **Persamaan 2.9, 2.10, 2.11, dan 2.12** akan bertindak untuk menentukan apakah partikel ke- j berada di dalam jangkauan partikel ke- i .

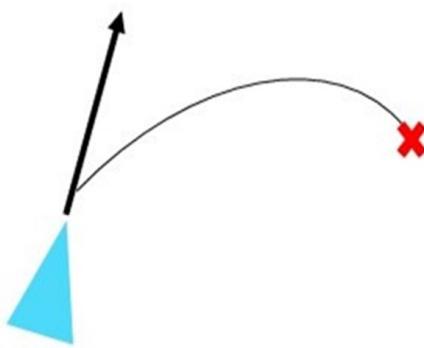
2.4.3 Steering Behavior

Kemampuan partikel untuk merespon lingkungannya dibatasi oleh ukuran *peripheral view* dan *perception zone*. Ketika ada partikel lain telah memenuhi syarat yang terdapat pada **Persamaan** (2.9), (2.10), (2.11), dan (2.12), maka setiap fungsi aturan dasar *flocking* pada **Persamaan** (2.3), (2.4), (2.5), dan (2.6) akan menghasilkan sebuah gaya yang akan menyebabkan perubahan kecepatan pada partikel.

$$D_i (\langle p_1, \dots, p_k \rangle, q), \quad (2.13)$$

$$p_i = P_i(x, v). \quad (2.14)$$

Adapun $P = \langle P_1, \dots, P_k \rangle$ adalah fungsi untuk menghitung *perception zone*, $D = \langle D_1, \dots, D_l \rangle$ adalah fungsi untuk menghitung aturan dasar *flocking*. Pada tahap pertama partikel akan bergerak dengan mengikuti skema pada **Persamaan** (2.14) untuk menentukan apakah partikel tetangga berada di dalam jangkauan. Jika syarat tersebut terpenuhi maka akan masuk ke skema kedua yang terdapat pada **Persamaan** (2.13). Pada tahap ini partikel partikel akan merespon sesuai dengan aturan dasar *flocking* yang berlaku. Adapun bentuk respon yang muncul bergantung pada fungsi *perception zone* mana yang aktif seperti yang diilustrasikan pada **Gambar 2.3**.



Gambar 2.3: Skema steering behavior.

2.5 Metode Euler

Metode Euler adalah satu-satu dari beberapa metode numerik untuk menyelesaikan persamaan diferensial. Metode ini adalah metode paling sederhana dari integrasi numerik persamaan diferensial biasa dan juga bentuk paling sederhana dari metode Runge-Kutta. Meskipun metode Euler terbilang sederhana, metode ini memiliki nilai error yang cukup besar jika dibandingkan dengan metode lain. Nilai error ini dapat diatasi dengan pengaturan nilai *step size* (Δt) yang kecil. Namun jika nilai step size terlalu kecil maka simulasi akan berjalan sangat lambat. Oleh karena itu *step size* yang digunakan pada penelitian ini sebesar $\Delta t = 1$. Pada penelitian ini metode Euler diterapkan untuk mengatasi pergerakan partikel. Kelajuan partikel yang konstan akan digunakan untuk menghitung posisi partikel setiap detiknya. Jika dituliskan dalam persamaan matematika, persamaan yang digunakan untuk mengatasi perubahan posisi, kecepatan dan percepatan sebagai berikut:

$$\vec{a}_i = \frac{1}{m_i} \sum \vec{F}, \quad (2.15)$$

$$\vec{v}_i^{k+1} = \vec{v}_i^k + \vec{a}_i \Delta t, \quad (2.16)$$

$$\vec{p}_i^{k+1} = \vec{p}_i^k + \vec{v}_i^{k+1} \Delta t. \quad (2.17)$$

Penggunaan metode Euler pada penelitian ini dikarenakan alokasi *memory* komputer yang digunakan tidak terlalu besar jika dibandingkan dengan metode yang lain. Hal ini menyebabkan kerja komputer akan lebih ringan. Pertimbangan ini meninjau pula banyaknya jumlah partikel yang akan saling berinteraksi.

BAB 3

METODOLOGI PENELITIAN

3.1 Alat yang Digunakan

3.1.1 Perangkat Keras

Perangkat keras yang digunakan adalah sebuah laptop dengan spesifikasi sebagai berikut:

Tabel 3.1: Spesifikasi Perangkat Keras.

Komponen	Spesifikasi
<i>Processor</i>	Intel(R) Core (TM) i3-5005U CPU @ 2.00Ghz (4 CPUs), 2.0Ghz
<i>Memory</i>	8192MB RAM
<i>Display</i>	Intel(R) HD Graphics 5500, 128 MB VRAM
<i>Render</i>	NVIDIA GeForce 930M, 2010 MB VRAM
<i>Operating System</i>	Windows 10

3.1.2 Perangkat Lunak

Perangkat lunak yang digunakan adalah sebagai berikut:

Tabel 3.2: Spesifikasi Perangkat Lunak.

Komponen	Software
<i>Text Editor</i>	<i>Visual Code Studio</i>
<i>Browser</i>	<i>Google Chrome</i>
<i>Library</i>	<i>P5js Library</i>

Perangkat keras pada **Tabel 3.1** adalah milik pribadi penulis. Meninjau simulasi akan menggambar banyak partikel yang dihitung terus menerus maka *graphic card* dan RAM yang digunakan sebaiknya tidak jauh berbeda dengan yang dimiliki penulis. Adapun perangkat lunak pada **Tabel 3.2** yang digunakan dapat ditemukan secara gratis di internet.

3.2 Penerapan Sebagian Aturan

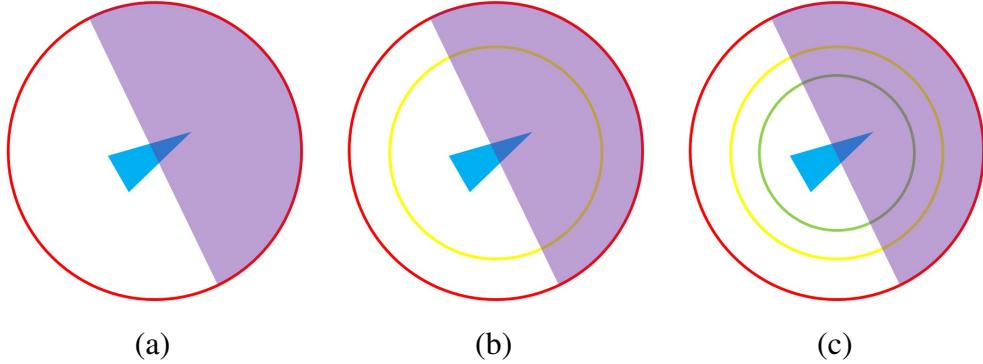
3.2.1 Satu Aturan Berlaku

Untuk mengetahui aturan dasar *flocking* mana yang menjadi pondasi agar terbentuknya *flocking*, partikel disimulasikan dalam beberapa kasus. Pada simulasi akan diatur untuk setiap partikel hanya akan bergerak dengan mematuhi aturan *alignment*, *cohesion*, atau *separation* saja.

3.2.2 Dua Aturan Berlaku

Pola interaksi partikel yang mengombinasikan dua aturan dasar *flocking* diharapkan akan menghasilkan formasi yang beragam. Untuk kasus pertama, *alignment* akan diikombinasikan dengan *cohesion*. Untuk kasus kedua, *alignment* akan dikombinasikan dengan *separation*. Percobaan pada kasus kedua ini mirip seperti penelitian yang dilakukan oleh (Chaté *et al.* , 2008a) (Chaté *et al.* , 2008b) yang mensimulasikan partikel tanpa adanya aturan *cohesion*. Sedangkan untuk kasus ketiga, *cohesion* akan dikombinasikan dengan *separation*.

3.3 Kombinasi Empat Aturan



Gambar 3.1: Skema *perception zone*, (a) partikel dengan jari-jari z_a , z_c dan z_s yang sama. (b) partikel dengan jari-jari z_s dan $z_a = z_c$. (c) partikel dengan jari-jari $z_a = z_c$, $z_a > z_s$ dan $z_a < z_e$.

Berdasarkan **Persamaan** (2.13) untuk keempat aturan dasar *flocking* pada persamaan (2.3), (2.4), (2.5), dan (2.6) dijumlahkan untuk mendapatkan resultan gaya. Secara matematik dapat dituliskan sebagai berikut:

$$\sum \vec{F}_i = \vec{D}_i^a + \vec{D}_i^c + \vec{D}_i^s + \vec{D}_i^e. \quad (3.1)$$

Pada kasus tanpa predator, partikel akan diatur untuk memprioritaskan agar tidak bertabrakan dengan partikel lain ketimbang harus melakukan *flocking*. Sedangkan untuk kasus dengan adanya predator, partikel akan diatur untuk memprioritaskan agar dapat menghindari predator. Untuk membuat partikel dapat memprioritaskan aturan tertentu maka *weighting factor* (ω) yang akan berperan untuk menangani hal tersebut.

$$\sum \vec{F}_i = \omega^a \vec{D}_i^a + \omega^c \vec{D}_i^c + \omega^s \vec{D}_i^s + \omega^e \vec{D}_i^e. \quad (3.2)$$

3.3.1 Interaksi Statis

Maksud dari interaksi statis adalah partikel hanya akan berinteraksi dengan partikel yang satu jenis dengannya. Dengan kata lain tidak akan ada gangguan dari luar yang mempengaruhi pergerakan partikel tersebut. Pada kasus pertama setiap partikel memiliki jari-jari *alignment zone* (z_a) dan *cohesion zone* (z_c) dan *separation zone* (z_s) yang sama. Pada kasus kedua setiap partikel akan memiliki jari-jari z_a yang lebih besar ketimbang z_c dan z_s .

3.3.2 Interaksi Dinamis

Pada kasus ini partikel tidak hanya berinteraksi dengan partikelnya yang sejenis melainkan akan berinteraksi pula dengan partikel pemburu yang disebut predator. Predator diindikasikan dengan warna merah yang akan selalu mengejar partikel biru. Sedangkan partikel biru akan berusaha menyelamatkan diri dari predator. Partikel biru mendekripsi jarak predator menggunakan z_e yang kemudian digunakan untuk menyelamatkan diri dari predator dengan bergerak menjauh dari predator. Jika jarak diantara keduanya lebih kecil dari $d(\lambda_i, q_j) < z_e$.

Adapun dengan hadirnya predator dalam simulasi ini adalah untuk menguji stabilitas partikel dalam berkelompok. *Flocking* yang stabil didefinisikan sebagai kelompok partikel yang tetap bisa menghindari predator namun di saat yang sama harus menjaga eksistensi kelompoknya.

Untuk kasus pertama partikel akan memiliki z_a yang sama besar dengan *separation zone* dengan z_e yang lebih besar. Untuk kasus kedua dan ketiga secara berturut-turut akan bertambah besar hingga z_a sama besar dengan z_e seperti yang terlihat pada **Gambar 2.3**.

3.4 Simulasi Komputer

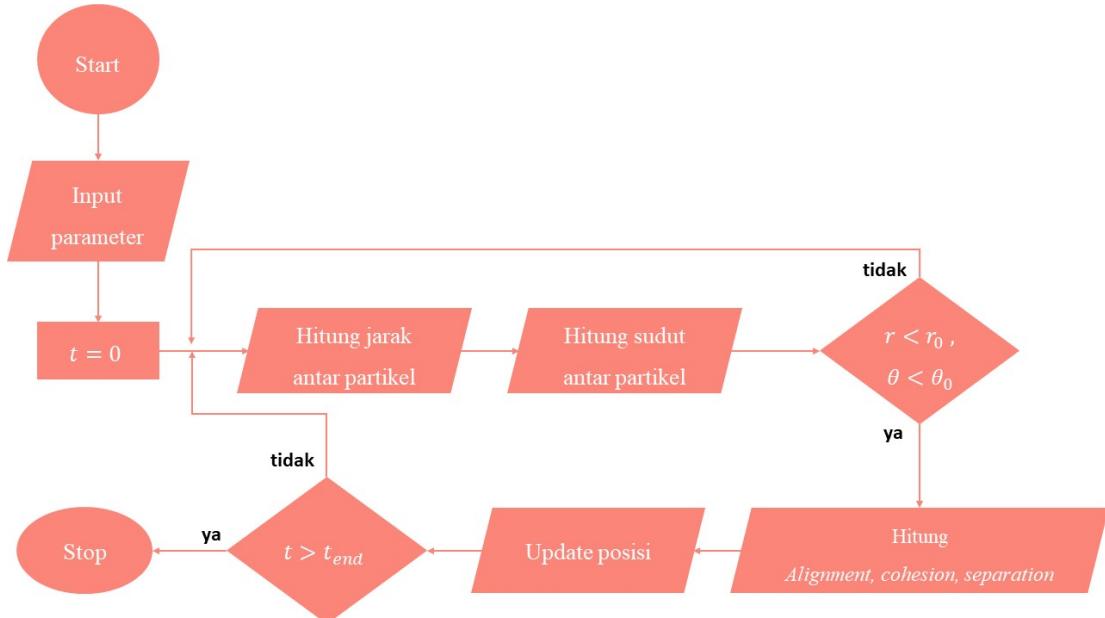
Simulasi dibangun berdasarkan bahasa pemrograman *JavaScript* sebagai bahasa utama dalam mengolah logika perhitungan. Visualisasi data disajikan dalam bentuk animasi dan grafik yang dibangun menggunakan *Hyper Text Markup Language* (HTML) dan *Cascading Style Sheet* (CSS). Kombinasi tiga komponen ini menghasilkan simulasi yang mudah diakses melalui browser (*Google Chrome*, *Mozilla*

Firefox). Berdasarkan (McCarthy *et al.*, 2016) P5js adalah sebuah *library* berbasis bahasa pemrograman *JavaScript*. Kemudahan yang ditawarkan *library* ini adalah ketika ingin menganimasikan suatu obyek dan memanggil fungsi-fungsi yang berguna dalam melakukan perhitungan vektor. Jika dibandingkan penulisan code menggunakan P5js dengan tanpa P5js akan terlihat seperti pada **Tabel 3.3**.

