MEKANISME FLOCKING PARTIKEL PADA DITERAPKAN PADA PERGERAKAN TAWAF DENGAN MENGGUNAKAN METODE RUNGE KUTTA

SKRIPSI

Diajukan Sebagai Salah Satu Syarat Untuk Memperoleh Gelar Sarjana
Jurusan Fisika Fakultas Sains dan Teknologi
Universitas Islam Negeri Sunan Gunung Djati Bandung

IKHSAN MOCHAMMAD NOOR

1157030026

FISIKA
FAKULTAS SAINS DAN TEKNOLOGI
UNIVERSITAS ISLAM NEGERI SUNAN GUNUNG DJATI BANDUNG
2022

	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 : IKHSAN MOCHAMMAD NOOR
NIM : 1157030026
Jurusan : FISIKA

Dengan ini menyatakan sebagai berikut :

1. Skripsi yang berjudul MEKANISME FLOCKING PARTIKEL PADA DITERAPKAN PADA PERGERAKAN TAWAF DENGAN MENGGUNAKAN METODE RUNGE KUTTA 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 2022
Yang Membuat Pernyataan

IKHSAN MOCHAMMAD NOOR
1157030026

LEMBAR PERSETUJUAN

MEKANISME FLOCKING PARTIKEL PADA DITERAPKAN PADA PERGERAKAN TAWAF DENGAN MENGGUNAKAN METODE RUNGE KUTTA

IKHSAN MOCHAMMAD NOOR

1157030026

Menyetujui,

Pembimbing 1,	Pembimbing 2,
Dr. Yudha Satya Perkasa, M.Si	**Ridwan Ramdani, S.Si., M.Si**
NIP. **197911172011011005**	NIP.

Mengetahui,

Dekan Fakultas Sains dan Teknologi,	Ketua Jurusan Fisika,
Dr. Hasniah Aliah, M.Si	**Dr. Moh. Nurul Subkhi, M.Si**
NIP. **197806132005012014**	NIP. **198102012009121003**

LEMBAR PENGESAHAN

Skripsi dengan judul : "**MEKANISME FLOCKING PARTIKEL PADA DITERAPKAN PADA PERGERAKAN TAWAF DENGAN MENGGUNAKAN METODE RUNGE KUTTA**" telah dipertanggung jawabkan dalam sidang Munaqasyah Jurusan Fisika Fakultas Sains dan Teknologi UIN Sunan Gunung Djati Bandung pada 26 Agustus 2022, dengan majelis yang terdiri dari:

Menyetujui ,

Penguji I	Penguji II,
Mada Sanjaya W.S., Ph.D	**Dr. rer.nat. Imamal Muttaqien, M.Si**
NIP. **198510112009121005**	NIP. **198310062009121009**

Mengetahui ,

Ketua Majelis,	Sekretaris,
Dr. Yudha Satya Perkasa, M.Si	**Ridwan Ramdani, S.Si., M.Si**
NIP. **197911172011011005**	NIP. **198904162019031016**

LEMBAR PERSEMBAHAN

"Allah akan meninggikan orang-orang yang beriman diantaramu dan orang-orang yang diberi ilmu pengetahuan beberapa derajat" (Firman Allah SWT: Al-Mujadalah,11)

"Sesungguhnya, Kami menciptakan segala sesuatu menurut ukuran."(Firman Allah SWT: Al Qamar,29)

"Menuntut ilmu itu diwajibkan bagi setiap muslim dan muslimah."(Sabda Nabi Muhammad SWA)

Ku persembahkan Skripsi dalam bentuk LaTeX ini untuk:

Wawa Wahyudin
Nursilawati
Terry Febriani
Aldy Rialdy Atmadja
Amelia Hasanah
dek Aura
dek Zara

ABSTRAK

Nama	:	IKHSAN MOCHAMMAD NOOR
Program Studi	:	Fisika
Judul	:	MEKANISME FLOCKING PARTIKEL PADA DITERAPKAN PADA PERGERAKAN TAWAF DENGAN MENGGUNAKAN METODE RUNGE KUTTA

Telah dilakukan penelitian tentang gerak berkelompok yang terdapat pada hewan seperti burung, ikan bahkan manusia. Pergerakan dapat diterapkan pada tawaf yang merupakan pola yang yang dipengaruhi oleh keinginan untuk berkelompok agar effisien dan mendekat pada jenis yang sama. keiginan tersebut bergantung pada parameter-parameter tertentu sebagai sistem. Untuk memahami sistem ini dapat dilakukan pendekatan partikel. Interaksi antar partikel dapat digolongkan sebagai keinginan partikel untuk berkelompok sejenis. dan interaksi kedua dipengaruhi oleh gaya dominan yang memmbuat keinginan tawaf tersebut terpola. Pendekatan Newtonian digunakan untuk menangani parameter fisika seperti posisi dan kecepatan partikel yang Kemudian dihitung menggunakan integrase runge-kutta. Pola yang muncul diantaranya adalah pola gerak menuju pusat gaya(kaaba).

Kata Kunci: gerak berkelompok, runge-kutta, gaya dominan

ABSTRACT

Name	:	IKHSAN MOCHAMMAD NOOR
Studies Program	:	Physics
Title	:	*judul Inggris*

Research has been done on group movement found in animals such as birds, fish and even humans. Movement can be applied to tawaf which is a pattern that is influenced by the desire to group together to be efficient and closer to the same type. the desire depends on certain parameters as a system. To understand this system, a particle approach can be used. The interaction between particles can be classified as the desire of particles to group together. and the second interaction is influenced by the dominant force that makes the tawaf desire patterned. The Newtonian approach is used to handles physical parameters such as position and velocity of the particles which are then calculated using the runge-kutta integration. The patterns that emerge include the pattern of motion towards the center of force (kaaba).

Keywords: group movement, runge-kutta, dominant force

KATA PENGANTAR

Segala puji bagi Allah SWT, Skripsi yang berjudul MEKANISME FLOCKING PARTIKEL PADA DITERAPKAN PADA PERGERAKAN TAWAF DENGAN MENGGUNAKAN METODE RUNGE KUTTA 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 SWA yang mana beliu adalah panutan bagi penulis.
3. Bapak Dr.Yudha Setya Perkasa.M.Si Sebagai Dosen Pembimbing Akademik Jurusan Fisika, karena atas Bimbingan dan kepercayaan yang beliau berikan kepada penulis untuk menyelesaikan Akademik di Jurusan Fisika ini.
4. Bapak Ridwan Ramdani, S.Si., M.Si Selaku Dosen Pembimbing Jurusan Fisika, karena atas Bimbingan dan kepercayaan yang beliau berikan kepada penulis untuk menyelesaikan skripsi ini.
5. 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.
6. Rekan-rekan Fisika angkatan 2015 yang memotivasi agar penulis lulus sebagai sarjana.

7. Mahasiswa fisika angkatan 2008, 2009, 2010, 2011, 2012, 2013, 2014 ,2015 ,2016 ,2017 yang selalu memberikan dukungan dan motivasi agar penulis lulus sebagai sarjana.

8. Rekan seperjuangan laboratorium sistem modeling Dinda, Ariq, Arum dan Indri.

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 khususnya yang mendalami model komputasi.

Bandung, 26 Agustus 2022

Penulis

DAFTAR ISI

DAFTAR GAMBAR

DAFTAR TABEL

DAFTAR SINGKATAN

FSM : Finite State Machine
HTML : Hyper Text Markup Language
CSS : Cascading Style Sheet
RAM : Random Access Memory

DAFTAR SIMBOL DAN OPERATOR

α	:	Alpha
υ	:	Kecepatan
p	:	Posisi
m	:	Massa
t	:	Waktu
Δt	:	Step Size Waktu
$\vec{F}$	:	Function Flocking
σ	:	Standard deviasi
Σ	:	Penjumlahan variabel
π	:	nilai pi
e	:	epsilon
ψ	:	derajat putaran
λ	:	input state
κ	:	output state
z	:	radius antar partikel

BAB 1

PENDAHULUAN

1.1 Latar Belakang

Tawaf merupakan rukun ibadah dalam melaksanakan Haji, dimana seluruh umat islam dianjurkan melaksakannya. Ditinjau dari praktiknya, sederhananya tawaf ialah melakukan pergerakan mengelilingi kaaba sebanyak 7 kali dimulai dari garis hajar aswad dalam satu putaran, ditambah dengan pergerakan menuju hajar aswad dengan sunnah ingin menciumnya.Pergerakan menuju ke hajar aswad menjadikan masalah karena skala yang . Namun sebenarnya dalam keadaan haji, praktik ini menjadi sangat masif karena jumlah variabel individu yang sangat banyak. Penulis mendapatkan cerita, saat melaksanakan praktik ibadah haji, sepasang suami istri sedang melaksanakan salah satu rukun ibadah tawaf mereka sangat ingin mencium hajar aswad, tetapi sangat sulit untuk mencapai inti dari kaaba tersebut, maka mereka bergerak secara berkelompok untuk mencapainya dan berhasil. Fenomena tersebut sangat menarik karena jika sepasang suami istri tersebut tidak melakukan gerakan berkelompok maka tujuan pusat kaaba tidak tercapai. Sedangkan tawaf merupakan salah satu praktik ibadah yang dilakukan pada saat memasuki Masjidil Al-Haram di Mekkah Saudi Arabi. Tawaf sendiri dilaksanakan pada Umrah dan Haji. Saat Haji, keaadaan tawaf ada di tingkat yang paling tinggi dimana seluruh umat islam berkumpul untuk melaksanakan salah satu pilar islam yang diwajibkan bagi seluruh umat yang mampu. Pada tahun 2017 tercatat sekitar 2.352.122 yang datang dari seluruh dunia untuk melaksanakan haji(for Statistics Kingdom of Saudi Arabia, 2018), maka asumsinya seluruh orang tersebut melakukan tawaf bersamaan di waktu yang sama yaitu saat memasuki Masjidil Al-Haram. Jika dibandingkan dengan umrah frekuensi orang yang melakukan tawaf lebih rendah tingkat kepadatan orang.