Tabel 3.3: Perbandingan penulisan *code* menggunakan dan tanpa *library* P5js.

Menggunakan P5js	Tanpa P5js
<pre>createCanvas(100, 50) arc(50, 55, 60, 60, 0, PI);</pre>	<pre>ctx.beginPath(); ctx.arc(95, 50, 40, 0, 2 * Math.PI); ctx.stroke();</pre>

Menimbang kemudahan dan alokasi *memory* komputer, maka metode integrasi yang digunakan adalah metode Euler. Hal ini dikarenakan jika menggunakan metode integrasi Euler, komputer tidak akan dibebani oleh banyaknya perhitungan yang harus dilakukan dalam sekali *looping*. Adapun algoritma pemrograman yang digunakan dijelaskan oleh **Gambar 3.2**



Gambar 3.2: Algoritma pemrograman.

Ketika tahap input parameter, dikarenakan simulasi yang dibangun adalah sistem benda banyak maka nilai dari setiap parameter yang akan dimasukan sangat sensi-

tif. Jika satu parameter saja dirubah nilainya walaupun kecil, hal ini akan mempengaruhi kinerja seluruh sistem. Ketika simulasi dimulai dari $t = 0$, setiap partikel yang dipanggil akan berusaha untuk menghitung jarak dirinya dengan semua partikel yang ada. Hal yang sama juga dilakukan ketika menghitung besar sudut antar partikel. Jika syarat jarak dan sudut dipenuhi maka hal ini mengindikasikan bahwa partikel tetangga berada di dalam jangkauan partikel ke- i . Ketika syarat terpenuhi maka partikel akan segera mempertimbangkan akan bergerak kemana berdasarkan akumulasi dari aturan dasar *flocking*. Kemudian hasil dari perhitungan tersebut digunakan untuk memperbaharui posisi partikel. Tahapan-tahapan tersebut akan terus diulang sampai batas waktu tertentu. Ketika mencapai batas waktu t_{end} maka simulasi akan berhenti. Adapun data perhitungan ditampilkan pada layar secara *real-time*.

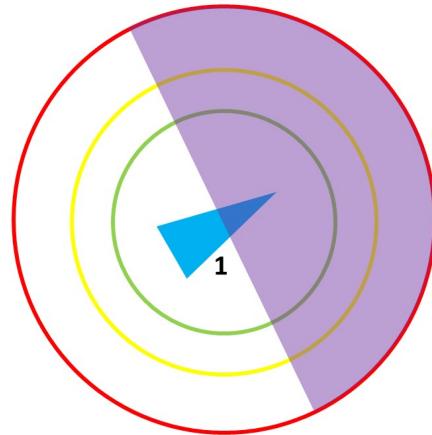
3.5 Penentuan Parameter Gerak

Dalam mengamati hasil simulasi komputer perlu dilakukannya penentuan parameter gerak secara kuantitatif sebagai indikasi kapan dapat dikatakan partikel berhasil membentuk formasi/kelompok. Maka dari itu konfigurasi besar jari-jari z_a akan berbeda sebagai upaya mendapatkan pola data yang baik. Niali z_a yang bervariasi ini akan berakibat pada jumlah kelompok yang terbentuk beserta jumlah anggota disetiap kelompoknya.

Ketika partikel mendeksi partikel tentangga yang jaraknya lebih kecil dari *attraction zone* ($d(\lambda_i, q_j) < z_a$) dan berada di dalam *peripheral view* (ϑ), maka jumlah partikel yang memenuhi syarat tersebut akan disimpan dalam sebuah variabel. Dengan kata lain, nilai inilah yang menunjukan berapa jumlah partikel yang berinteraksi dengan partikel ke- i . Kemudian untuk mengetahui partikel ke- i berada di kelompok mana, setiap partikel diberi nama/ID berdasarkan urutan partikel tersebut di dalam *array* seperti pada **Gambar 3.3**. Adapun definisi sebuah kelompok adalah ketika terdapat irisan ID partikel ke- i pada setiap *array*.

Namun jika logika tersebut diterapkan sebagai pembernanar bahwa partikel berhasil membentuk kelompok, dalam simulasi bisa saja partikel ke- i sedang berada di dalam radius partikel ke- j namun partikel ke- i nyatanya tidak sedang berkelompok/menolak untuk berkelompok. Hal ini dapat diatasi dengan syarat setidaknya aturan dasar *alignment* diterapkan dalam simulasi. Ketika *alignment* diterapkan

maka secara otomatis setiap partikel yang mendeteksi partikel lain masuk ke dalam radius persepsinya, partikel tersebut akan memprioritaskan untuk berkelompok.

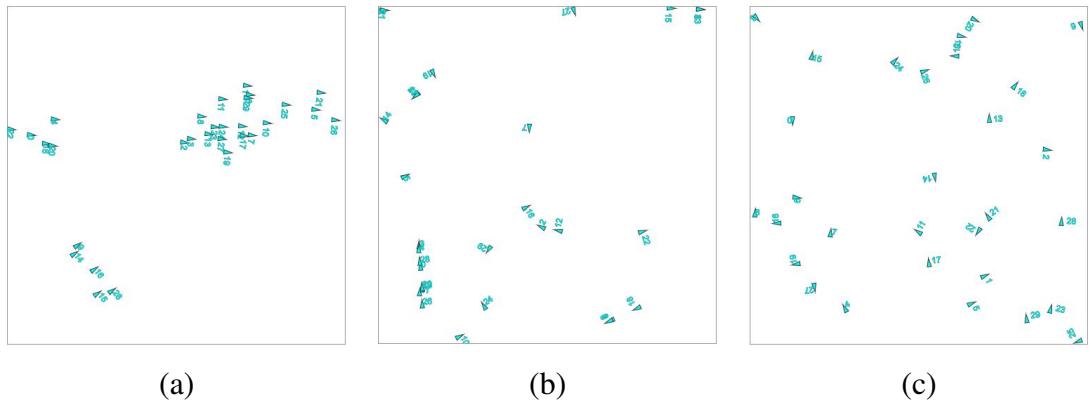


Gambar 3.3: Model partikel dengan ID.

BAB 4

HASIL DAN PEMBAHASAN

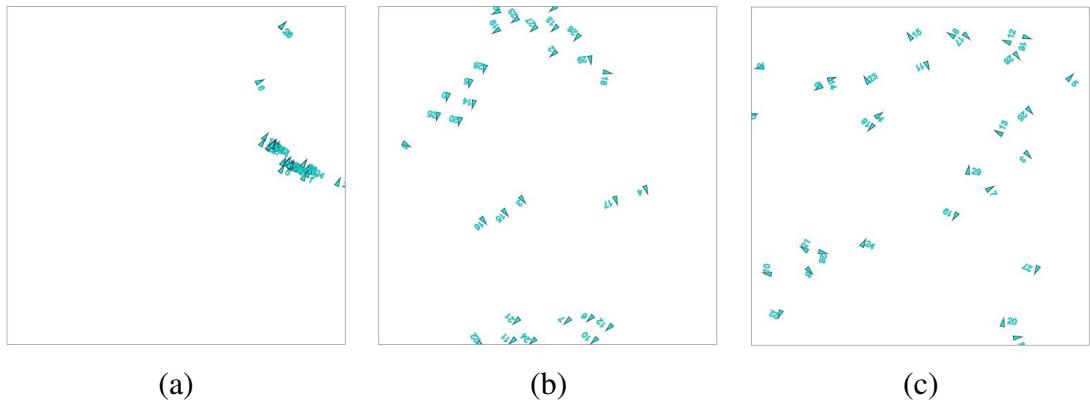
4.1 Sebagian Aturan



Gambar 4.1: Hanya satu aturan *flocking* yang diterapkan dengan $N = 30$, $z_a = 0.8$, $z_c = 0.8$, $z_s = 0.5$, dan $t = 300$. (a) hanya *alignment*, (b) hanya *cohesion*, dan (c) hanya *separation*.

Pada **Gambar 4.1** terdapat perbedaan formasi yang cukup mencolok. Ketika hanya aturan *alignment* yang diterapkan pada **Gambar 4.1(a)**, partikel mampu membentuk kelompoknya masing-masing dan bergerak bersama-sama. Disisi lain pada **Gambar 4.1(b)** tidak terlihat ada kelompok yang terbentuk, namun ada beberapa partikel yang saling menumpuk. Penumpukan ini diakibatkan karena partikel berusaha mencari titik pusat massa dari partikel tetangganya yang kemudian membuat setiap partikel menuju ke titik yang sama. Sedangkan pada **Gambar 4.1(c)** sama

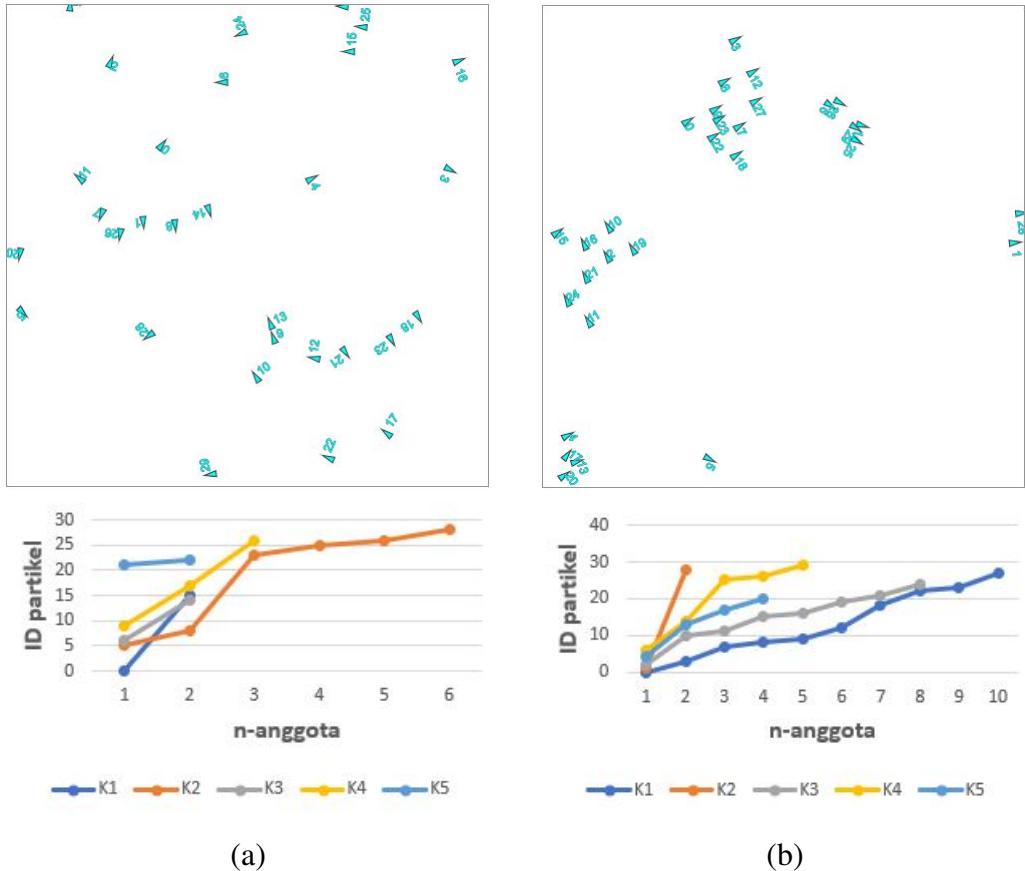
sekali tidak ada partikel yang membentuk kelompok karena partikel hanya mampu saling menjaga jarak agar tidak bertabrakan ketimbang untuk berkelompok. Merujuk pada (Levine *et al.*, 2001) yang membuat simulasi flocking minimal. Minimal dalam artian hanya menggunakan aturan *alignment* saja. Maka pada penelitian ini didapat pula hasil yang tidak jauh berbeda bahwa jika ingin membuat partikel yang mampu melakukan *flocking*, aturan yang mestinya ada adalah *alignment*.



Gambar 4.2: Hanya dua aturan *flocking* yang diterapkan dengan $N = 30$, $z_a = 0.8$, $z_c = 0.8$, $z_s = 0.5$, dan $t = 300$. (a) *alignment* dan *cohesion*, (b) *alignment* dan *separation*, (c) *cohesion* dan *separation*.