Dan ini menjadikan tawaf, sebagai kegiatan dengan melibatkan jumlah orang terpadat di dunia. Maka merupakan hal yang sulit untuk mencapai pusat kaaba dengan paparan diatas. Menjadikan fenomena ini hal yang menarik untuk dibahas.

Fenomena ini dapat diterjemahkan dalam model yang terdiri dari model visual kerumunan dan model teknik numerik yang digunakan. Pertama adalah model visual, karena skalanya yang cenderung besar dengan melibatkan orang banyak. Maka penting untuk mengerti bentuk dari perilaku kerumunan saat melakukan tawaf. Variasi yang dilakukan kerumunan saat berada di Masjidil Al-Haram. Pemahaman terhadap model dari masalah pergerakan kerumunan ini mempunyai solusi berupa sebuah model yang dapat diterapkan, untuk itu diperlukan napak tilas kepada riset tentang perilaku kerumunan.

Riset sebelumnya para ahli dalam bidang kajian aliran kerumunan menggolongkan pada 9 varian major system diantaranya *flocking system*, *behavioral system*, *chaos model system*(Saiwaki *et al.*, 1997), *agent base model*(Khan & McLeod, 2012), *social forces model*(Zainuddin *et al.*, 2009), *hybrid model system*(Shuaibu, 2015), *fluid dynamic model*(Narain *et al.*, 2009),*cellular automata* (Lim & Zainuddin, 2012), *cognitive model*(Mulyana & Gunawan, 2010) and *pedestrian model system*(Haghighati & Hassan, 2013). Beberapa studi seperti kim dalam penelitiannya mengguunakan basis kecepatan dan FSM untuk membuat *behaviour system* untuk kerumunan sehingga objek yang berinteraksi dapat mengantisipasi tabrakan dan menghindarinya karena kekuatannya dapat diperhitungkan dari basis kecepatan tersebut(Kim *et al.*, 2014). Studi lainnya deperkenalkan oleh Shuaibu, dengan memodelkan tawaf membentuk jalur spiral menggunakan simulasi *dicrete-event* sistem antri(Shuaibu, 2015). Dapat dikatakan ini sebagai model algoritmanya. Dalam riset ini penulis cenderung menggunakan sistem basis agent untuk memodelkan.

Kedua adalah tentang model teknik numerik, model numerik ini adalah bagian yang digunakan untuk menghitung sistem dari pergerakan kerumunan ini. Dimana hal ini sebagai fondasi dari model untuk variabel dalam model visual. Basis riset untuk teknik numerik ini diantaranya *Finite state machine*(Curtis *et al.*, 2011)(Bicho *et al.*, 2012), *Reciprocal Collision Avoidance*, *velocity based model*(Kim *et al.*, 2014), *Ruled Based Model*, Metode Euler dan Langrangian(Narain *et al.*, 2009), *Discrete-event model*.Penulis cenderung menggunakan velocity based model dalam memodelkan sebagai basis fisika interaksinya.

Berikut referensi waktu tempuh seseorang untuk menyelesaikan tawaf:

No. Sample	Keterangan	Waktu Tempuh(menit)	Sumber Video
1	3/4 tawaf	2.14	matt zimbo banzai
2	full tawaf	3.07	mustafa aydemir
3	not full circle	3	-
4	full	4.2	haris munandar
5	full	4.32	patarani channel
6	not full	3/4 tawaf	sukmasari ahmad
7	full	8.03	best islamic videos
8	full tawaf	5.05	muhammad raziur rahmad
9	full tawaf	5.49	stunecity
10	full tawaf	3.06 umrah	Asep Hendra
11	full tawaf	4.27	Palmerah Televisi
12	full tawafkhana	2.2	Najamuddin Shahwani
13	full tawaf	5.26	All In One Pakistan
14	full tawaf	4.41	I M Zakria
15	full tawaf	3.44	Samee ur Rahman HSE Trainer
16	full tawaf	4.48/4.52	Akram Baryar Official
17	half tawaf	2.57	HSS pk
18	full tawaf	4.46	best islamic videos
19	3/4 tawaf	2.4	Ali Akhtar
20	full tawaf	3.48	best islamic videos
21	full tawaf	3.28 bagus	jag Nannn
22	3/4 tawaf	3.34	My Tech Zone
23	full tawaf	4.08	bagus akram baryar
24	full tawaf	-	ikha mabrur pratama
25	3/4 tawaf	4.24	Panduan Gampang Ibadah Umroh dan Haji Plus 2014
26	1/4 tawaf	1.15	Muhammad Kahif
27	1/4 tawaf	1.08	Abdul Waheed kakepoto
28	full tawaf	3.54	Aryo Pamungkas
29	3/4 tawaf	3	-
30	7 tawaf	1=3.01 dan 7=22.47	eljunaed
31	full tawaf	5.2	Haramain Travels
32	full tawaf	3.18	Saad Al Qureshi

Pada Riset ini berfokus pada simulasi kerumunan tawaf menggunakan metode flocking dalam pergerakan secara berkelompok. Dengan melakukan variasi *flocking* untuk membentuk pergerakan 1 ras hingga dapat terbentuk sebuah tawaf yang berkelompok sesuai ras pattern partikel. Dengan menggunakan metode runge-kutta. Metode Runge-Kutta ini dipilih karena mempertimbangkan kompleksitas dari metode perhitungan. Jika dibandingkan dengan metode Euler approksimasinya lebih tinggi, dan jika dibandingkan metode newton rapshon lebih rendah aporksimasinya akan tetapi akan menghasilkan kompleksitas perhitungan partikel yang tinggi, akibatnya memperlambat simulasi. Dalam riset kedepan dapat dikembangkan sebuah metode khusus dalam melaksanakan tawaf yang berbasis struktur dan sistematis. Untuk kedepannya dengan model yang sudah stabil dapat menggambarkan pergerakan tawaf dengan mencapai hajar aswad dengan cara berkelompok

1.2 Tujuan Penelitian

Tujuan dari penelitian ini adalah membuat sebuah model tawaf mekanisme minimal. Dan menganalisis gaya yang diterapkan agar simulasi stabil antara kelompok tawaf untuk mencapai hajar aswad.

1.3 Rumusan Masalah

Berdasarkan latar belakang yang telah diuraikan maka rumusan masalah dalam penelitian proposal tugas akhir ini adalah sebagai berikut:

1. Bagaimana proses simulasi tawaf bebentuk.
2. Bagaimana proses group terbentuk dalam tawaf
3. Bagaimana penyederhanaan ruang dan pembatasan individu
4. Ruang dalam melakukan tawaf disederhanakan pembatasan individu agar menjaga kestabilan simulasi
5. Bagaimana tawaf dapat berkelompok dapat terbentuk dengan tujuan mencapai hajar aswad
6. Bagaimana gaya antar kelompok dapat stabil dengan kelompok lain dapat mencapai hajar aswad
7. Bagaimana variable yang sudah ditambahkan mempengaruhi simulasi.

1.4 Batasan Masalah

Berdasarkan latar belakang yang telah diuraikan maka rumusan masalah dalam penelitian proposal tugas akhir ini adalah sebagai berikut:

1. Semua partikel dianggap sama dan mengikuti aturan yang sama.
2. Berbentuk 2 dimensi.
3. Dibatasi dengan gaya dan perilaku yang telah ditentukan.
4. Bentuk ruang mataf tempat partikel tawaf diserdeharnakan
5. Bagaimana proses simulasi tawaf berkelompok bebentuk.
6. Bagaimana variable yang sudah ditambahkan mempengaruhi simulasi.

1.5 Metode Pengumpulan Data

Dalam penelitian ini digunakan dua metode pengumpulan data yaitu:

1. Studi Literatur Sebelum melakukan ekperimen terlebih dahulu dilakukan studi literatur yang dapat bersumber dari berbagai buku, jurnal dan skripsi untuk mendapat informasi yang dapat dijadikan sebagai acuan selama penelitian.
2. Metode numerik Besaran-besaran pada simulasi akan diupdate menggunakan metode interasi Runge-Kutta
3. Simulasi Dalam simulasi ini digunakan agent-base model sebagai variable individu yang bergerak secara simulatan dengan variable yang telah ditentukan. Dalam model Masjidil Al-Haram.

1.6 Sistematika Penulisan

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

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

2. BAB II TEORI DASAR Bab ini berisi teori-teori penunjang penelitian diantaranya agent-base, Behaviour Reciprocal, Reciprocal Collition Avoidance, dll

3. BAB III METODOLOGI PENELITIAN Pada bab ini diuraikan tahap-tahap dalam penelitian. Tahapan tersebut meliputi: tahap pembacaan data, algoritma program, simulasi perjalanan agent dalam model yang telah dibentuk.

4. BAB IV HASIL SEMENTARA Pada bab ini akan dibahas tentang hasil sementara penelitian dan analisis yang dibahas dengan acuan dasar teori yang berkaitan dengan penelitian.

5. Bab V Kesimpulan dan Saran Pada bab ini memuat kesimpulan dan saran yang dapat diambil dari Hasil.

BAB 2

TEORI DASAR

2.1 Pengertian *Flocking*

Dalam melaksanakan tawaf gerak individu dapat dikaitkan dengan perilaku interaksi individual, karena bentuk kecepatan pergerakan orang pertama dalam melaksanakan tawaf bergantung dengan orang kedua didepannya dan orang didepannya tergantung dengan orang ketiga didepannya lagi dan seterusnya. Kita dapat meninjaunya dari segi fisika dengan mengasumsikan setiap individu adalah partikel. Partikel bergerak berkelompok karena kecepatan tetap dan arah setiap partikel resultan dari semua partikel berkelompok tersebut.

Proses resultan dari partikel gerak individual menuju partikel yang berkelompok diakibatkan oleh perubahan variabel dalam sistem gaya yang mempengaruh.Variabel yang digunakan tersebut terletak pada order parameter. Orde parameter ini yang diterapkan kedalam dinamika *flocking*.

Istilah *flocking* digunakan untuk menjelaskan gerak berkelompok khususnya pada kawanan burung (Toner & Tu, 1998). Dimana setiap variabel dalam kawanan burung bergerak dengan arah dan kelajuan yang relatif sama tanpa saling bertabrakan satu sama lain.

Dalam dinamika *flocking* yang dijalankan, terdapat *input state* dan *output state* yang digunakan dalam proses properti model partikel . Properti partikel ini berbentuk posisi, kecepatan, massa dan jarak gaya yang akan ditinjau dalam simulasi. Properti ini dalam (Bajec *et al.*, 2007) disebut dengan *current state properties* q. Yang dimaksud *current state properties* adalah karakteristik yang dimiliki setiap partikel.

$$q = p, \upsilon, z, m, \psi \tag{2.1}$$

2.2 Gaya dalam *Flocking*

Mekanika *Flocking* berdasarkan reynolds mempunyai 3 gaya yang digunakan untuk membentuk kerumunan (Reynolds, 1987) (Nasir & Sunar, 2016). Dimana hal ini dipakai oleh (Chaté *et al.*, 2008) dalam modelnya *boids model* untuk simulasi. Simulasi kerumunan akan terbentuk jika partikel bergerak sama dengan partikel sekitarnya, ,membentuk 1 titik area pemadatan partikel, dan mempunyai gaya pemisah agar partikel bergerak tanpa adanya partikel yang melekat/ tumbukan tidak elastis. Interaksi antar kerumunan ini dalam (Chaté *et al.*, 2008) dinamakan SPP(*Self-Propelled Particles*).

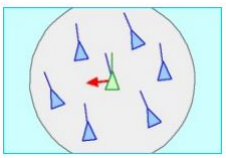
$$\vec{F_i} = \sum_{i \neq j} (\sum_{n=1}^{\infty} R_n \frac{\vec{P}_{ij}}{|\vec{P}_{ij}|^n} + \sum_{n=1}^{\infty} I_n \frac{\vec{P}_{ij}}{|\vec{P}_{ij}|^n}). \tag{2.2}$$

2.3 Aplikasi gaya

2.3.1 *Alignment*

Gaya *Allignment* digunakan untuk membuat partikel tetap bergerak satu grup. Berkaitan dengan sistem dalam tawaf, digunakan karena di dalam masjidil haram beberapa orang sering berkerumun untuk membentuk 1 grup agar tidak tersesat. Alasan lainnya, untuk penggunaan alignment ini membuat partikel menyamakan persepsi kecepatan dengan konsep yang terdepan akan mengurangi kecepatannya juga yang belakang akan menambah kecepatannya untuk berjalan bersamaan membentuk grup. secara matematika dapat ditulis sebagai berikut.

$$F_i^a(p,q) = \alpha^a \frac{1}{N^c} \sum_{i \neq j}^{N^c} \frac{\vec{V}_{ji}}{|\vec{V}_{ji}|}, \alpha^a = 1N \tag{2.3}$$

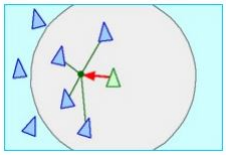


Gambar 2.1: Bentuk Gerak Pensejajaran

2.3.2 *Cohesion*

Gaya kohesi digunakan untuk membuat partikel menarik tetangganya untuk berdekatan. Dimana posisi pusat dijadikan titik arah pergerakan sehingga partikel bergerak ke titik tersebut: berikut secara matematik dapat ditulis:

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



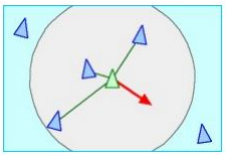
Gambar 2.2: Bentuk Gerak Pemusatan

2.3.3 *Separation*

Gaya separasi digunakan untuk partikel agar terjadi pemisahan dan meminimalisir pemisahan. Dalam konteks tawaf digunakan untuk menghindari seseorang menempal terhadap orang lain. dalam gaya separasi ini dilakukan variasi terhadap koefisiennya yaitu koefisien introversion atau tingkatjaga jarak dan tingkat racis untuk mengelompokkan jenis ras partikel berdasarkan warna.

$$F_i^s(p,q) = \alpha^s \sum_{i \neq j}^{N^c} \frac{\vec{p}_{ji}}{|\vec{p}_{ji}|^2}, \tag{2.5}$$

$$\alpha^a = radius + radiusboidstetangga + koefisienintroversion + koefisienracism \tag{2.6}$$



Gambar 2.3: Bentuk Gerak Pemisahan

2.3.4 *Circular Motion*

Circular Motion atau bisa disebut gaya gerak memutar mengelilingi pusat gaya, dapat diaplikasikan kepada partikel yang bergerak secara *flocking* untuk membuat partikel seperti bergerak tawaf mengelilingi pusat gaya. Dimana pusat gaya berada di pusat kaaba berbentuk persegi. Gerak *Flocking* yang diterapkan perlu memenuhi syarat area tawaf. Dengan begitu gerak mengeliligi kaaba dapat berada di area dekat kaaba.

$$f(x) = \begin{cases} F_m^a(p,q) = \alpha^m \sum_{i \neq j}^{N^c} \frac{\vec{p}_{ji}}{|\vec{p}_{ji}|}, \alpha^m = 1N, & \text{if x<50.} \\ F_m^a(p,q) = \alpha^m \sum_{i \neq j}^{N^c} \frac{\vec{p}_{ji}}{|\vec{p}_{ji}|}, \alpha^m = -1N, & \text{if x>50 and if x>300 .} \end{cases}$$

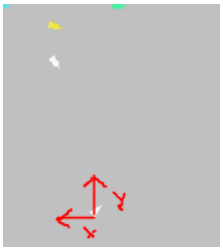
Pada persamaan pertama penerapan gaya menjauhi kaaba dengan syarat angka 50 jarak dari pusat kaaba sampai dinding. Sedangkan pada persamaan kedua penerapan gaya mendekati kaaba dengan asumsi partikel berada di luar area 50 sampai 300. Kedua gaya ini adalah gaya dominan yang diterapkan agar circular Motionnya bekerja.

Penerapan gaya terbagi dalam 3 area,yaitu area dibagian luar putar,area dimana tawaf diterapkan dan area dinding kaaba. Pada bagian luar diterapkan gaya tarik menuju pusat, dibagian diding kaaba diterapkan gaya keluar untuk menjaga dinding kaaba. di area diterapkan gaya putar untuk partikel melakukan tawaf.