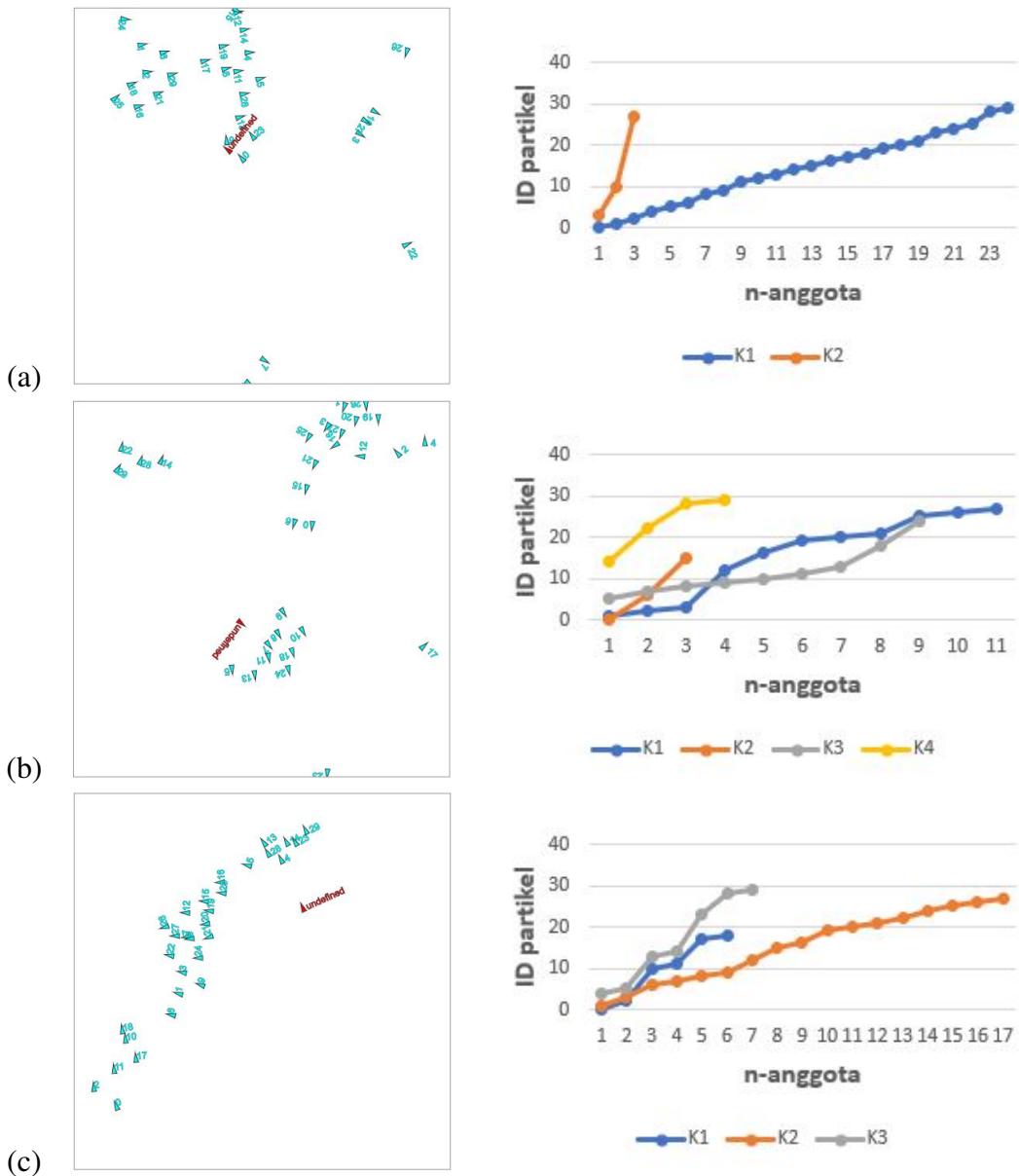
Terlihat bahwa dua aturan yang dikombinasikan pada **Gambar 4.2(a)** yaitu *alignment* dan *cohesion*. Partikel berhasil membentuk kelompok, namun karena *alignment* dikombinasikan dengan *cohesion* tanpa ada aturan yang membuat mereka untuk menjaga jarak satu sama lain, maka partikelpun saling tumpang tindih. Hal ini berkebalikan dengan **Gambar 4.2(b)** yang mana selain partikel dapat berkelompok namun juga tetap menjaga jarak dengan tetangganya sehingga tidak ada tabrakan. Adapun hal yang serupa telah dilakukan oleh (Chaté *et al.*, 2008a) yang simulasinya tidak menerapkan aturan kohesi. Sedangkan pada **Gambar 4.2(c)** tidak ada partikel yang berhasil membentuk kelompok.

4.2 Kombinasi Empat Aturan



Gambar 4.3: Interaksi tanpa pedator dengan $N = 30$. (a) $z_a = 0.5$, $z_c = 0.5$, dan $z_s = 0.5$ (b) $z_a = 0.8$, $z_c = 0.8$, dan $z_s = 0.5$.

Perbedaan yang terdapat dari **Gambar** 4.3 adalah partikel yang memiliki dua *perception zone* dengan ukuran berbeda terlihat lebih optimal dalam membentuk kelompok. Seperti yang dilakukan oleh (Huth & Wissel, 1992) yang mana membuat dua jenis *perception zone* dengan ukuran berbeda menghasilkan pergerakan partikel untuk membentuk kelompok lebih optimal. Pada **Gambar** 4.3(b) partikel memiliki jari-jari z_a dan z_c yang lebih besar, maka kecenderungan untuk berkelompok lebih besar juga. Terdapat 5 kelompok yang terbentuk dengan rata-rata jumlah kelompok yang lebih besar ketimbang **Gambar** 4.3(a). Hal ini sewajarnya terjadi dikarenakan konfigurasi dengan besar jari-jari z_a , z_c dan z_s yang sama besar membuat partikel kesulitan untuk berkelompok.



Gambar 4.4: Interaksi dengan pedator. $N = 30$. (a) $z_a = 1$, $z_c = 1$, $z_s = 0.5$, dan $z_e = 0.2$. (b) $z_a = 1$, $z_c = 1$, $z_s = 0.5$, dan $z_e = 0.8$. (c) $z_a = 1$, $z_c = 1$, $z_s = 0.5$, dan $z_e = 1.4$.

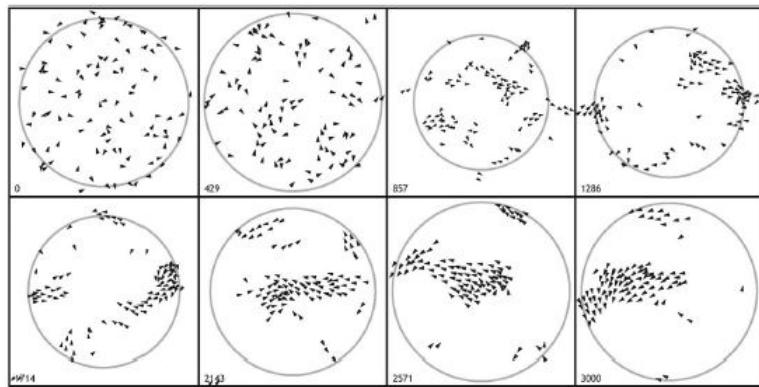
Predator yang ditandai dengan partikel berwarna merah selalu mencoba mengejar partikel biru yang jaraknya paling dekat dengannya. Dalam kasus ini meskipun terdapat gangguan dari predator, partikel biru harus tetap bergerak dalam kelompok dan pada saat yang sama, partikel biru juga harus berusaha melarikan diri dari predator. Pada kasus ini, jari-jari z_e diatur dalam tiga kondisi. Pada kondisi perta-

ma, z_e bernilai 0.2 (terkecil) jika dibandingkan dengan *perception zone* yang lain. Konfigurasi ini menghasilkan partikel yang hanya fokus untuk berkelompok. jika diperhatikan pada **Gambar 4.4(a)**, terbentuk dua kelompok dengan kelompok terbesar memiliki jumlah anggota sebanyak 22 anggota. Disisi lain, kemampuan partikel untuk mendeteksi predator sangat rendah. Hal ini dapat terlihat bahwa posisi predator berada diantara partikel biru.

Pada **Gambar 4.4(b)**, jari-jari z_e bernilai 0.8 atau lebih besar dari z_s namun lebih kecil dari z_a dan z_c . Dengan konfigurasi seperti ini, partikel biru mampu membentuk empat kelompok dengan jumlah anggota tidak kurang dari dua. Adapun kelompok satu memiliki jumlah anggota terbanyak yakni 10 anggota. Partikel biru mampu menghindari kejaran predator dan tetap berusaha bergerak dalam kelompok.

Pada kasus ketiga konfigurasi jari-jari z_e bernilai paling besar ketimbang *perception zone* yang lain yakni sebesar 1.4. Maka seharusnya dengan konfigurasi seperti ini, partikel biru mampu bergerak secara berkelompok sekaligus menjaga jarak dari predator dengan baik. Pada **Gambar 4.4(c)** terdapat tiga kelompok yang terbentuk dengan kelompok kedua yang memiliki jumlah anggota terbanyak yaitu 16 anggota. Terlihat ketiga kelompok mampu bergerak menghindari kejaran predator, jarak yang dihasilkan cukup jauh jika dibandingkan dengan kasus sebelumnya. Hal ini dikarenakan nilai z_e yang besar membuat partikel biru lebih berhati-hati ketika berkelompok.

Jika ketiga kasus tersebut dibandingkan satu sama lain untuk menentukan konfigurasi yang lebih optimal untuk partikel biru berkelompok dan menghindari kejaran predator. Maka konfigurasi pada **Gambar 4.4(c)** yang memiliki z_e nilai terbesar menjadi yang paling optimal. Pola konfigurasi tersebut mirip dengan apa yang telah dilakukan oleh (Iliass *et al.*, 2016) yang melakukan percobaan dengan mengubah nilai z_e dari $z_e = 0$ hingga $z_e = L$.



Gambar 4.5: Hasil simulasi yang telah dilakukan oleh Lebar Bajec.

Hasil penelitian yang telah dilakukan oleh (Bajec *et al.*, 2007) ditunjukkan oleh **Gambar 4.5**. Lingkungan yang dimiliki partikel berbentuk lingkaran, Ketika partikel melewati batas tersebut maka akan ada gaya yang memaksanya bergerak kembali ke dalam. Jika dibandingkan antara hasil pada **Gambar 4.3** dengan **Gambar 4.5**, maka tidak jauh berbeda. Hal ini terlihat dari pola gerak partikel yang pada awalnya tidak berkelompok kemudian menjadi berkelompok dengan beberapa kelompok kecil dan satu kelompok terbesar.

BAB 5

PENUTUP

5.1 Kesimpulan

Berdasarkan simulasi yang telah dilakukan, didapat data pergerakan partikel dengan beberapa konfigurasi. Adapun konfigurasi minimal agar terjadi *flocking* adalah cukup dengan menerapkan aturan *alignment* saja. Pergerakan yang buruk didapat ketika mensimulasikan *flocking* tanpa menggunakan *alignment*, baik itu hanya satu aturan maupun dua aturan yang diterapkan. Jika berkaca pada hasil yang didapat, agar membentuk *flocking* yang baik maka kombinasi empat aturan dasar *flocking* sangat diperlukan. Partikel mampu bergerak dengan baik, membentuk kelompok dan tetap menjaga jarak agar tidak saling bertabrakan.

Pergerakan partikel akan lebih optimal jika diberi *weighting factor* yang mengatur agar kemampuan untuk menjaga jarak yang diprioritaskan. Dari percobaan yang telah dilakukan sebanyak tiga kali, konfigurasi *weighting factor* yang optimal adalah $\omega^s > \omega^a > \omega^c$. Hasil dari konfigurasi tersebut serupa dengan yang dipaparkan oleh (Frame & Flake, 2000). Adapun untuk kasus interaksi dinamis yang menghadirkan predator, kemampuan untuk menyelamatkan diri menjadi prioritas yang utama. Maka konfigurasinya berubah menjadi $\omega^e > \omega^s > \omega^a > \omega^c$.

Konfigurasi yang dilakukan pada *perception zone* secara intens memiliki dampak yang cukup besar terhadap perubahan pola gerak partikel. Secara umum, jika partikel memiliki jari-jari *attraction zone* yang cukup besar, maka partikel akan mampu melakukan *flocking* dengan cukup baik. Karena fungsi utama *attraction zone* adalah untuk membentuk kelompok. Didapat kesimpulan bahwa dengan mengatur jari-jari

z_a dan z_c lebih besar dari z_s akan menghasilkan partikel yang mampu bergerak lebih baik. Hal ini dikarenakan partikel akan lebih memprioritaskan untuk bergerak dalam kelompok. Adapun untuk kasus dimana dihadirkan predator sebagai sosok berbahaya yang membuat partikel lain berusaha untuk menghindar. Konfigurasi *perception zone* $z_e > z_a = z_c > z_s$ menjadi konfigurasi yang paling optimal. Karena keputusan partikel untuk memprioritaskan eksistensinya membuatnya akan lebih berhati-hati ketika melakukan *flocking*.

5.2 Saran

Penelitian ini hanya berfokus pada pembuatan mekanisme dasar agar terjadi *flocking*. Faktor-faktor fisis seperti gesekan partikel dengan medium yang dilaluinya tidak turut diperhitungkan. Adapun data yang diperoleh dari simulasi belum dibandingkan dengan data hasil eksperimen. Maka dari itu untuk mendapatkan data yang lebih valid, pengambilan data dari eksperimen perlu dipertimbangkan seperti pada penelitian (Bhattacharya & Vicsek, 2010)(Niwa, 1996)(Huth & Wissel, 1994).

Dari sisi simulasi, sistem yang dibangun masih dalam bentuk dua dimensi. Akian menjadi sebuah pendekatan yang lebih baik jika simulasi dapat dibangun dalam bentuk tiga dimensi seperti yang dilakukan oleh (Czirók & Vicsek, 2000). Terdapat keterbatasan jumlah partikel yang digambar. Jika partikel yang Digambar terlalu banyak maka simulasi akan berjalan sangat lambat. Penulis menyarankan menggunakan metode optimasi seperti metode *Quad-Tree* (Raynaud, 2018) yang mampu melokalisasi setiap *pixel* yang ditampilkan pada *browser*.

DAFTAR PUSTAKA

- Baglietto, Gabriel, & Albano, Ezequiel V. 2009. Computer simulations of the collective displacement of self-propelled agents. *Computer Physics Communications*, **180**(4), 527–531.
- Bajec, I Lebar, Zimic, Nikolaj, & Mraz, Miha. 2007. The computational beauty of flocking: boids revisited. *Mathematical and Computer Modelling of Dynamical Systems*, **13**(4), 331–347.
- Bhattacharya, K., & Vicsek, Tamás. 2010. Collective decision making in cohesive flocks. *New Journal of Physics*, **12**.
- Chaté, H., Ginelli, F., Grégoire, G., Peruani, F., & Raynaud, F. 2008a. Modeling collective motion: Variations on the Vicsek model. *European Physical Journal B*, **64**(3-4), 451–456.
- Chaté, Hugues, Ginelli, Francesco, Grégoire, Guillaume, & Raynaud, Franck. 2008b. Collective motion of self-propelled particles interacting without cohesion. *Physical Review E - Statistical, Nonlinear, and Soft Matter Physics*, **77**(4), 1–15.
- Couzin, Iain D, Krause, Jens, James, Richard, Ruxton, Graeme D, & Franks, Nigel R. 2002. Collective Memory and Spatial Sorting in Animal Groups. 1–11.
- Czirók, András, & Vicsek, Tamás. 2000. Czirók Collective behavior of interacting self-propelled particles PA 2000.pdf. **281**, 17–29.
- Czirók, András, Vicsek, Mária, & Vicsek, Tamás. 1999. Collective motion of organisms in three dimensions. *Physica A: Statistical Mechanics and its Applications*, **264**(1-2), 299–304.