2.4 Model Simulasi

2.4.1 Model Partikel

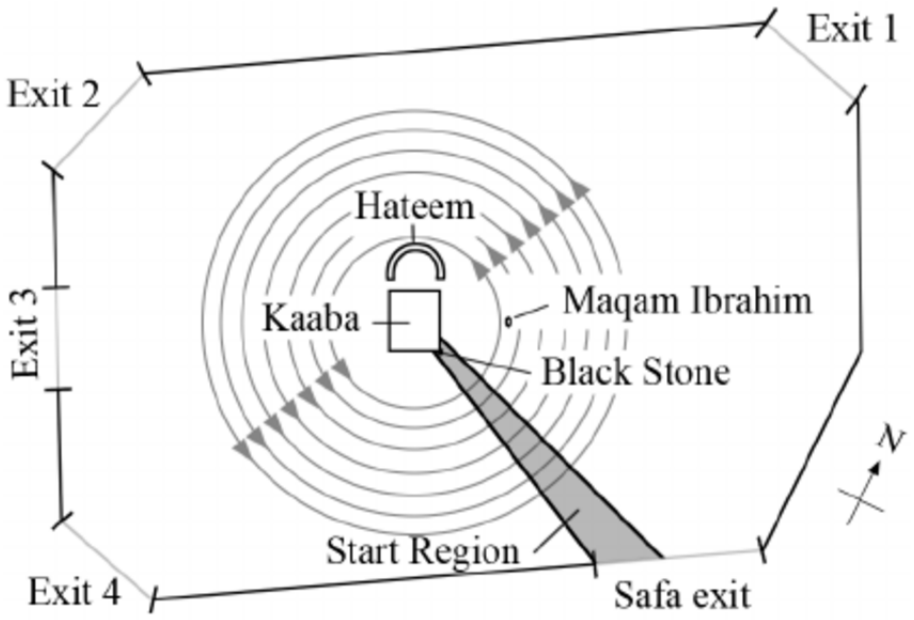
Model partikel sederhana dengan bentuk bulat menggambarkan partikel agent yang sedang bergerak



Gambar 2.4: Model partikel sebagai *boids*

2.4.2 Model Ruang Partikel

Model ruang Partikel diformulasikan seperti gambar masjidil haram yang tertera pada gambar. Partikel bergerak berlawanan arah dengan ruang dibatasi boundary state yaitu dinding masjidil haram dan bagian dalam ada boundary state lagi berupa dinding kaaba. Akan tetapi dalam simulasi ini model ruang partikel dibatasi hanya pada kaaba dan area tawaf kaaba.



Gambar 2.5: Model Area Masjidil Haram

2.5 Persamaan Distribusi Normal

Distribusi normal dalam program ini digunakan untuk menentukan nilai real dari koeffisien yang ada dalam nilai introversi, rasisme dan ketangkasan(quirkness). Hal

ini mempengaruhi bentuk dari inisial tawaf yang akan terjadi, dalam kasus ini secara acak.

$$f(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{1}{2}(\frac{x-\mu}{\sigma})^2} \tag{2.7}$$

Normal distribusi penting digunakan dalam statistik karena sering digunakan untuk representasi nilai real dari variabel random yang distribusinya tidak diketahui(Casella & Berger, 2001). Pada simulasi ini distribusi normal digunaakan sebagai perbandingan untuk koefisien partikel agar dapat divariasikan.

2.6 Metode Runge Kutta

Metode Runge-Kutta adalah metode numerik yang banyak digunakan untuk menyelesaikan persamaan differensial. Metode ini adalah versi yang lebih kompleks dari metode euler yang banyak digunakan. Karena kompleksitasnya maka metode ini menawarkan grafik eksponensial yang lebih mendekati fungsi eksak.
Untuk fungsi y

$$\frac{dy}{dt} = f(y,t), y(t_0) = y_0 \tag{2.8}$$

$$y_{n+1} = y_n + \frac{1}{6}h(k_1 + 2k_2 + 2k_3 + k_4), \tag{2.9}$$

$$Untukn = 0, 1, 2, 3, 4..... \tag{2.10}$$

$$k_1 = f(y_n, t_n), \tag{2.11}$$

$$k_2 = f(y_n + h\frac{k_1}{2}, t_n + \frac{h}{2}), \tag{2.12}$$

$$k_3 = f(y_n + h\frac{k_2}{2}, t_n + \frac{h}{2}), \tag{2.13}$$

$$k_4 = f(y_n + hk_1, t_n + h). \tag{2.14}$$

Untuk fungsi x

$$\frac{dx}{dt} = f(x,t), x(t_0) = x_0 \tag{2.15}$$

$$x_{n+1} = x_n + \frac{1}{6}h(k_1 + 2k_2 + 2k_3 + k_4), \tag{2.16}$$

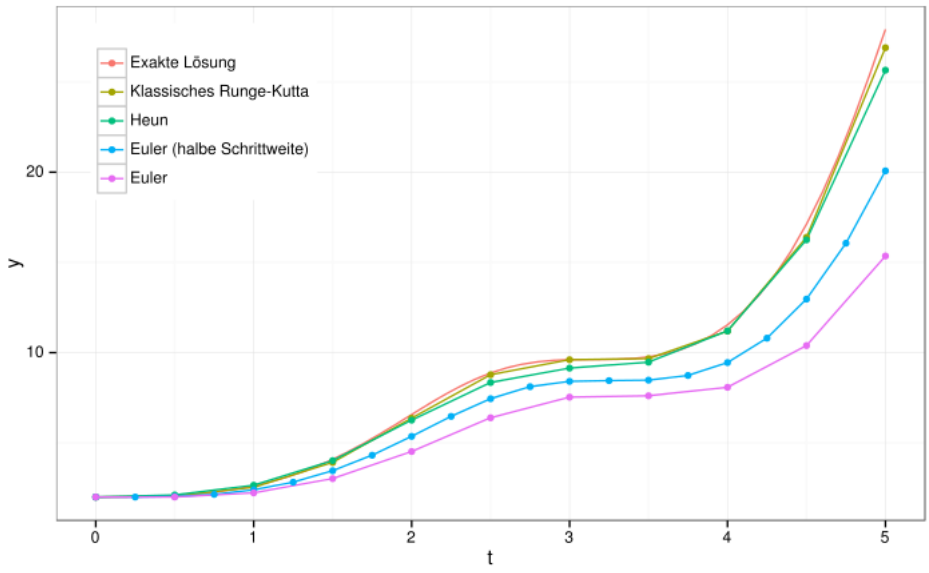
$$Untukn = 0, 1, 2, 3, 4..... \tag{2.17}$$

$$k_1 = f(x_n, t_n), \tag{2.18}$$

$$k_2 = f(x_n + h\frac{k_1}{2}, t_n + \frac{h}{2}), \tag{2.19}$$

$$k_3 = f(x_n + h\frac{k_2}{2}, t_n + \frac{h}{2}), \tag{2.20}$$

$$k_4 = f(x_n + hk_1, t_n + h). \tag{2.21}$$



Gambar 2.6: Perbandingan euler dan runge-kutta

Dalam simulasi ini penggunaan runge-kutta digunakan karena alokasi memori komputer yang digunakan tidak terlalu besar. Metode ini juga dipertimbangkan

dengan metode euler dimana tingkat error yang lebih rendah. Mempertimbangkan jumlah partikel yang digunakan maka diambil metode ini. Metode runge-kutta digunakan karena kinerja pada komputer ringan.

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.

No	Komponen	Spesifikasi
1	Processor	Intel(R) Core(TM) i5-4300M CPU @ 2.60GHz 2.59
2	Memory	12GB
3	Display	Intel(R) HD Graphics 4600
4	Renderer	-
5	OS	Windows 10

3.1.2 Perangkat Lunak

Perangkat lunak yang digunakan :

Tabel 3.2: Spesifikasi Perangkat Lunak.

No	Komponen	Software
1	Text Editor	Sublime
2	Browser	Google Chrome
3	Library	victor js library

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 Gaya

3.2.1 Penerapan 3 gaya

Pola flocking dimana partikel diproyeksikan bergerak membentuk formasi berkelompok sesuai dengan groupnya.Untuk hal tersebut diterapkan partikel disimulasikan dalam kasus dimana partikel dimunculkan pada tempat yang acak. setelahnya partikel akan bergerak dipengaruhi 3 gaya yaitu alignment, cohesion, dan separation. Sampai pola gerak berkelompok terbentuk

3.2.2 Penerapan 3 gaya ditambah gaya circular

$$\vec{F_i} = \vec{F^a}_i + \vec{F^k}_i + \vec{F^s}_i + \vec{F^c}_i \tag{3.1}$$

Pada kasus dengan tambahan gaya circular, partikel akan diterapkan gaya pusat circular sebagai prioritas agar partikel bergerak didalam area tawaf. dan ketika berada diarea tawaf partikel bergerak secara circular berlawanan arah jarum jam. dalam area tawaf ada gaya yang sama untuk mencegah agar partikel tetap berada di area tawaf. untuk prioritas gaya maka ada massa factor (m) sebagai besar factor prioritasnya.

$$\vec{F_i} = m^a\vec{F^a}_i + m^k\vec{F^k}_i + m^s\vec{F^s}_i + m^c\vec{F^c}_i \tag{3.2}$$

interaksi diluar zona tawaf

Maksud dari interaksi diluar tawaf. adalah partikel hanya bergerak menuju ke zona tawaf. Dengan gaya partikel lain yang minim. Hal ini menggiring partikel menuju pergerakan di area tawaf.

interaksi didalam zona tawaf

Pada kasus ini partikel berinteraksi dengan group partikel lain, juga berinteraksi dengan partikel yang berbeda group juga. Partikel dengan group sama diwakilkan dengan warna yang sama mempunyai gaya separasi yang lebih kecil dibandingkan dengan partikel yang berbeda groupnya.

Adapun gaya circular dalam simulasi digunakan untuk partikel melakukan putaran tawaf. Flocking juga dapat distabilkan jika kelompok partikel berkumpul dengan sejenisnya, Membentuk group sambil melakukan tawaf. Hal ini mencerminkan partikel menjaga eksistensi groupnya, sambil melakukan tawaf.

3.3 Simulasi Komputer

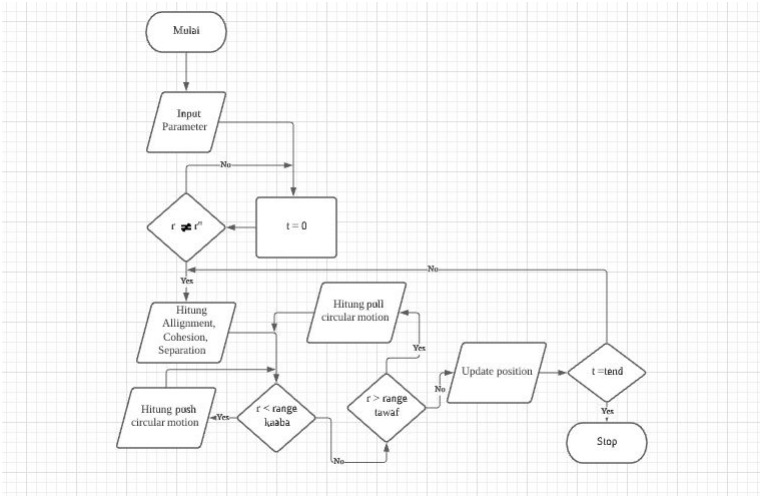
Simulasi dan program dibagun dalam bahasa JavaScript sebagai dasar codenya. secara visual menampikan animasi dan grafik yang menghasilkan outputnya.Bentuk matematis program dimuat dalam library Victorjs. Dimana fungsi kalkulasi vektor terdapat dalamnya, sehingga dapat dipanggil dengan mudah.

Tabel 3.3: Sekilas tentang pengunaan library VictorJS .

Penulisan koordinat gaya menggunakan victorjs	Penulisan koordinat gaya tanpa victor js menggunakan array javascript
var Vector = new Victor(diff.x,diff.y);	var Vector = [diff.y,diff.y]
sum.add(Vector)	Vector = [diff.y,diff.y] + [sum.y,sum.y]

Tabel diatas menggambarkan penulisan Koordinat menggunakan *library* Victor.js(Kueng & Crabtree, 2014). Gaya berbentuk array dalam program biasa dapat dikonversi kedalam *library* menggunakan victor.js, di dalam victor.js juga terdapat perintah untuk menjumlahkan gaya dengan singkat seperti terdapat pada no 2.

Terdapat banyak lagi selain penjumlahan vector, pengurangan, akar kuadrat dan lain-lain.



Gambar 3.1: Algoritma Pemograman.

3.4 Parameter Gaya

3.4.1 Parameter dalam pergerakan gaya

Dalam penerapannya, hasil simulasi kaaba diletakan di pusat dengan titik (sekian) dimana parameter penting lainnya adalah koordinat partikel yang dimunculkan secara acak dalam ruang interaksi zona tawaf maupun zona diluar tawaf. Partikel juga memiliki sebuah id untuk mengkatagorikan partikel dalam sebuah kelompok. Parameter selajutnya adalah gaya antar partikel yang diatur agar partikel membuat sebuah group, dalam hal ini, gaya separasi yang (koefisien *boid* < koefisien partikel lain) akan membuat partikel memilih jarak yang yang berbeda sesuai dengan group partikelnya. parameter lain yang berpengaruh adalah gaya tarik kedalam zona tawaf agar partikel membuat gerakan tawaf. Jumlah partikel N ditentukan untuk menguji kestabilan simulasi dalam hal ini dibatasi dengan jumlah orde 3 digit untuk menghindari adanya laging dalam proses simulasinya.

Dalam *interface*, gaya separasi dibentuk dari salah satu variabel yaitu jarak separasi(z_s) didalamnya dalam terdapat koefisien introversi dan koefisien rasis dimana koefisien ini membentuk radius maksimum untuk melakukan gaya separasi dimana desiredSeparation = radius boid + radius boid lain + (25 * koefisien introversion) + (50 * racismMultiplier koefisien). Pada koefisien racism diterapkan

logika bahwa jika partikel memiliki warna sama maka koefisien racism akan menjadi nol, dan jika tidak maka akan mempunyai nilai. Nilai ini yang mengaktifkan gaya separasi.

Parameter lain yang mempengaruhi gaya adalah jarak pensejajaran z_a, dimana ada aturan jarak yang dibutuhkan untuk gaya mempengaruhi boid agar melakukan pensejajaran dengan *boid* lain parameter jarak ini adalah 25.

Parameter lain yang mempengaruhi gaya adalah jarak kohesiz_k, dimana ada aturan jarak yang dibutuhkan untuk gaya mempengaruhi boid agar melakukan kompresi dengan boid lain parameter jarak ini adalah 25.

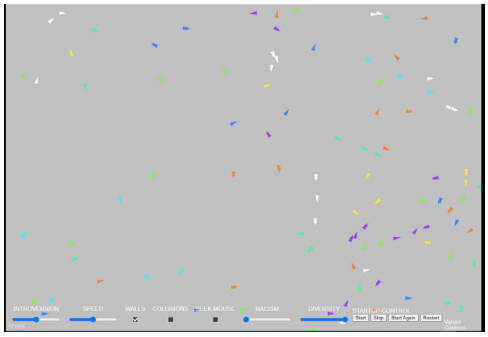
3.4.2 Parameter variasi partikel yang ditambahkan dalam simulasi

Pada simulasi ditambahkan partikel dengan speed yang berbeda. Terdapat 3 variasi kecepatan yang digunakan diantaranyaa quickness = 1 ,agroQuickness = 1.25 dan blackQuickness = 0.5. Pada partikel yang merah ditentukan parameter kecepatan yang 1.25 koefisien dari kecepatan normal. Pada partikell hitam ditentukan koefisien nya yaitu 0.45 dari kecepatan normal.

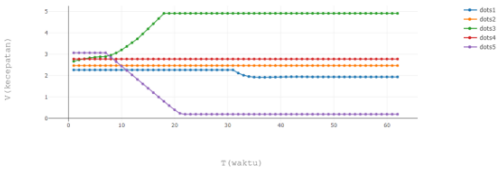
BAB 4

Hasil dan Pembahasan

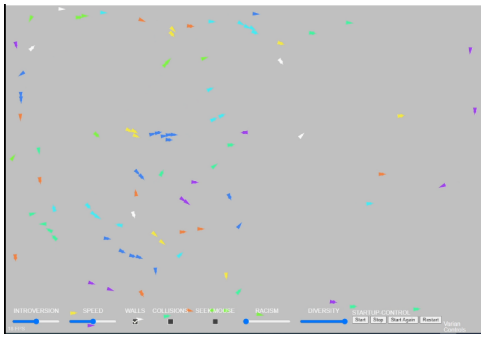
4.1 Flocking tanpa Circular Motion



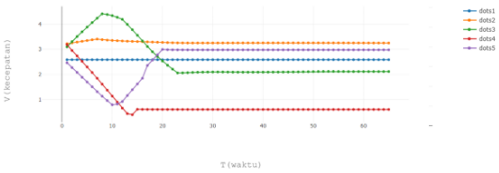
(a) Model Flocking dengan range 25 N=100



(b) Grafik Model Flocking dengan range 25 N=100



(c) Model Flocking dengan range 50 N=100



(d) Grafik Model Flocking dengan range 50 N=100

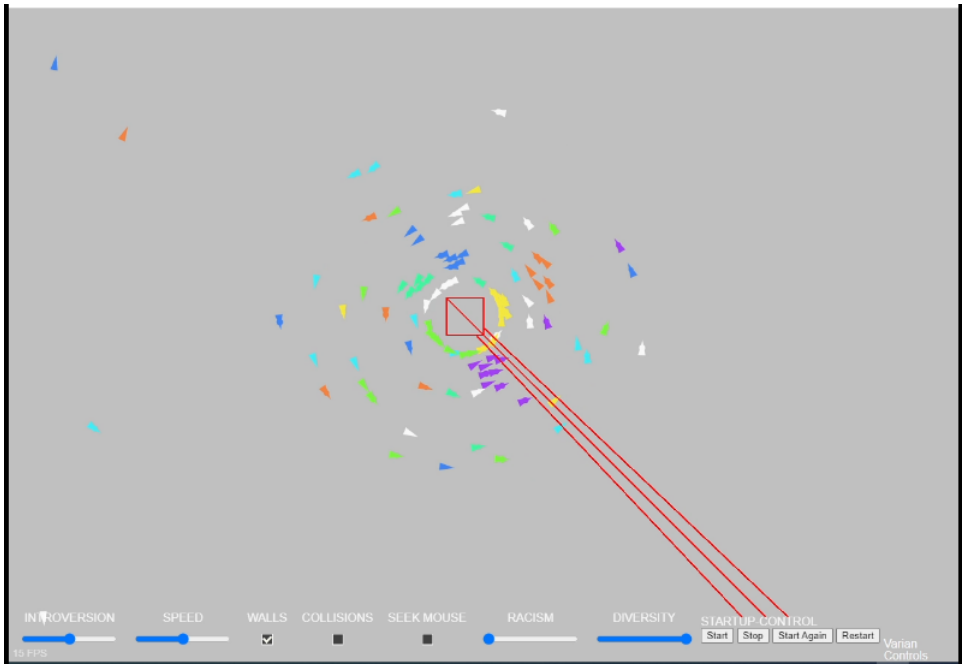
Gambar 4.1: Hasil Model alligment,separation dan kohesi dengan range 25 dan range 50