- D'Orsogna, M. R., Chuang, Y. L., Bertozzi, A. L., & Chayes, L. S. 2006. Self-propelled particles with soft-core interactions: Patterns, stability, and collapse. *Physical Review Letters*, **96**(10).
- Frame, Michael, & Flake, Gary William. 2000. The Computational Beauty of Nature: Computer Explorations of Fractals, Chaos, Complex Systems, and Adaptation. *The American Mathematical Monthly*, **107**(6), 576.
- Ginelli, Francesco. 2016. The Physics of the Vicsek model. *European Physical Journal: Special Topics*, **225**(11-12), 2099–2117.
- Huth, Andreas, & Wissel, Christian. 1992. The simulation of the movement of fish schools. *Journal of Theoretical Biology*, **156**(3), 365–385.
- Huth, Andreas, & Wissel, Christian. 1994. The simulation of fish schools in comparison with experimental data. *Ecological Modelling*, **75-76**(C), 135–146.
- Iliass, Tarras, & Cambui, Dorilson. 2016. The combined effect of attraction and orientation zones in 2D flocking models. *International Journal of Modern Physics B*, **30**(4), 1650002.
- Iliass, Tarras, Cambui, Dorilson, Grosso, Mato, & Grosso, Mato. 2016. The combined effect of attraction and orientation zones in 2D flocking models. **30**(4), 1–11.
- Levine, Herbert, Rappel, Wouter Jan, & Cohen, Inon. 2001. Self-organization in systems of self-propelled particles. *Physical review. E, Statistical, nonlinear, and soft matter physics*, **63**(1), 4.
- Li, Yue-xian, Lukeman, Ryan, & Edelstein-keshet, Leah. 2008. Minimal mechanisms for school formation in self-propelled particles. **237**, 699–720.
- Li Chuang, Yao, D'Orsogna, Maria R., Marthaler, Daniel, Bertozzi, Andrea L., & Chayes, Lincoln S. 2007. State transitions and the continuum limit for a 2D interacting, self-propelled particle system. *Physica D: Nonlinear Phenomena*, **232**(1), 33–47.
- M. Brown, Joshua; Bossomaier, Terry. 2013. Flock Stability in the Vicsek Model. **8076**, 89–102.

- McCarthy, Lauren, Reas, Casey, & Fry, Ben. 2016. *Make: Getting Started with P5js*. San Francisco: Maker Media, Inc.
- Mogilner, A., Edelstein-Keshet, L., Bent, L., & Spiros, A. 2003. *Mutual interactions, potentials, and individual distance in a social aggregation*. Vol. 47. Springer.
- Niwa, Hiro-Sato. 1994. *Niwa 1994 (J Theor Biol)*.
- Niwa, Hiro Sato. 1996. Newtonian dynamical approach to fish schooling. *Journal of Theoretical Biology*, **181**(1), 47–63.
- Raynaud, Sebastien. 2018. *3D Boids with QuadTree*.
- Reynolds, Craig W. 1987. Flocks, Herds, and Schools: A Distributed Behavioral Model, in Computer Graphics. *ACM SIGGRAPH Computer Graphics*, **21**(4), 25–34.
- Shiffman, Daniel. 2012. *The Nature of Code*. Daniel Shiffman.
- Silvester, John R. 2000. Determinants of Block Matrices on JSTOR. *The Mathematical Gazette*, **84**(501), 460.
- Toner, John. 1998. Flocks , herds , and schools : A quantitative theory of flocking. **58**(4), 4828–4858.
- Toner, John, & Tu, Yuhai. 1995. Long-range order in a two-dimensional dynamical XY model: How birds fly together. *Physical Review Letters*, **75**(23), 4326–4329.
- Vicsek, Tamás, & Zafeiris, Anna. 2012. Collective motion. *Physics Reports*, **517**(3-4), 71–140.
- Vicsek, Tamás, Czirák, András, Ben-Jacob, Eshel, Cohen, Inon, & Shochet, Ofer. 1995. NOVEL TYPE OF PHASE TRANSITION IN A SYSTEM OF SELF-DRIVEN PARTICLES. *Physical Review Letters*, **75**(1226).

INDEX

A

Alignment **bab** 2;

B

Boids **bab** 1;

C

Craig Reynolds **bab** 1; *Collision Rule* **bab** 2; *Current State Properties* **bab** 2; *Cohesion* **bab** 2; CSS **bab** 3;

E

Escaping **bab** 2; Euler **bab** 2;

F

Flocking **bab** 1;

H

Herd **bab** 1; HTML **bab** 3;

J

JavaScript **bab** 3;

O

OOP **bab** 1; *Order Paramter* **bab** 2;

P

Predator **bab** 1; *Perception Zone* **bab** 2; *Peripheral View* **bab** 2; *p5js* **bab** 2;

Q

Quad-Tree **bab 5;**

S

School **bab 1;** *SPPs* **bab 2;** *Separation* **bab 2;** *Steering Behavior* **bab 2;** *Step Size* **bab 2;**

V

Vicsek **bab 1;**

W

Weighting Factor **bab 2;**

Lampiran A

Kode Program

A.1 Kode Program index.html

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="UTF-8" />
    <meta http-equiv="X-UA-Compatible" content="IE=edge" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0" />
    <title>school2D</title>

    <script src="libraries/p5.js"></script>
    <script src="libraries/p5.dom.js"></script>
    <!-- <script src="libraries/p5.sound.js"></script> -->
    <script src="sketch.js"></script>
    <script src="loopedThings.js"></script>
    <script src="boid.js"></script>
    <script src="flock.js"></script>

  <style>
    body {
      margin: 0;
      padding: 0;
      overflow: hidden;
    }
  </style>

```

```

}

.inputDiv {
    float: left;
    width: 250px;
    height: 170px;
    background: #eee;
    margin-top: 10px;
}

canvas {
    float: left;
    border: 1px solid rgb(148, 148, 148);
    margin-top: 10px;
    margin-left: 30px;
}

.outputDiv {
    float: left;
}

}

.inputs {
    list-style-type: none;
    font-size: 15px;
}

</style>
</head>
<body>
    <div class="inputDiv">
        <div class="inputs">
            <table>
                <tr>
                    <td id="bI"></td>
                    <td>Number of boid</td>
                </tr>
                <tr>
                    <td id="aS"></td>
                    <td>Alignment</td>
                </tr>

```

```

<tr>
    <td id="cS"></td>
    <td>Cohesion</td>
</tr>
<tr>
    <td id="sS"></td>
    <td>Separation</td>
</tr>
<tr>
    <td id="dC"></td>
    <td>Debug</td>
</tr>
</table>
</div>
<div class="mainButton">
    <button id="start" type="button" value="Start" onclick="buttonClick()">
        Start
    </button>
</button>
    <button name="btStop" id="stop" class="button" onclick="buttonClick()">
        Stop
    </button>
    <button name="reRun" id="rerun" class="button" onclick="buttonClick()">
        Rerun
    </button>
    <button
        id="restart"
        type="button"
        value="Restart"
        onclick="buttonClick()">
        Restart
    </button>
</div>
<div class="outputDiv">
</div>

```

```
<div class="canvasDiv"></div>

<textarea name="taOut" id="taOut" cols="33" rows="10"></textarea>
<textarea name="taOut2" id="taOut2" cols="33" rows="10"></textarea>
</div>
</body>
</html>
```

A.2 Kode Program sketch.js

```
// Global
var canvas;
const cvx = 480;
const cvy = 480;
var XMAX = cvx * 2;
var YMAX = cvy * 2;
var xyMAX = 8;

var fps = 60;
var trigger = false;
var tBegin = 0;
var tEnd = 300;
var t = tBegin;
var dt = 1;
var tData = 10;
var tPredator = 60;
var iter = 0;
var Niter = Math.floor(tData / dt);

var taOut;
var taOut2;
var debug;
var boidInput, alignSlider, cohesionSlider, separationSlider;

var N;
var neighbors = [];
var boids = [];
var hunters = [];
var guardians = [];

function setup() {
  frameRate(fps);
  noLoop();
```

```

var cD = select('.canvasDiv');
cD.child(canvas);
canvas = createCanvas(cx, cy);
// canvas.style("z-index", "-1");
var bI = select('#bI');
var aS = select('#aS');
var cS = select('#cS');
var sS = select('#sS');
var dC = select('#dC');
boidInput = createInput();
boidInput.value(30);
boidInput.style('width:130px');
alignSlider = createSlider(0, 2, 1, 0.01);
cohesionSlider = createSlider(0, 2, 0.5, 0.01);
separationSlider = createSlider(0, 2, 1.2, 0.01);
debug = createCheckbox();
bI.child(boidInput);
aS.child(alignSlider);
cS.child(cohesionSlider);
sS.child(separationSlider);
dC.child(debug);
}

function buttonClick() {
var id = event.target.id;
switch (id) {
case 'start':
N = boidInput.value();
for (var i = 0; i < N; i++) {
let x = random(0, 8);
let y = random(0, 8);
boids.push(new Boid(x, y, 0.03, 0.8, 0.5, 1, i));
}
trigger = true;
loop();
break;
}
}

```

```

        case 'stop':
            noLoop();
            // trigger = false;
            break;
        case 'rerun':
            loop();
            trigger = true;
            break;
        case 'restart':
            restart();
            break;
    }
}

function keyPressed() {
    if (key === 'p') {
        for (var i = 0; i < 1; i++) {
            hunters.push(new Boid());
        }
    } else if (key === 'o') {
        hunters.splice(i, 1);
    }
}

function restart() {
    taOut.value = '';
    t = 0;
    N = boidInput.value();
    canvas.resize(cx, cy);
    background(255);

    boids = [];
    for (var i = 0; i < N; i++) {
        let x = random(0, 8);
        let y = random(0, 8);
        boids.push(new Boid(x, y));
    }
}

```

```
}

hunters = [];

// A Special function to looping something over and over again automatically
function draw() {
    background(255);
    simulate();
}
```

A.3 Kode Program `loopedThings.js`

```
function simulate() {
    if (trigger == true) {
        if (iter >= Niter) {
            iter = 0;
        }

        if (t == 60) {
            for (var i = 0; i < 1; i++) {
                hunters.push(new Boid(0, 0, 0.03));
            }
        }

        showData();
        boidBuilder();

        if (t >= tEnd) {
            noLoop();
        }

        iter++;
        t += dt;
    }
}

function showData() {
    // Showing data to textarea
    taOut = document.getElementById('taOut');
    var xsum, ysum, vxsum, vysum;
    if (boids.length > 0) {
        taOut.value += t + ' ';
        xsum = 0;
        ysum = 0;
        vxsum = 0;
        vysum = 0;
    }
}
```

```

}

var taOut2 = document.getElementById('taOut2');
var str = t + '\n';
for (var i = boids.length - 1; i >= 0; i--) {
    str += boids[i].id;
    str += ', ';
    str += boids[i].neighbors;
    str += '\n';
}

str += '\n';
taOut2.value += str;
taOut2.scrollTop = taOut2.scrollHeight;

for (var i = 0; i < boids.length; i++) {
    var xx = boids[i].pos.x;
    var yy = boids[i].pos.y;
    var vxx = boids[i].vel.x;
    var vyy = boids[i].vel.y;

    var x = parseFloat(Math.round(xx * 100) / 100).toFixed(3);
    var y = parseFloat(Math.round(yy * 100) / 100).toFixed(3);
    var vx = parseFloat(Math.round(vxx * 100) / 100).toFixed(3);
    var vy = parseFloat(Math.round(vyy * 100) / 100).toFixed(3);
    var text = x + ' ' + y + ' ' + vx + ' ' + vy + '\n';
    //taOut.value += text;
    //taOut.scrollTop = taOut.scrollHeight;

    xsum += parseFloat(x);
    ysum += parseFloat(y);
    vxsum += parseFloat(vx);
    vysum += parseFloat(vy);
}

if (boids.length > 0) {
    var xavg = (xsum / boids.length).toFixed(3);
}

```

```

        var yavg = (ysum / boids.length).toFixed(3);
        var vxavg = (vxsum / boids.length).toFixed(3);
        var vyavg = (vysum / boids.length).toFixed(3);

        taOut.value += xavg + ' ';
        taOut.value += yavg + ' ';
        taOut.value += vxavg + ' ';
        taOut.value += vyavg + ' ';
        // var xx = sqrt((xavg ^ 2) + (yavg ^ 2)).toFixed(3);
        // var vv = sqrt((vxavg ^ 2) + (vyavg ^ 2));

        // taOut.value += xx + " ";
        // taOut.value += vv + " ";
        taOut.value += '\n';
        taOut.scrollTop = taOut.scrollHeight;
    }
}

function boidBuilder() {
    for (boid of boids) {
        boid.transform();
        boid.edges();
        // boid.boundaries();
        boid.fear(hunters, 1);
        boid.applyFlocks(boids);
        boid.update();
        boid.showBoid([0, 255, 255]);
    }

    for (hunter of hunters) {
        hunter.transform();
        hunter.edges();
        // hunter.boundaries();
        hunter.eat(boids, 0);
        hunter.applyHunt(boids);
        hunter.update();
    }
}

```

```
    hunter.showBoid([255, 0, 0]);  
}  
}
```

A.4 Kode Program boid.js

```
class Boid {
  constructor(
    x,
    y,
    maxSpeed,
    bluePerceptron,
    greenPerceptron,
    redPerceptron,
    id
  ) {
    this.post = createVector(0, 0);
    this.pos = createVector(x, y);
    // this.vel = createVector(1, 0);
    this.vel = p5.Vector.random2D();
    this.vel.setMag(random(0.01, 0.02));
    this.acc = createVector();
    this.mass = 1;
    this.dumping = -0.1;

    this.id = id;
    this.neighbors = [];

    this.maxSpeed = maxSpeed || 0.03;
    this.maxForce = 0.002;
    this.radius;
    this.r = 3;
    this.periphery = PI / 2;
    this.color = color(0, 255, 255);

    //index 0:seek|1:fear|2:radius of align & cohe|3: radius of sep
    this.dna = [];
    this.dna[0] = 0.1;
    this.dna[1] = -2;
    this.dna[2] = bluePerceptron || random(0.5, 8);
```

```

        this.dna[3] = greenPerceptron || random(0.3, 0.5);
        this.dna[4] = redPerceptron || random(0.8, 1);

        this.flock = new Flock(this);
    }

    transform() {
        this.post.x = map(this.pos.x, 0, xyMAX, 0, XMAX);
        this.post.y = map(this.pos.y, 0, xyMAX, YMAX, 0);
        this.ar = map(this.dna[2], 0, xyMAX, 0, XMAX);
        this.rr = map(this.dna[3], 0, xyMAX, 0, XMAX);
        this.er = map(this.dna[4], 0, xyMAX, 0, XMAX);
    }

    edges() {
        if (this.pos.x > xyMAX) {
            this.pos.x = 0;
        } else if (this.pos.x < 0) {
            this.pos.x = xyMAX;
        }
        if (this.pos.y > xyMAX) {
            this.pos.y = 0;
        } else if (this.pos.y < 0) {
            this.pos.y = xyMAX;
        }
    }

    fear(targets, index) {
        let record = Infinity;
        let closest = null;
        let d;
        for (let i = targets.length - 1; i >= 0; i--) {
            d = p5.Vector.dist(targets[i].pos, this.pos);

            if (d < record && d < this.dna[3 + index]) {
                record = d;
            }
        }
    }
}

```

```

        closest = targets[i];
    }

    // if (record < this.radius) {
    // }
}

if (closest) {
    let fear = this.flock.seek(closest);
    fear.mult(this.dna[index]);
    fear.limit(this.maxForce);
    this.summation(fear);
    // return fear;
}
}

eat(targets, index) {
    let record = Infinity;
    let closest = null;
    let d;
    for (let i = targets.length - 1; i >= 0; i--) {
        d = p5.Vector.dist(targets[i].pos, this.pos);

        if (record < 0.3) {
            // targets.splice(closest, 1);
        } else if (d < record) {
            record = d;
            closest = targets[i];
        }
    }
}

if (closest !== null) {
    let eat = this.flock.seek(closest);
    // eat.mult(this.dna[index]);
    eat.limit(this.maxForce);
    // this.summation(eat);
}

```

```

        return eat;
    }
}

applyHunt(targets) {
    let eat = this.eat(targets);
    this.summation(eat);
    // this.acc.add(eat);
}

applyFlocks(boids) {
    let alignment = this.flock.align(boids);
    let cohesion = this.flock.cohesion(boids);
    let separation = this.flock.separation(boids);

    alignment.mult(alignSlider.value());
    cohesion.mult(cohesionSlider.value());
    separation.mult(separationSlider.value());

    // this.acc.add(alignment);
    // this.acc.add(cohesion);
    // this.acc.add(separation);

    this.summation(alignment);
    this.summation(cohesion);
    this.summation(separation);
}

summation(stuff) {
    this.vel.add(stuff);
}

update() {
    // this.vel.add(this.acc);
    this.pos.add(this.vel);
    this.vel.limit(this.maxSpeed);
}

```

```

    // this.acc.mult(0);
}

showBoid(col) {
    let theta = -1 * this.vel.heading() + PI / 2;
    push();
    translate(this.posT.x / 2, this.posT.y / 2);
    rotate(theta);

    fill(col);
    strokeWeight(1);
    stroke(50);
    beginShape();
    vertex(0, -this.r * 2);
    vertex(-this.r, this.r * 2);
    vertex(this.r, this.r * 2);
    endShape(CLOSE);
    text(this.id + '', this.pos.x, this.pos.y);
    pop();

    if (debug.checked()) {
        let currentHeading = -1 * this.vel.heading();

        push();
        translate(this.posT.x / 2, this.posT.y / 2);
        rotate(currentHeading);
        fill(0, 100, 255, 40);
        noStroke();
        arc(0, 0, this.er, this.er, -this.periphery, this.periphery);

        //perception raidii
        stroke([0, 200, 255]);
        noFill();
        ellipse(0, 0, this.ar);
        stroke(200, 255, 0);
        ellipse(0, 0, this.rr);
    }
}

```