Hasil pada model *flocking* tanpa gerakan tawaf(*circular motion*) dengan *range flocking* yang berbeda pada kohesi dan pensejajarannnya, dapat dibandingkan kedua

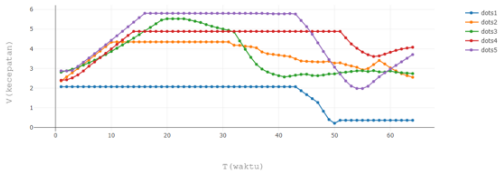
pengoptimalan *range* dalam menentukan pergerakan partikel . Hal ini dapat dibandingkan dengan (Huth & Wissel, 1992) juga yang membuat *range*, agar perbedaan pergerakan partikel dapat terlihat.Pada gambar 4.1 partikel yang mempunyai *range* 50 cenderung lebih mudah berkelompok dari pada *range* 25. hal ini wajar karena semakin besar *range* dalam *flocking* yang digunakan maka semakin mudah dalam partikel berkelompok. Pada partikel dengan *range* 50 berkelompok adalah 4 partikel sedangkan pada partikel dengan *range* 25 berkelompok adalah 2 partikel.

Pada grafik 4.1(b) partikel mulai dari kecepatan yang berbeda-beda sampai kecepatan tetap pada iterasi ke 10. Dan grafik4.1(d) sampai kecepatan tetap pada iterasi ke 20. Hal ini sesuai dengan kaidah *flocking* pada (Bajec *et al.*, 2007)

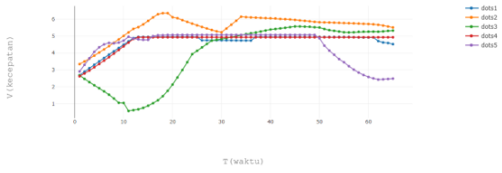
4.2 Flocking dengan Gaya Dominan(Circular Motion)



(a) Hasil Model dengan ditambahkan gaya dominan, N=100



(b) Grafik Model Flocking dengan gaya dominan, range 25 N=100



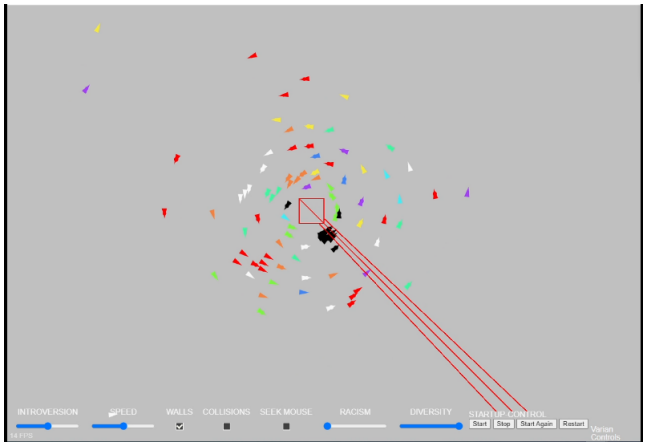
(c) Grafik Model Flocking dengan gaya dominan range 50 N=100

Pada bagian ini ditambahkannya gaya dominan pada persamaan 2.5 dimana gaya dominan ini berfungsi mengikat partikel untuk berputar pada porosnya.Terdapat perbedaan antara pergerakan tanpa circular motion adalah *flocking* hanya bekerja pada area tawaf. Dan partikel diluar area tersebut akan perlahan mendekati

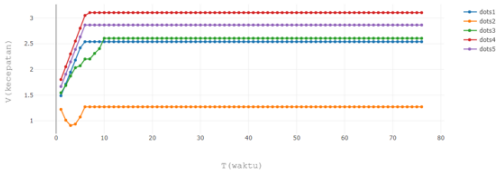
area tawaf. Hal ini dilakukan untuk mendekati (Nasir & Sunar, 2016) dan (Kim *et al.*, 2014). Dimana apa yang dilakukan nasir adalah mengelompokan partikel dalam sebuah group dikelilingi partikel yang homogen. sedangkan pada (Kim *et al.*, 2014) mengamplikasikan *flocking* ditambahkan dengan reciprocal coalition avoidance. untuk memperlihatkan interaksi partikel yang lebih tidak bersinggungan. Jika dapat dianalisis hasil pada 4.2(a) didapat konsep (Nasir & Sunar, 2016) yang menggolongkan partikel dan konsep dari (Kim *et al.*, 2014) untuk *flocking* berbasis kecepatan.

Pada kedua grafik, model *flocking* dengan gaya dominan 4.2(b),4.2(c) menggunakan 2 variabel *range* yang berbeda yaitu 25 dan 50. Kedua grafik jika dibandingkan dengan hasil grafik awal tidak mencapai keadaan *flocking*. Hal ini dipengaruhi oleh gaya dominan, untuk mencapai *flocking* yang linier maka dilakukan simulasi kembali pada bagian bab selanjutnya.

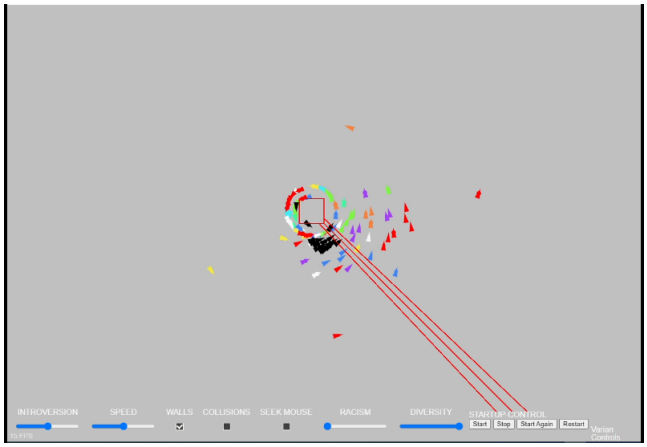
4.3 Flocking dengan Gaya Dominan(Circular Motion) partikel dengan Karakteristik



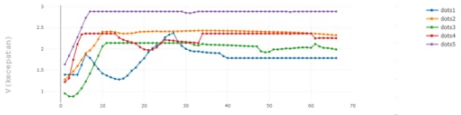
(d) Model Flocking dengan range 25 N=100 gaya dominan tawaf



(e) Grafik Model Flocking dengan range 25 N=100 gaya dominan tawaf



(f) Model Flocking dengan range 50 N=100 gaya dominan tawaf



(g) Grafik Model Flocking dengan range 50 N=100 gaya dominan tawaf

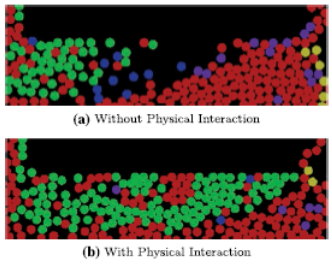
Dan terakhir pada bagian ini kami menambahkan 2 jenis partikel dengan kecepatan yang berbeda di kedua sistem *range* yang berbeda. Disini dapat terlihat pada gambar 4.2 (kanan) ternyata *range* yang lebih besar mendistrupsi sistem *flocking* pada gerakan tawaf. Hal ini dapat terlihat dari partikel yang tertarik menuju pusat. Hasil yang berbeda didapat dari sistem yang menggunakan *range* lebih kecil 4.2(kiri). Partikel cenderung melakukan menyebar dalam area tawaf dan melakukan *flocking* dengan lebih optimal.

Penambahan variabel dari kecepatan partikel ini (merah dengan 1.25 kecepatan partikel biasa) dan (hitam dengan 0.75 partikel biasa). dapat memperlihatkan pada bagian optimal gambar 4.2(kiri). Bahwa partikel hitam cenderung bergerak di pusat sedangkan partikel merah berada di luar. Hal ini disebabkan karena gaya dominan melebihi gerak *flocking*.