```
stroke(255, 0, 70);  
ellipse(0, 0, this.er);  
pop();  
}  
}  
}
```

Lampiran B

Karya Tulis

B.1 Simposium Nasional Inovasi dan Pembelajaran Sains 2018

Pemodelan Gerak Parabola yang Terjun ke Air Menggunakan Metode Euler Berbasis Javascript

Ariq Dhia Irfanudin^{1,a)}, Dinda Ravi Algifar^{1,b)}, Ikhsan Mohammad Noor^{1,c)}
Sparisoma Viridi^{2,d)} Yudha Satya Perkasa^{3,e)}

¹Program Sarjana Fisika,
Kelompok Keilmuan Fisika Nuklir dan Komputasi,
Fakultas Sains dan Teknologi, Universitas Islam Negeri Sunan Gunung Djati Bandung,
Jl. A.H Nasution no. 105 Bandung, Indonesia, 40614

²Laboratorium Fisika Nuklir dan Biofisika,
Kelompok Keilmuan Fisika Nuklir dan Biofisika,
Fakultas Matematika dan Ilmu Pengetahuan Alam, Institut Teknologi Bandung,
Jl. Ganesha no. 10 Bandung, Indonesia, 40132

³Laboratorium Sistem Modeling,
Kelompok Keilmuan Fisika Nuklir Teori,
Fakultas Sains dan Teknologi, Universitas Islam Negeri Sunan Gunung Djati Bandung,
Jl. A.H Nasution 105, Bandung, Indonesia 40614

^{a)}1157030004@student.uinsgd.ac.id
^{b)}dinda.ravi.algivar@gmail.com
^{c)}ikhsanmnoor@gmail.com
^{d)}dudung@fi.itb.ac.id
^{e)}yudha@uinsgd.ac.id

Abstrak

Telah dibuat sebuah program untuk memodelkan fenomena fisis khususnya gerak parabola yang terjun ke air sebagai bahan pembelajaran yang mudah diakses melalui browser. Besaran-besaran seperti posisi awal, kecepatan awal, massa benda dan sudut tembakar dapat diatur sendiri oleh pengguna. Untuk menghasilkan data dan ilustrasi yang mendekati kejadian sebenarnya, ditambahkan faktor gravitasi maupun hambatan udara dan air. Perhitungan dilakukan dengan menggunakan metode Euler untuk mendapatkan nilai perubahan posisi setiap waktunya. Program dan perhitungan disusun dalam bahasa pemrograman javascript dan library p5.js agar dapat diakses dengan mudah melalui browser. Adapun hasil akhir pemodelan ini adalah nilai posisi dan waktu yang ditunjukkan dalam bentuk animasi. Animasi pertama menunjukkan gerak parabola yang hanya bergerak pada medium udara sedangkan animasi kedua ditambahkan medium air. Jika diamati, lintasan yang dihasilkan, pada animasi pertama terlihat lebih lebar ketimbang animasi kedua.

Kata-kata kunci: Pemodelan, Javascript, Gerak parabola, Metode Euler

PENDAHULUAN

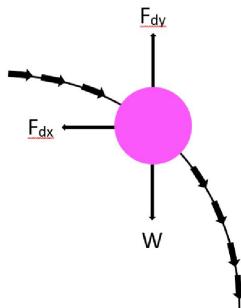
Gerak parabola adalah salah satu kasus klasik yang banyak dimanfaatkan prinsipnya dalam kehidupan sehari-hari, para pembelajar mengamati gerak benda ketika mulai melesat sampai jatuh kembali menuju

tanah. Benda yang bergerak melalui medium udara sudah dapat diprediksi kapan dan dimana benda itu akan mencapai titik tertinggi maupun ketika menyentuh tanah. Namun apa yang akan terjadi ketika benda tersebut tidak hanya melintasi medium udara namun turut melintasi medium air yang memiliki massa jenis berbeda. Hal ini menjadi kasus yang cukup menarik untuk dipelajari dengan harapan dapat menambah nilai manfaat untuk banyak pihak.

Untuk mendapatkan nilai yang mendekati kejadian sebenarnya, digunakan metode perhitungan Euler karena relatif lebih mudah untuk digunakan maupun dipahami. Pemodelan dilakukan menggunakan bahasa pemrograman *javascript*. *Javascript* adalah bahasa pemrograman yang lazim digunakan pada *browser* seperti *mozilla Firefox*, *Google Chrome* dan *Opera*. Untuk memudahkan dan mengefisiensikan *code*, digunakan library *p5.js* dalam pembuatan gambar, animasi dan interaksi. “ Idenya adalah untuk menulis sebaris *code* yang dapat menampilkan sebuah lingkaran pada layar [3].”.

METODOLOGI

Model



Gambar 1. Diagram Gaya

Dalam bahasan kinematika yang mengabaikan penyebab benda bergerak, gerak parabola dihitung berdasarkan hukum ke-2 Newton

$$a = \frac{\sum F}{m} \quad (1)$$

Dimana (*a*) adalah percepatan, (ΣF) adalah total gaya dan (*m*) adalah massa benda. Metode perhitungan dilakukan dengan menggunakan integrasi Euler. Metode integrasi Euler digunakan sebagai alternatif dalam memperhitungkan gerak parabola dalam kondisi yang tidak ideal, dalam kasus ini benda mengalami gesekan dengan medium yang dilewatinya [2]. Dari persamaan (1) akan didapat nilai percepatan yang akan disubstitusikan ke persamaan kecepatan kemudian nilai kecepatan disubstitusikan ke persamaan posisi (GLB dan GLBB).

Total gaya dihitung berdasarkan gaya-gaya yang bekerja pada benda arah x dan y. Untuk total gaya arah x digambarkan dalam persamaan

$$F_{dx} = -\frac{1}{2} \rho v^2 C_d A \cos\theta \quad (2)$$

Sedangkan total gaya pada arah y adalah

$$F_{dy} = \frac{1}{2} \rho v^2 C_d A \sin\theta \quad (3)$$

$$W = -mg \quad (4)$$

$$F_y = F_{dy} + W \quad (5)$$

Persamaan (2) dan (3) disebut dengan *drag force*, atau dalam penamaan lain sering disebut *fluid resistance*. Gaya gesek yang arahnya berlawanan dengan gerak benda terjadi karena benda bergerak melewati suatu

medium baik itu medium udara maupun air [1]. Dimana (ρ) adalah massa jenis medium yang dilewati benda. (C_d) adalah koefisien gesek benda yang nilainya bergantung pada bentuk, kemiringan dan beberapa kondisi aliran [4]. Kemudian percepatan dihitung untuk setiap arahnya menggunakan persamaan

$$a_x = \frac{\sum F_x}{m} \quad (6)$$

$$a_y = \frac{\sum F_y}{m} \quad (7)$$

Nilai percepatan yang didapat kemudian disubstitusikan ke persamaan

$$v_x(t + \Delta t) = v_x(t) + a_x \Delta t \quad (8)$$

$$v_y(t + \Delta t) = v_y(t) + a_y \Delta t \quad (9)$$

Didapatlah nilai kecepatan yang kemudian disubstitusikan ke persamaan

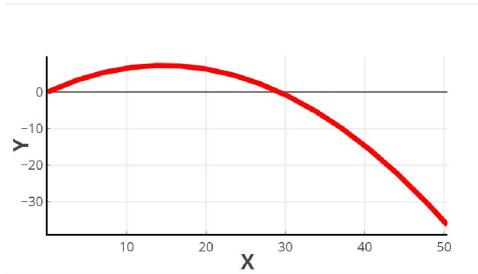
$$x(t + \Delta t) = x(t) + v_x(t) \Delta t \quad (10)$$

$$y(t + \Delta t) = y(t) + v_y(t) \Delta t \quad (11)$$

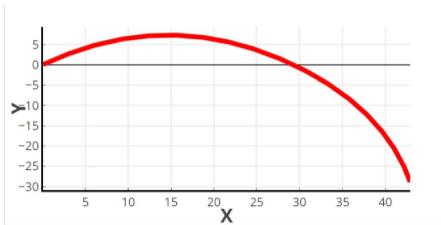
PROSEDUR PERHITUNGAN

Parameter-parameter awal yang telah ditentukan akan dimasukkan ke dalam persamaan yang ada. Untuk mencari nilai percepatan, terlebih dahulu harus dicermati gaya apa saja yang hendak dimasukkan dalam perhitungan. Dalam kasus ini penulis hanya memperhitungkan gaya berat, gravitas, gaya gesek pada medium udara dan air. Dari persamaan (6) sampai (11) disebut dengan metode Integrasi Euler. Hal ini dikarenakan nilai posisi benda didapat dari hasil integral kecepatan dan kecepatan didapat dari hasil integral percepatan. Ketika benda di udara nilai massa jenis pada persamaan (2) dan (3) bernilai 1.275 kg/m^3 sedangkan ketika benda terjun ke air (ketinggian $\leq 0 \text{ m}$) maka massa jenisnya bernilai 1.000 kg/m^3 . Persamaan fisis dihitung di dalam sebuah perulangan dengan pertambahan waktu tertentu yang akan berhenti ketika $t_{\max} = n$. Kemudian nilai posisi yang didapat digunakan untuk menampilkan benda pada browser.

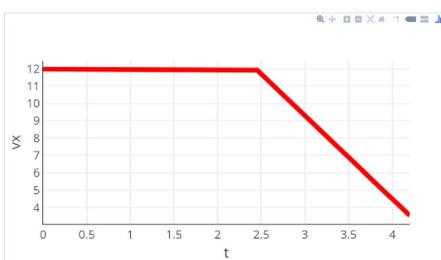
HASIL DAN ANALISIS



Gambar 5. Gerak parabola hanya medium udara



Gambar 6. Gerak Parabola medium udara dan air



Gambar 7. Perubahan kecepatan (v_x) terhadap waktu medium udara dan air

Gerak lurus beraturan (GLB) dalam arah x berubah menjadi gerak lurus berubah beraturan (GLBB), artinya kecepatan arah x tidak lagi konstan dan membuat lintasan tidak lagi berbentuk parabola.[5]

Benda diluncurkan dari ketinggian 0 m di atas permukaan air dengan sudut 45° dan kecepatan 12 m/s. Garis hitam melintang pada $x = 0$ menunjukkan batas medium antara udara dan air. Pada gambar 5, benda diluncurkan tanpa menambahkan medium air sedangkan pada gambar 6 medium air ditambahkan di bawah garis hitam. Terlihat pada gambar 5 lintasan yang dibentuk cenderung lebih lebar ketimbang gambar 6. Karena air memiliki nilai massa jenis yang lebih besar ketimbang udara, hal ini menyebabkan nilai gaya gesek yang dialami benda akan lebih besar ketika berada pada medium air dan mengakibatkan gerak benda pada arah x lebih terhambat.

Perhitungan diulang terus-menerus dengan pertambahan (*step size*) sebesar 0.005 yang akan berhenti pada $t_{max} = 4.2$ detik. Walaupun kedua percobaan tersebut akan berhenti ketika $t_{max} = 4.2$ detik, tetapi posisi benda pada gambar 5 terlihat lebih rendah. Massa jenis air yang lebih besar ketimbang udara mengakibatkan gerak benda lebih terhambat dan membuat benda pada kedua percobaan memiliki posisi yang berbeda.

Dalam kenyataanya, gerak parabola tidak hanya mengalami gaya gesek dan berat saja. Karena hanya faktor gaya gesek medium dan gaya berat saja yang diperhitungkan dalam pemodelan ini. Terdapat batasan nilai pada besaran-besaran awal seperti massa benda yang harus lebih dari 15 kg. Jika massa benda kurang dari 15 kg maka hasilnya tidak akan bagus. Hal ini dapat langsung diamati pada bentuk lintasan yang tidak terlihat seperti parabola. Batasan ini hanya berlaku ketika medium air ditambahkan dalam pemodelan. Dimana perubahan nilai massa jenis medium yang cukup drastis mengakibatkan perubahan gaya yang diterima benda cukup besar.

KESIMPULAN

Dari hasil pemodelan gerak parabola yang terjun ke air berbasis *javascript*, didapat ilustrasi yang dapat menggambarkan perbedaan gerak benda yang hanya bergerak di medium udara ataupun yang terjun ke air. Massa jenis air yang lebih besar ketimbang udara membuat nilai gaya gesek ketika melewati medium air turut menjadi lebih besar dan ini menyebabkan nilai kecepatan (v_x) tidak lagi konstan. Perlambatan kecepatan (v_x) menghasilkan lintasan yang tidak lagi berbentuk parabola.

Hasil yang lebih baik akan didapatkan jika faktor-faktor fisik yang lain turut diperhitungkan dan dengan menggunakan metode perhitungan yang lebih baik.

UCAPAN TERIMA KASIH

Penulis mengucapkan terima kasih kepada berbagai pihak yang telah membantu dalam penulisan makalah ini.

REFERENSI

1. Daniel Shiffman. *The Nature of Code*, ITP, New York (2012)
2. Jonathan Thomas-Palmer, *A Brief Look at the Force of Drag using Numerical Modeling (or The Euler Method)*, (Tersedia di <https://www.flippingphysics.com/the-euler-method.html>, diakses pada 7 April 2018)
3. Lauran McCarthy, Casey Reas & Ben Fry, *Getting Started with p5.js*.San Francisco, United State of America (2016)
4. National Aeronautics and Space Administration (NASA), *Shape effects on Drag*, (Tersedia di <https://www.grc.nasa.gov/www/K-12/airplane/shaped.html>, diakses pada 7 April 2018)
5. Purwadi dan Ishafit, *Pemodelan Gerak Parabola yang Dipengaruhi Seretan serta Spin Efek Magnus Bola dengan program Modellus dan Excell*, JRKPF UAD Vol.I No.1 (2014)

B.2 Simulasi Peracunan Produk Fisi pada Teras Reaktor Nuklir Berbasis JavaScript

Simulasi Peracunan Produk Fisi pada Teras Reaktor Nuklir Berbasis JavaScript

**Ariq Dhia Irfanudin^{1*}, Dinda Ravi Algifari¹, Mochammad Nurul Subkhi²
Yudha Satya Perkasa³**

¹ *Program Studi Fisika, Universitas Islam Negeri Sunan Gunung Djati Bandung, Jl. A.H.
Nasution 105 Bandung 40614, Indonesia*

² *Laboratorium Fisika Nuklir, Universitas Islam Negeri Sunan Gunung Djati Bandung, Jl. A.H.
Nasution 105 Bandung 40614, Indonesia,*

³ *Laboratorium Sistem Modeling, Universitas Islam Negeri Sunan Gunung Djati Bandung, Jl.
A.H. Nasution 105 Bandung 40614, Indonesia,*

*E-mail: 1157030004@student.uinsgd.ac.id
hp: +62-85-741450513

ABSTRAK

Sangat penting untuk menganalisis perubahan komposisi teras reaktor pada proses menghidupkan dan mematikan reaktor (*startup and shutdown* reaktor). Pada skema sederhana reaksi fisi, isotop ^{135}Xe dapat dihasilkan secara langsung maupun dari produksi fisi isotop ^{135}I . Produk ^{135}I ^{135}Xe berperan dalam penampang lintang penyerapan neutron termal. Proses peracunan yang terjadi pada ^{135}Xe menjadi salah satu produk fisi yang paling signifikan dalam penyerapan neutron termal dan cenderung menghasilkan hasil fisi yang besar. Untuk menyimulasikan proses peracunan produk fisi digunakan bahasa pemrograman *JavaScript* yang hasilnya dapat dengan mudah diakses melalui *web browser*. Simulasi ini dapat mempermudah pengguna dalam mempelajari dan menganalisis apa saja yang terjadi ketika teras reaktor nuklir dihidupkan ataupun dimatikan, hanya dengan memasukan beberapa parameter fisik, program akan mengalkulasikan data secara otomatis dan menghasilkan output berupa grafik konsentrasi produk fisi ^{135}Xe dan ^{135}I terhadap perubahan waktu.

Kata Kunci : Simulasi; ^{135}Xe dan ^{135}I ; Peracunan produk fisi;

ABSTRACT

It's essential to analyze reactor core composition changes when startup and shutdown reactor process. In the simple scheme of fission reaction, isotope of ^{135}Xe can be produced directly or from isotope fission product ^{135}I . Fission product of ^{135}I and ^{135}Xe contribute in thermal absorption cross section. ^{135}Xe fission product poisoning that happen become a very significant fission product in thermal neutron absorption and tends to produce large fission. To simulate fission product poisoning, JavaScript programming language become a base constructor. The result easily accessed through a web browser. This simulation can facilitate the users in learn and analyze whatever happens when reactor core is turned on or turned off, just by entering some physical parameters, the program will calculate data automatically and produce output in the form of fission product concentrate graph of ^{135}Xe and ^{135}I against time.

Keywords : Simulation; ^{135}Xe and ^{135}I ; Fission product poisoning.

1. Pendahuluan

Jika berkaca pada kejadian kecelakaan reaktor nuklir *Chernobyl* pada tahun 1986 yang membuat zat radioaktif dalam jumlah besar terlepas ke lingkungan. Terdapat karakteristik isotop ^{135}Xe yang berperan penting dalam tahap terjadinya kecelakaan reaktor tersebut. ^{135}Xe yang diproduksi di dalam bahan bakar nuklir memiliki sifat menyerap neutron hasil produk fisi, peristiwa ini disebut peracunan ^{135}Xe . Peristiwa ini yang membuat reaktor tidak boleh dihidupkan kembali sampai periode waktu tertentu sebelum reaktor dianggap bersih dari ^{135}Xe . Hal ini berkaitan dengan reaktivitas daya reaktor yang jika daya reaktor naik dengan sangat cepat maka akan mengubah semua air pendingin pada reaktor menjadi uap. Ketika tekanan uap melebihi kapasitas akan mengakibatkan dinding reaktor rusak dan mengalami kebocoran. Maka dari itu menganalisis perubahan komposisi pada teras reaktor sangatlah penting terutama peristiwa peracunan ^{135}Xe .

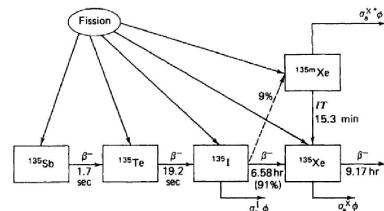
Untuk memambantu dalam menganalisis komposisi reaktor, dibuatlah sebuah program simulasi yang mudah digunakan dan diakses untuk menganalisis

peracunan produk fisi menggunakan bahasa pemrograman *JavaScript*. Jika dibandingkan dengan program sejenis maka dapat dikatakan simulasi yang dibuat dalam bahasa pemrograman *JavaScript* lebih mudah diakses karena cukup menggunakan sebuah *web browser*.

2. Bahan dan Metode

2.1. Dasar Teori

Jika diperhatikan dalam skema sederhana reaksi fisi, ^{135}Xe dapat dihasilkan dari reaksi fisi secara langsung dari ^{235}U maupun dari hasil peluruhan ^{135}Te yang menghasilkan ^{135}I menjadi ^{135}Xe dengan waktu paruh sebesar 9,2 jam. Karena waktu paruh dari ^{135}Sb sampai ke ^{135}I sangat singkat, maka skema peluruhan tersebut dapat disederhanakan menjadi ^{135}I akan meluruh langsung menjadi ^{135}Xe .



Gambar 1. Skema pembentukan ^{135}Xe produk fisi

Untuk menentukan perubahan konsentrasi ^{135}I dan ^{135}Xe dengan

menganggap γ_I dan γ_X sebagai fraksi fisi efektif dari kedua isotop tersebut serta λ_I dan λ_X sebagai konstanta peluruhan maka persamaan pembentukan ^{135}I dan ^{135}Xe dapat ditulis sebagai berikut:

$$I(t) = \gamma_I \sum_f \phi - \lambda_I I \quad (1)$$

$$X(t) = \gamma_I I + \gamma_X e \sum_f \phi - \sigma_a^X \phi X - \lambda_X X \quad (1)$$

$$I(t) = \frac{\gamma_I \sum_f \phi_0}{\lambda_I} (1 - \exp(-\lambda_I t)) \quad (3)$$

$$X(t) = \frac{(\gamma_I + \gamma_X) \sum_f \phi_0}{\lambda_X + \sigma_a^X \phi_0} [1 - \exp((-\lambda_X + \sigma_a^X \phi_0)t)] + \frac{\gamma_I \sum_f \phi_0}{\lambda_X - \lambda_I + \sigma_a^X \phi_0} [\exp(-(\lambda_X + \sigma_a^X \phi_0)t) - \exp(-\lambda_I t)] \quad (4)$$

Pada fluks yang konstan, ^{135}I dan ^{135}Xe dihitung menggunakan persamaan

$$I(t) \xrightarrow{t \rightarrow \infty} I_\infty = \frac{\gamma_I \sum_f \phi_0}{\lambda_I} \quad (5)$$

$$X(t) \xrightarrow{t \rightarrow \infty} X_\infty = \frac{(\gamma_I + \gamma_X) \sum_f \phi_0}{\lambda_X + \sigma_a^X \phi_0} \quad (6)$$

$$I(r, t) = I_\infty(r) \exp(-\lambda_I t) \quad (7)$$

Persamaan (7) dapat disubstitusikan untuk menentukan konsentrasi ^{135}Xe sebagai fungsi ruang dan waktu

$$X(r, t) = X_\infty(r) \exp(-\lambda_X t) + \frac{\lambda_I I_\infty(r)}{\lambda_I - \lambda_X} [\exp(-\lambda_X t) - \exp(-\lambda_I t)] \quad (8)$$

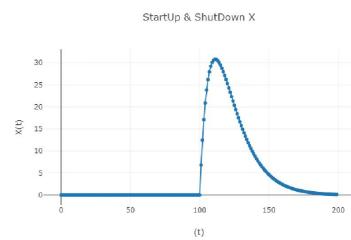
Reaktivitas negatif dari ^{135}Xe dapat dicari dengan persamaan

$$\Delta\rho(t) = -\frac{1}{vpe} \left[\frac{(\gamma_I + \gamma_X) \phi_0}{\lambda_X / \sigma_a^X + \phi_0} \exp(-\lambda_X t) + \frac{\lambda_I \sigma_a^X \phi_0}{\lambda_I - \lambda_X} [\exp(-\lambda_X t) - \exp(-\lambda_I t)] \right] \quad (9)$$

Bahasa pemrograman *JavaScript* digunakan untuk mensimulasikan perubahan konsentrasi ^{135}I dan peracunan ^{135}Xe yang bergantung pada waktu. Adapun Metode Integrasi numerik diaplikasikan dalam program yang dibuat. Nilai fluks dan waktu digunakan sebagai data masukan yang bisa diatur oleh pengguna.

3. Hasil dan Pembahasan

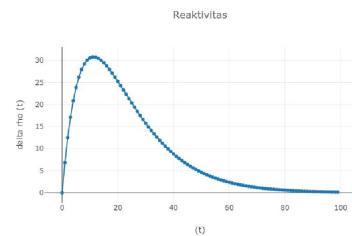
Dari hasil pengukuran didapat data konsentrasi ^{135}Xe yang berubah bergantung waktu.



Gambar 1. Konsentrasi ^{135}Xe kondisi Startup & Shutdown

Nilai *cross section* absorpsi ^{135}Xe yang sangat besar membuat ^{135}Xe banyak menyerap neutron. Hal ini ditunjukkan pada grafik (1) yang ketika reaktor dimatikan secara mendadak maka jumlah ^{135}Xe meningkat dengan sangat signifikan dan inilah yang memicu terjadinya reaktivitas negatif dimana nilainya berbanding lurus dengan konsentrasi ^{135}Xe yang ditunjukkan pada grafik (2). Jika diperhatikan lebih lanjut, setelah konsentrasi ^{135}Xe mencapai titik tertinggi maka konsentrasi ^{135}Xe mulai turun secara perlahan. Oleh karena itu ketika reaktor dimatikan, maka reaktor tidak boleh dinyalakannya kembali dalam periode waktu tertentu sampai reaktor sudah bersih dari ^{135}Xe [2]. Jika reaktor dipaksa untuk dinyalakan kembali sebelum bersih dari ^{135}Xe , maka akan muncul lonjakan daya sesaat yang menyebabkan kenaikan temperatur pada pendingin air pada reaktor. Ketika terjadi kenaikan temperatur tersebut membuat air yang berfungsi sebagai pendingin reaktor berubah menjadi uap, maka bertambah pula tekanan di dalam reaktor. Jika reaktor tidak mampu menahan tekanan tersebut maka akan

terjadi ledakan seperti halnya yang terjadi pada reaktor *Chernobyl* di Rusia.



Gambar 2 Reaktivitas negatif ^{135}Xe

4. Simpulan

Dengan membuat sebuah simulasi peracunan produk fisi ^{135}Xe yang dapat diakses melalui *web browser*, akan memudahkan proses pembelajaran dan analisis perubahan konsentrasi ^{135}Xe yang ternyata sangat berpengaruh dalam proses menyalakan dan mematikan reaktor dan menjadi salah satu faktor penting pada kejadian kecelakaan reaktor *Chernobyl*.

5. Referensi

1. Duderstadt, James J. (1976). *Nuclear Reactor Analysis*. New York: John Wiley & Sons.
2. Septilarso, Anggoro. (2012). Chernobyl 25 Tahun Lalu. *Prosiding Pertemuan Ilmiah XXV HFI Jateng*

- & DIY. HFI: Purwokerto.
3. Stacey, Wetson. (2017). *Nuclear Reaktor Physic. Second Edition.* WILEY-VCH Verlag GmbH & Co. KGaA, Weinheim
 4. Suparlina, Lily. (1991). Penentuan Reaktivitas ^{135}Xe untuk Pengoperasian Reaktor Pada Teras V RSG Gas. *Seminar Reaktor Nuklir dalam Penelitian Sains dan Teknologi Menuju Era Tunggal Landas.* PPTN-BATAN: Bandung.

B.3 Static and Dynamic Neighbor Interaction of Floccing Model in Newtonian Approach

Static and Dynamic Neighbor Interaction of Flocking Model in Newtonian Approach

Ariq Dhia Irfanudin^{1, a)} Dinda Ravi Algifari^{1, b)} Sparisoma Viridi^{2,c)} Yudha Satya Perkasa^{1, d)}

¹*Nuclear Physics and Computation Research Division, Departement of Physics, Universitas Islam Negeri Sunan Gunung Djati Bandung*

²*Nuclear Physics and Biophysics Research Division, Departement of Physics, Institut Teknologi Bandung*

a)1157030004@student.uinsgd.ac.id
b)1157030007@student.uinsgd.ac.id
c)dudung@fi.itb.ac.id
d)yudha@uinsgd.ac.id

Abstract. There is some collective motion that does by animal groups, such as bird flocks, fish school, insect swarms. In nature, Structural order showed movement and direction every single individual make them safer and more efficient. The desire to flock is dependent on the nearest-neighbor distance even though the other parameters that can affect flocking. To get more insight into this system, we manipulate interaction with each particle in two situations. In a static condition, we set each particle to interact with their group only, which means the particle will not respond to other entities that not selected. In a dynamic situation, we set each particle can interact with a predator as long as the neighbor is in the orientation and attraction zones of the particle. The Newtonian approach used to handle physical parameters such as position, velocity, and, acceleration of the particles, then these physical parameters are calculated and updated by the Euler integration. Some patterns appear in our simulation, such as line formation and dynamic parallel group.

Keywords: Collective motion, Runge-Kutta, static neighbor, dynamic neighbor.

INTRODUCTION

In nature, to be safer is one of some reason why bird, insect, and, fish do flock. When a group of animal move in flocks, it had been attracted scholars' attention to know how this phenomenon happens. In 1986, Craig Reynolds proposed a computer model to visualize flocking system¹. In his simulation, Boids is a term made by him to describe creatures that do flocking. As long as we know about collective behavior, flocking behavior is a complex system which has three fundamental principles of a complex system. There are simple units with the short-range relationship; simple units operate in parallel, and order as a whole exhibit emergent phenomenon. The basic idea of how flocking visualized in a computer builds a vehicle that obeys three simple collective behavior rules. The first one is alignment, an ability to move towards the average direction of local neighbors. Cohesion makes particles look for the group center of mass. Separation moves out to avoid collision to local neighbors. Particles do not interact with all the neighbors, but only to neighbors within their perception zone. This characteristic is measured based on the distance of particles relative to its neighbors.

There are several models to describe collective behavior^{2,3,4} proposed a model of neighbors interaction that considered distance-dependent force and velocity-dependent force. Particles can interact with their neighbors that located at the edge (front/near) of flocking, and only pairwise interactions considered. In a different approach, Vicsek

serve a simple model to establish a quantitative interpretation of collective behavior, which currently known “Vicsek Model”⁵. This model focused on the average directional orientation of particle with some stochastic and deterministic factors like perturbations. In self-propelled particles (SPP) system, each particle moves with fixed velocity then assume average direction of other particles within a perception radius.

Furthermore, SPP has three basic rules of flocking, where the particle obeys alignment, cohesion, and separation as Reynold's model did. Nevertheless, it does not make any sense if a creature (bird, fish, and insect) capable of looking at all direction, so to get more realistic flocking simulation, perception radius must be detailed. Here Couzin has a different way to add a blind zone, an unseeable area, and view zone, a seeable area⁶.

The primary goal of this research is to get more understanding of what will happen if we create two kinds of neighbor interaction. We set particle can interact to their group only. On the other situation, the particle can interact with a predator. Indeed, we create a particle that obeys the basic rules of flocking then give them limitation to looking for their nearest neighbors. We hope that improvement will make our work useful to anyone.

SIMULATION AND METHODOLOGY

Motion Model

Our model built on Newton's second law that integrates each motion parameters to get a new position. We serve a general formula to define any force that will happen in the simulation. Nevertheless, the value of n on equation (1) will not bigger than 2.

$$\vec{F}_i = \sum_{i \neq j} \left(\sum_{n=1}^{\infty} \alpha_n \frac{\vec{p}_{ij}}{\left| \vec{p}_{ji} \right|^n} + \sum_{n=1}^{\infty} \beta_n \frac{\dot{\vec{p}}_{ji}}{\left| \dot{\vec{p}}_{ji} \right|^n} \right), \quad (1)$$

$$\vec{a}_i = \frac{1}{m_i} \sum \vec{F}_i, \quad (2)$$

$$\vec{v}_i^{k+1} = \vec{v}_i^k + \vec{a}_i \Delta t, \quad (3)$$

$$\vec{p}_i^{k+1} = \vec{p}_i^k + \vec{v}_i^{k+1} \Delta t. \quad (4)$$

We consider the mass of each vehicle is scaled to be $m = 1$. The number of particles represented by N with $i = 1, 2, 3, \dots, N$, with following by simple rules of motion. The particle placed in position \vec{p}_i , move in \vec{v}_i , 2D space and updated by time t with $\Delta t = 1$ as an increment time. An initial condition, such as position and velocity set randomly. There is a square with length L as the area with periodic boundary condition.