Sedangkan dengan sudut pandang grafik, pada grafik 4.2(e) kecepatan tetap pada titik tertentu dibandingkan grafik 4.2(g). Hal ini karena variasi penulis untuk mendapatkan hasil yang linier mengikuti kaidah *flocking* bahwa kecepatan partikel akan tetap pada titik tertentu seperti dalam (Bajec *et al.*, 2007). Gaya dominan seperti pada grafik 4.2(b),4.2(c) tidak mengganggu dengan hasil 4.2(e)

4.4 Perbandingan Hasil Simulasi dengan Beberapa Referensi

4.4.1 Referensi Hasil Simulasi dengan (Kim *et al.*, 2014)

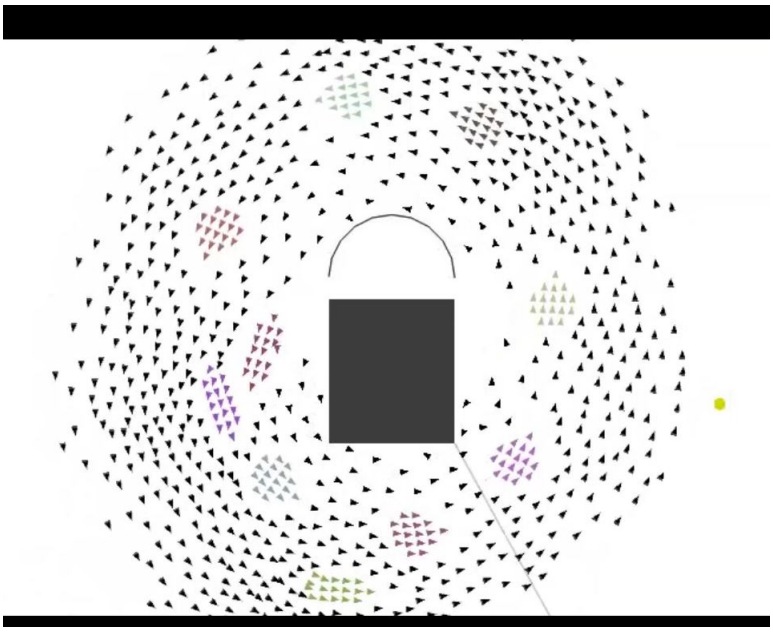


Gambar 4.3: Hasil Gerak Partikel pada (Kim *et al.*, 2014)

Pada gambar 4.3 diperlihatkan hasil kim bahwa partikel (agen) terpadatkan di dekat pusat area gaya(kaaba). Hal ini dikarenakan terdorong oleh kerumunan agen. Pada gambar agen hijau menunjukan agen yang menunggu menyentuh titik pusat

gaya(hajar aswad). Diawal simulasi agent berada dalam keadaan tidak dapat bergerak dan menghasilkan kerumunan padat. Tanpa interaksi fisis meskipun ada ruang gerak, partikel agent tidak bisa memproses ke posisi selanjutnya. Dengan tambahan interaksi fisis dengan agent selanjutnya agent dapat terdorong oleh kerumunan.

4.4.2 Referensi Hasil Simulasi dengan (Nasir & Sunar, 2016)

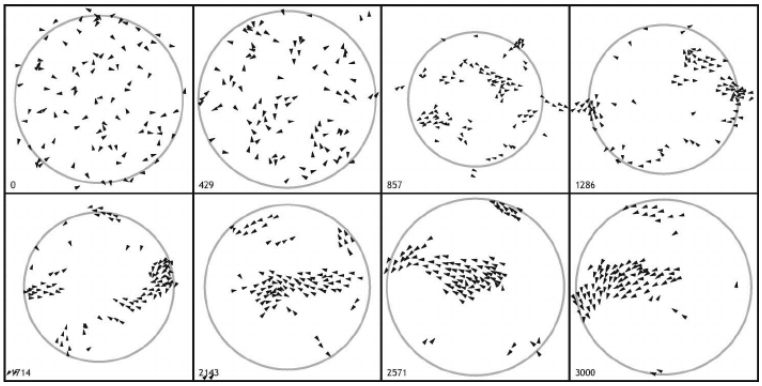


Gambar 4.4: Hasil model oleh Nasir

Bentuk model tawaf dari nasir(Nasir & Sunar, 2016) ditunjukan pada gambar diatas. Perilaku partikel berkelompok pada nasir memiliki kecepatan yang sama atau tidak lebih cepat dari partikel sekitarnya. Hal ini yang membuat formasi lebih solid, jika dibandingkan.

Hasil penelitian yang telah dilakukan oleh (Nasir & Sunar, 2016) ditunjukan oleh 4.4. Lingkungan yang dimiliki partikel diletakan kabah berbentuk persegi bersama dengan area bidang perisai untuk hijir ismail. Partikel bergerak berkelompok bersama dengan partikel sejenis dan membuat gerakan tawaf. Hal ini jika dibandingkan dengan dengan hasil pada 4.2(kiri) terdapat banyak perbedaan, diantaranya pergerakan partikel yang cenderung lebih mendekati pusat gaya(kaaba) dan kelompok lebih dapat terpencar, kemudian membuat kelompok yang lebih kecil lagi.

4.4.3 Referensi Hasil Simulasi dengan (Bajec *et al.*, 2007)



Gambar 4.5: Hasil Flocking Bajec

Hasil penelitian yang telah dilakukan oleh (Bajec *et al.*, 2007) ditunjukan oleh 4.5. Area bergerak 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 4.1 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.

4.4.4 Referensi Hasil Simulasi dengan (Nasir & Sunar, 2016)

BAB 5

PENUTUP

5.1 Kesimpulan

Berdasarkan hasil penelitian dan study literatur dapat disimpulkan bahwa:

1. Pergerakan tawaf seperti nasir dengan partikel dapat flocking(mencapai kecepatan yang tetap), dapat membentuk kelompok dan mengelilingi Kaaba, sedangkan skema dalam bergerak mendekati pergerakan tawaf hasil kim. Hasil simulasi sendiri merupakan gabungan keduanya dengan beberapa penyesuaikan.

2. Pergerakan partikel berkelompok yang dapat mendekati Kaaba(terutama pada partikel hitam dalam optimasi) adalah partikel yang paling sesuai dengan flocking.

3. Penggunakan runge-kutta,dikunakan untuk pemedatan data dalam bentuk grafik. Berdasarkan simulasi yang telah dilakukan.

5.2 Saran

Penelitian ini hanya berfokus pada pembuatan mekanisme flocking pada gerak tawaf. Faktor-faktor fisis seperti gesekan partikel dengan medium yang dilalui dan bentuk skema sebelum tawaf terjadi tidak diperhitungkan. Begitu juga dengan object yang ada dalam garis tawaf sebenarnya kecuali kaaba sebagai pusat gaya.

DAFTAR PUSTAKA

Bajec, I. Lebar, Zimic, N., & Mraz, M. 2007. The computational beauty of flocking: boids revisited. *Mathematical and Computer Modelling of Dynamical Systems*, **13**(4), 331–347.

Bicho, Alessandro, Rodrigues, Rafael, Musse, Soraia, Jung, Claudio, Paravisi, Marcelo, & MagalhÃ£es, LÃ©o. 2012. Simulating crowds based on a space colonization algorithm. *Computers Graphics*, **36**(04), 70â€“79.

Casella, George, & Berger, Roger. 2001. *Statistical Inference*. Duxbury Resource Center.

Chaté, Hugues, Ginelli, Francesco, Grégoire, Guillaume, & Raynaud, Franck. 2008. Collective motion of self-propelled particles interacting without cohesion. *Phys. Rev. E*, **77**(Apr), 046113.

Curtis, Sean, Guy, Stephen, Zafar, Basim, & Manocha, Dinesh. 2011 (11). Virtual Tawaf: A case study in simulating the behavior of dense, heterogeneous crowds.

for Statistics Kingdom of Saudi Arabia, General Authority. 2018. Hajj Statistics. *Hajj statisics 1439*, **1**(12).

Haghighati, Razieh, & Hassan, Adnan. 2013. MODELING THE FLOW OF CROWD DURING TAWAF AT MASJID AL-HARAM. *Jurnal Mekanikal*, 06.

Huth, Andreas, & Wissel, Christian. 1992. The simulation of the movement of fish schools. *Journal of Theoretical Biology*, **156**(3), 365–385.

Khan, Imran, & McLeod, Robert Donald. 2012. MANAGING HAJJ CROWD COMPLEXITY : SUPERIOR THROUGHPUT , SATISFACTION , HEALTH , & SAFETY.

Kim, Sujeong, Guy, StephenJ., Hillesland, Karl, Zafar, Basim, Gutub, AdnanAbdul-Aziz, & Manocha, Dinesh. 2014. Velocity-based modeling of physical interactions in dense crowds. *The Visual Computer*, 1–15.

Kueng, Max, & Crabtree, George. 2014. *Victor.js - 2D Vectors for JavaScript*.

Lim, Eng Aik, & Zainuddin, Zarita. 2012. Response Surface Analysis of Crowd Dynamics during Tawaf. *Chinese Physics Letters*, **29**(7), 078901.

Mulyana, Willy, & Gunawan, Teddy. 2010 (06). Hajj crowd simulation based on intelligent agent.

Narain, Rahul, Golas, Abhinav, Curtis, Sean, & Lin, Ming. 2009. Aggregate Dynamics for Dense Crowd Simulation. *ACM Trans. Graph.*, **28**(12).

Nasir, Fawwaz Mohd, & Sunar, Mohd Shahrizal. 2016. Simulating large group behaviour in tawaf crowd. *Pages 42–46 of: 2016 Asia Pacific Conference on Multimedia and Broadcasting (APMediaCast)*.

Reynolds, Craig W. 1987. Flocks, Herds, and Schools: A Distributed Behavioral Model. *In: (ACM SIGGRAPH Computer Graphics*. ACM.

Saiwaki, N., Komatsu, T., Yoshida, T., & Nishida, S. 1997. Automatic generation of moving crowd using chaos model. *Pages 3715–3721 vol.4 of: 1997 IEEE International Conference on Systems, Man, and Cybernetics. Computational Cybernetics and Simulation*, vol. 4.