There are input state $x = \langle \lambda_1, \dots, \lambda_n \rangle$ also, output state $\lambda(q) = \langle p, v \rangle$. These states used in processing particle properties. Each particle has properties such as position, velocity, perception zone, mass, and peripheral view. These properties called current state properties (q).

$$q = \langle p, v, z, m, \theta \rangle. \quad (5)$$

Basic Rules of Flocking

We consider four basic rules of flocking. There are alignment, cohesion, separation, and escaping. Each particle has a short-range relationship that makes the vehicle being able to sense its closest neighbor. This relationship triggers the particle to flock. Those rules represent forces that happen while the particle moves.

The alignment rule makes particle want to keep in flocking and move in the same speed and direction.

$$D_i^a(p, q) = \alpha^a \frac{1}{N^a} \sum_{i \neq j}^{N^a} \frac{\vec{v}_{ji}}{|\vec{v}_{ji}|}, \alpha^a = 1N, \quad (6)$$

where $\vec{v}_{ji} = \vec{v}_j - \vec{v}_i$.

The cohesion rule makes the particle always look for the center position of the group.

$$D_i^c(p, q) = \alpha^c \frac{1}{N^c} \sum_{i \neq j}^{N^c} \frac{\vec{p}_{ji}}{|\vec{p}_{ji}|}, \alpha^c = 1N, \quad (7)$$

where $\vec{p}_{ji} = \vec{p}_j - \vec{p}_i$.

As a precaution to make the particle do not run into one another, if there are neighbors too close, the particle must go to the opposite direction. How far the particle has to avoid the others, it depends on how close they are. Therefore, separation rule is a rule that every particle can never wholly ignore.

$$D_i^s(p, q) = \alpha^s \sum_{i \neq j}^{N^s} \frac{\vec{p}_{ji}}{|\vec{p}_{ji}|^2}, \alpha^s = 1Nm. \quad (8)$$

We add a rule that particle has ability escaping from the predator. This scenario will provide us to know the dynamic neighbor interaction when a predator disturbs particles' flocking. It will look like

$$D_i^e(p, q) = \alpha^e \sum_{i \neq j}^{N^e} \frac{\vec{p}_{ji}}{|\vec{p}_{ji}|}, \alpha^e = -1N. \quad (9)$$

Particle Model

Those rules only considered by a vehicle if there are neighbors within the vehicle's perception radius. The perception radius depends on the distance between the two-vector position. There are two kinds of perception radius. The blue circle outside is a perception radius of a vehicle to handle alignment and cohesion rules and the red circle inside to handle separation and escaping rules. Honestly, it is not realistic for a particle that aware of every other particle in any direction. Thus, we set a peripheral view (purple area) with a fixed viewing angle as their eye, see FIGURE 1. This feature is similar to what⁷ done as an effort to build a flocking V formation.

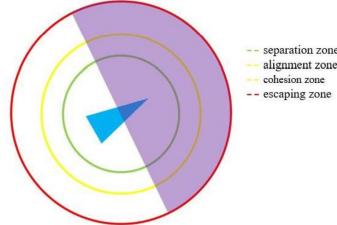


FIGURE 1. Representation of a particle with a peripheral view (θ), separation zone, (z_s), alignment zone (z_a) and, cohesion zone (z_c), and escaping zone (z_e).

During the particle moving and trying to flock, there are several technical things have to do. If we want to know whether the neighbor is inside the peripheral view, the angle between the two particles must be measured.

$$\varphi(\lambda_i, q_j) = \arccos \left(\frac{\mathbf{v}_j \cdot (\mathbf{p}_i - \mathbf{p}_j)}{\|\mathbf{v}_j\| \|\mathbf{p}_i - \mathbf{p}_j\|} \right), \quad (10)$$

$$d(\lambda_i, q_j) = \|p_i - p_j\|. \quad (11)$$

The distance between the two particles used to give us information is there any neighbor within the zones.

$$P_i^a(x, q) = \{d(\lambda_i, q_j) < z_a \wedge \varphi(\lambda_i, q_j) < \vartheta\}. \quad (12)$$

$$P_i^c(x, q) = \{d(\lambda_i, q_j) < z_c \wedge \varphi(\lambda_i, q_j) < \vartheta\}. \quad (13)$$

$$P_i^s(x, q) = \{d(\lambda_i, q_j) < z_s \wedge \varphi(\lambda_i, q_j) < \vartheta\}. \quad (14)$$

$$P_i^e(x, q) = \{d(\lambda_i, q_j) < z_e \wedge \varphi(\lambda_i, q_j) < \vartheta\}. \quad (15)$$

The function of perception zone will handle A and B then give us boolean value. If the neighbor is in range, the particle receives specific forces such as alignment, cohesion, and separation.

$$D_i(\{p_1, \dots, p_k\}, q). \quad (16)$$

$$p_i = P_i(x, v). \quad (17)$$

While the interaction among particles occurs, resultant force will update their acceleration.

Where the predator does not exist, in the first condition, particles will be set to prioritize the alignment rule. However, for the second condition, the predator will try to catch the particle. Consequently, the weighting factor (ω) is needed adjusting which rule that has to prioritize.

$$\sum \vec{F}_i = \vec{D}_i^a + \vec{D}_i^c + \vec{D}_i^s + \vec{D}_i^e, \quad (18)$$

$$\sum \vec{F}_i = \omega^a \vec{D}_i^a + \omega^c \vec{D}_i^c + \omega^s \vec{D}_i^s + \omega^e \vec{D}_i^e. \quad (19)$$

RESULT

We set several scenarios to know the difference interaction in the particles' group. There are two main kinds of situation. In the first scenario, we observe the particle group movement for five minutes ($t = 300$) without predator. Otherwise, we add a predator in the second scenario. Predator is an entity that figure played to catch the particles. As long as we know, the perception zone represents a particle's view radius. The perception zone also adjusted in each simulation.

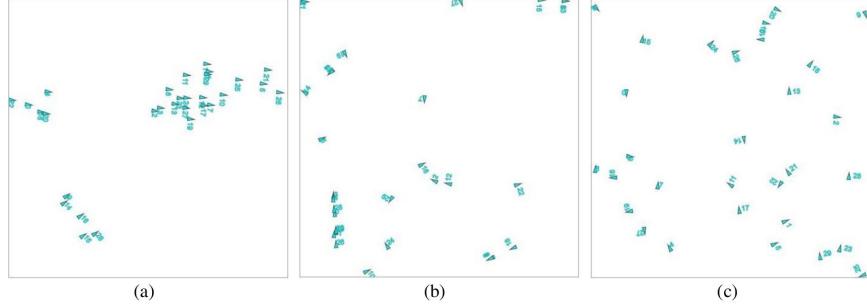


FIGURE 2. Collective behavior of 30 particles, $t = 300$ with the same value $z_a = 0.8$, $z_c = 0.8$, and, $z_s = 0.5$, for (a) Alignment, (b) cohesion, (c) separation.

In the beginning, we set the particle only has one rule of flocking. The configuration of perception zones causes a particle's decision tends to give more attention to what will happen next, especially for the separation and escaping rules. In FIGURE 2(a), the whole particles can flock in their group that caused by the wide z_a and z_c . Another pattern shown by FIGURE 2(b), it is difficult for the particles to find out the center of mass since only cohesion rule that applied. In FIGURE 2(c), there is no group formed that caused by the separation rule. This kind of rule makes the particle does not want to flock.

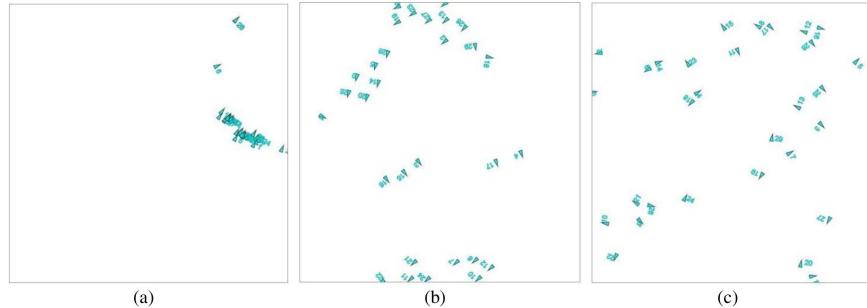


FIGURE 3. Collective behavior of 30 particles $t = 300$ with $z_a = 0.8$, $z_c = 0.8$, and, $z_s = 0.5$, for (a) Alignment and cohesion, (b) alignment and separation, (c) cohesion and separation.

In this scenario, there is a combination of two rules applied. Distinguishing thing in FIGURE 3(a) and FIGURE 3(a) is the particles able to flock with/without collision. Meanwhile, the particle in FIGURE 3(b) fail to flock.

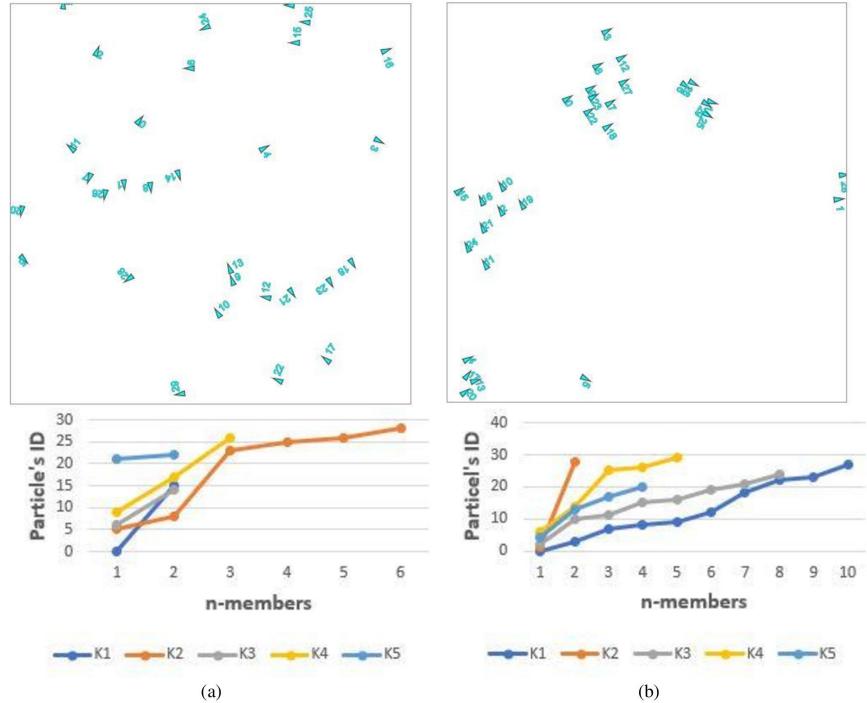


FIGURE 4. Collective behavior of 30 particles with for (a) $z_a = 0.5, z_c = 0.5, z_s = 0.5$. (b) $z_a = 0.8, z_c = 0.8, z_s = 0.5$.

As we can see, FIGURE 4(a) and FIGURE 4(b) the difference in this scenario is the combination of three rules of flocking. As a result of FIGURE 4(b) which has a bigger z_a and z_c than FIGURE 4(a), the formed group has more members.

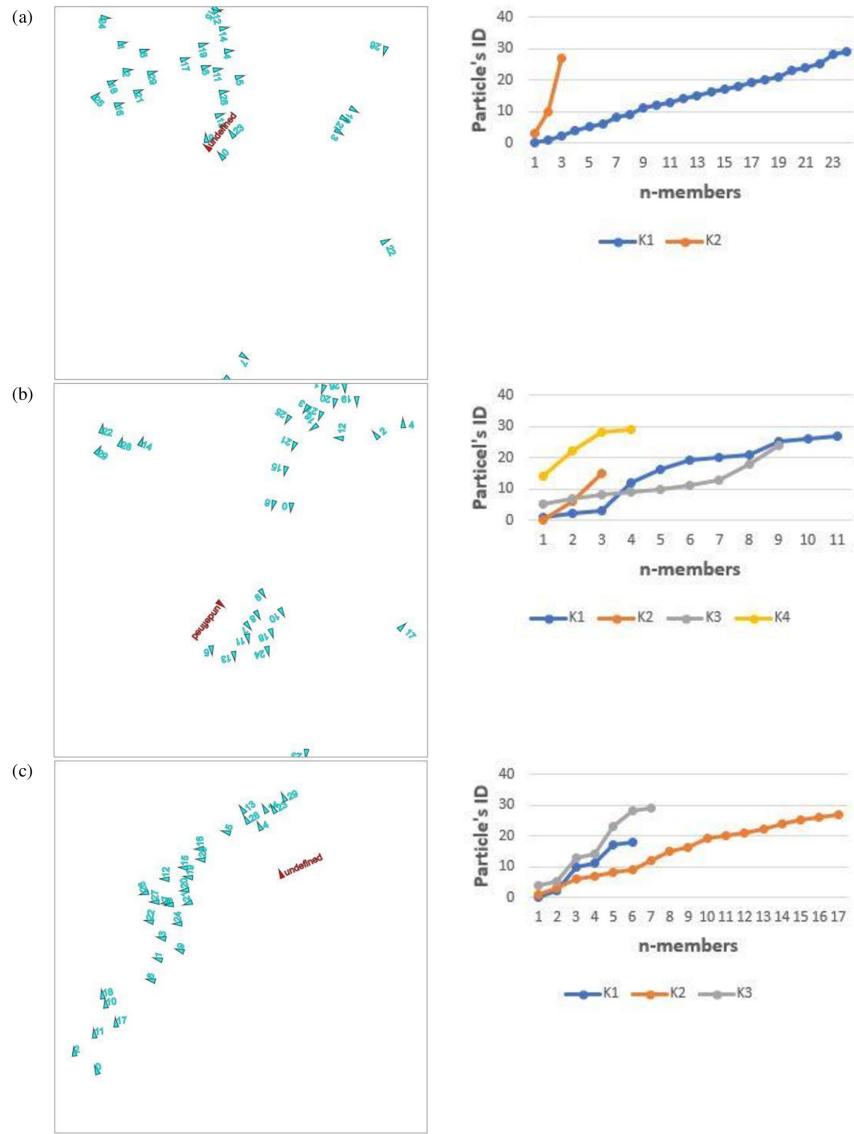


FIGURE 5. Collective behavior of 30 particles with for (a) $z_a = 1$, $z_c = 1$, $z_s = 0.5$, $z_e = 0.2$. (b) $z_a = 0.8$, $z_c = 0.8$, $z_s = 0.5$, $z_e = 0.8$. (c) $z_a = 1$, $z_c = 1$, $z_s = 0.5$, $z_e = 1.4$.

In the last scenario, there is a predator that has a desire to catch the particles. When the predator has been in the group of the particle escaping zone, the closest particle to the predator will move away as soon as possible. When the predator comes from the opposite direction, the group just split into two groups while the predator will only target the closest one. The smallest escaping zone (z_e) FIGURE 5(a) make the particles flock and stay away by the predator at the same time so hard. Meanwhile, in FIGURE 5(c), the particles could make it.

DISCUSSION

Many of the original works on flocking model that more detailed in perception zones. Such as who was exploited the mathematical structure of this feature⁸. Although Our model considers the simple schema of perception zones, we got some structure of collective behaviors. The particle was more cohesive than before when the z_a dan z_c increased. The size of the particle's perception zone is essential not only to align with one another and to minimize collision but also in permitting the particle to transfer information.

In this paper, the shaped formation formed by particles is parallel and line structure. There are other formations like torus formation⁶⁹. When we adjusted the weighting factor of alignment and cohesion, we got a torus formation. Nevertheless, we show a set of weight that can form parallel and line formation.

Present a predator causes significant changes in the neighbor interaction. The particles are kept trying to stick in the group. The predator is harder to catch the particles while the particles are capable of transferring the information to each other, thus escape easily. Set the size of escaping zone be the biggest one is a better idea to make a particle more careful when deciding, especially for escaping from the predator. On the other hand, if we set the z_s bigger than the z_a and z_c , the particles totally cannot do flock.

Meanwhile, if we take an example from nature when a herd of zebra is crossing a river, the predator will target the weakest, farthest and, the nearest one. The predator quite knowledgeable for does not get into the group. Meanwhile, our predator has the ability to targeting the nearest vehicle on it. Thus, the predator impetuously gets into the vehicle group. It is better to add that ability in further research.

CONCLUSION

Flocking simulation in the Newtonian approach has been built. Through the combination of the basic rule of flocking, the alignment rule become the most crucial rule if we want the particles to flock. The result shows that configuring the size of the perception zones could make a powerful impact. Notably, when we set the escaping zone to be $z_e = 1.4$, the particles easily to flock, avoid the predator, and make a dynamic parallel group. It is worth to build the flocking simulation with Quad-Tree method that can optimize the simulation¹⁰. For further research, It had better implement the algorithm to drones as Vásárhelyi has done¹¹.

REFERENCES

1. Reynolds, C. W. Flocks, Herds, and Schools: A Distributed Behavioral Model, in Computer Graphics. *ACM SIGGRAPH Comput. Graph.* **21**, 25–34 (1987).
2. Li, Y., Lukeman, R. & Edelstein-keshet, L. Minimal mechanisms for school formation in self-propelled particles. **237**, 699–720 (2008).
3. Czirók, A., Vicsek, M. & Vicsek, T. Collective motion of organisms in three dimensions. *Phys. A Stat. Mech. its Appl.* **264**, 299–304 (1999).
4. Niwa, H. S. Newtonian dynamical approach to fish schooling. *J. Theor. Biol.* **181**, 47–63 (1996).
5. Vicsek, T. & Zafeiris, A. Collective motion. *Phys. Rep.* **517**, 71–140 (2012).
6. Couzin, I. D., Krause, J., James, R., Ruxton, G. D. & Franks, N. R. Collective Memory and Spatial Sorting in Animal Groups. 1–11 (2002). doi:10.1006/jtbi.3065
7. Frame, M. & Flake, G. W. The Computational Beauty of Nature: Computer Explorations of Fractals, Chaos,

- Complex Systems, and Adaptation. *Am. Math. Mon.* **107**, 576 (2000).
- 8. Mogilner, A., Edelstein-Keshet, L., Bent, L. & Spiros, A. *Mutual interactions, potentials, and individual distance in a social aggregation*. *Journal of Mathematical Biology* **47**, (Springer, 2003).
 - 9. Iliass, T., Cambui, D., Grosso, M. & Grosso, M. The combined effect of attraction and orientation zones in 2D flocking models. **30**, 1–11 (2016).
 - 10. Raynaud, S. 3D Boids with QuadTree. (2018).
 - 11. Vásárhelyi, G., Virág, C., Somorjai, G. & Nepusz, T. Optimized flocking of autonomous drones in confined environments. **3536**, 1–14 (2018).

Lampiran C

Riwayat Hidup



Ariq Dhia Irfanudin, seorang pria keturunan jawa asal Purbalingga yang tidak bisa bicara bahasa jawa. Pendidikan dasar hingga masa perkuliahan ia tempuh di Kota Bandung. Latar belakang pendidikan dalam bidang ilmu fisika mengantarkannya dengan berbagai hobi. Bermain dengan bahasa pemrograman dan data demi menghasilkan berbagai aplikasi sederhana nan kreatif. Bermain dengan rangkaian kata dalam upaya menyusun sebuah naskah cerita. Hal tersebut yang mengantarkan ia menjadi *programmer* dan *scriptwriter* amatir. Ia lebih mudah ditemukan di sebuah tempat bernama Masjid Salman ITB ketimbang di kampusnya sendiri. Mengumpulkan kawan untuk bersua, mencari kawan yang sevisi demi mewujudkan *The Next Baitul Hikmah*.