Shuaibu, Aliyu. 2015. Simulation of Crowd Movement in Spiral Pattern during Tawaf, in Makkah, Saudi Arabia. *Modern Applied Science*, **9**(09), 192.

Toner, John, & Tu, Yuhai. 1998. Flocks, herds, and schools: A quantitative theory of flocking. *Phys. Rev. E*, **58**(Oct), 4828–4858.

Zainuddin, Zarita, Thinakaran, Kumatha, & Abu-Sulyman, Ibtesam. 2009. Simulating the Circumambulation of the Ka'aba using SimWalk. *European Journal of Scientific Research ISSN*, **38**(12), 1450–216.

INDEX

Vicsek **bab** 1;
W
Weighting Factor **bab** 2;

Lampiran A

sumber kode

A.1 Kode Program gambarBoids.js

```
function draw(){
        var c = arguments[0];
        var ctx = c.getContext("2d");
        let x=arguments[1];
        let y=arguments[2];
        let vx=arguments[3];
        let vy=arguments[4];
        let radius=arguments[5];
        let sudut1=arguments[6];
        let sudut2=arguments[7];
        let colored =arguments[8];
        let r=5; //3
        let theta=1*Math.atan2(vy,vx) + Math.PI / 2;
        ctx.save();
        ctx.beginPath();
        ctx.translate(x,y);
        ctx.rotate(theta);
        //ctx.fillStyle="rgba(255,000,000,1)"; //this.color
        ctx.fillStyle= colored;
        //ctx.arc(0,0,radius,-sudut1,-sudut2,true);
        //ctx.arc(0,0,radius,1.2*Math.PI,1.8*Math.PI);
        ctx.moveTo(0,-r*2);
```

```
        ctx.lineTo(r,r*2);
        ctx.lineTo(-r,r*2);
        ctx.lineTo(0,-r*2);
        ctx.fill();
        ctx.closePath();
        ctx.restore();
        //draw(caOut,X,Y,f[i].vx,f[i].vy,radiusA,sudut1,sudut2);
}
function clear(){
        var ca = arguments[0];
        var ctx = ca.getContext("2d");
        ctx.beginPath();
        ctx.clearRect(0,0,caOut.width,caOut.height);
        ctx.closePath();
}
```

A.2 Kode Program sides2.js

```
class Sides2 {
        //create constructor
        constructor() {
                if(arguments.length == 0){
                        this.p = [
                        new Vect3(),
                        new Vect3()
                        ];
                } else if(arguments.length == 2) {
                        this.p = [
                        arguments[0],
                        arguments[1]
                        ];
                } else if(arguments.length == 1){
                        if(arguments[0] instanceof Sides2)
                        this.p = [
                                arguments[0].p[0],
```

```
                                arguments[0].p[1]
                        ];
                }
        }
        // Get string value
        strval() {
                var s = "(";
                s += this.p[0].strval() + ", ";
                s += this.p[1].strval() + ", ";
                // s += this.p[2].strval() + ", ";
                // s += this.p[3].strval() + ", ";
                s += ")";
                return s;
        }

        // Get center coordinate
        center() {
                var N = this.p.length;
                var p0 = new Vect3();
                for(var i in this.p) {
                        p0 = Vect3.add(p0, this.p[i]);
                }
                p0 = Vect3.div(p0, N);
                return p0;
        }
}
```

A.3 vect3.js

```
/*
        vect3.js
        Vector in 3-d Cartesian coordinate system

        Sparisoma Viridi | dudung@gmail.com
```

```
        20171226
        Create this object (again).
        20171227
        Add some comments for clearer documentation.
*/

// Define class of Vect3
class Vect3 {
        // Create three different types of constructor
        constructor() {
                if(arguments.length == 0) {
                        this.x = 0.0;
                        this.y = 0.0;
                        this.z = 0.0;
                } else if(arguments.length == 3) {
                        this.x = arguments[0];
                        this.y = arguments[1];
                        this.z = arguments[2];
                } else if(arguments.length == 1) {
                        if(arguments[0] instanceof Vect3)
                        this.x = arguments[0].x;
                        this.y = arguments[0].y;
                        this.z = arguments[0].z;
                }
        }

        // Get string value
        strval() {
                var s = "(";
                s += this.x + ", ";
                s += this.y + ", ";
                s += this.z;
                s += ")";
                return s;
        }
```

```
// Add some Vect3
static add() {
        var N = arguments.length;
        var x = 0.0;
        var y = 0.0;
        var z = 0.0;
        for(var i = 0; i < N; i++) {
                x += arguments[i].x;
                y += arguments[i].y;
                z += arguments[i].z;
        }
        var p = new Vect3(x, y, z);
        return p;
}

// Substract two Vect3
static sub() {
        var x = arguments[0].x - arguments[1].x;
        var y = arguments[0].y - arguments[1].y;
        var z = arguments[0].z - arguments[1].z;
        var p = new Vect3(x, y, z);
        return p;
}

// Multiply Vect3 with scalar or vice versa
static mul() {
        var a = arguments[0];
        var b = arguments[1];
        var x, y, z;
        if(a instanceof Vect3) {
                x = a.x * b;
                y = a.y * b;
                z = a.z * b;
        } else if(b instanceof Vect3) {
                x = a * b.x;
                y = a * b.y;
```

```
                z = a * b.z;
        }
        var p = new Vect3(x, y, z);
        return p;
}

// Divide Vect3 with scalar
static div() {
        var a = arguments[0];
        var b = arguments[1];
        var     x = a.x / b;
        var     y = a.y / b;
        var     z = a.z / b;
        var p = new Vect3(x, y, z);
        return p;
}

// Dot two Vect3
static dot() {
        var a = arguments[0];
        var b = arguments[1];
        var     xx = a.x * b.x;
        var     yy = a.y * b.y;
        var     zz = a.z * b.z;
        var d = xx + yy + zz;
        return d;
}

// Cross two Vect3
static cross() {
        var a = arguments[0];
        var b = arguments[1];
        var     x = a.y * b.z - a.z * b.y;
        var     y = a.z * b.x - a.x * b.z;
        var     z = a.x * b.y - a.y * b.x;
        var p = new Vect3(x, y, z)
```

```
            return p;
        }

        // Get length of a Vect3
        len() {
            var l = Math.sqrt(Vect3.dot(this, this));
            return l;
        }

        // Get unit vector of a Vect3
        unit() {
            var l = this.len();
            var p = Vect3.div(this, l);
            return p;
        }

        // create for 2d cross
        static cross(){
            var a = arguments[1];
            var b = arguments[2];
            var x = (a.x*b.y) - (a.y*b.x);
            var p = new Vect3()
        }

}
```

A.4 Kode Program index.html

```
<!DOCTYPE html>
<html lang="en">
<head>
        <meta charset="utf-8">
        <title>BOIDS!</title>
        <link rel="stylesheet" href="css/style.css">
        <!--[if IE]>
```

```
                <script src="http://html5shiv.googlecode.com/svn/tru
        <![endif]-->
        <script src="https://cdn.plot.ly/plotly-latest.min.js"></sc
</head>

<body id="home">

  <div id="boids-wrapper">
    <canvas id="boids"></canvas>
                <div id="boids-controls-container">
                        <div id="mobile-boids-controls">
                                <button id="introversion-mobile">In
                                <button id="speed-mobile">Speed</bu
                                <button id="walls-mobile" class="bo
                                <button id="collisions-mobile">Coll
                                <button id="mouse-seek-mobile">Seek
                                <button id="racism-mobile">Racism</
                                <button id="diversity-mobile">Diver
                        </div>
                        <div id="introversion-control-container" cl
                                <span class="boids-control-close"><
                                <div class="range-slider">
                                        <label for="introversion"><
                                        <input class="input-range"
                    <span class="range-value"></span>
                        </div
>                       </div>
                        <div id="speed-control-container" class="bo
                                <span class="boids-control-close"><
                                <div class="range-slider">
                                        <label for="introversion"><
                                        <input class="input-range"
                    <span class="range-value"></span>
                        </div>
                        </div>
                        <div class="boids-control boids-control-che
```

```
        <div class="checkbox">
                <p>Walls</p>
                <input type="checkbox" id="
                <label for="walls"></label>
        </div>
</div>
<div class="boids-control boids-control-che
        <div class="checkbox">
                <p>Collisions</p>
                <input type="checkbox" id="
                <label for="collision-detec
        </div>
</div>
<div class="boids-control boids-control-che
        <div class="checkbox">
                <p>Seek Mouse</p>
                <input type="checkbox" id="
                <label for="mouse-seek"></la
        </div>
</div>
<div id="racism-control-container" class="b
        <span class="boids-control-close"><
        <div class="range-slider">
                <label for="introversion"><
                <input class="input-range"
    <span class="range-value"></span>
        </div>
</div>
<div id="diversity-control-container" class
        <span class="boids-control-close"><
        <div class="range-slider">
                <label for="introversion"><
                <input class="input-range"
    <span class="range-value"></span>
        </div>
</div>
```

```
                        <div id="fps">
                            <p><span id="fps-number"></span> fps
                        </div>

                        <div class="startup-control">

                                <p>Startup-Control</p>
                                <button id="Start" onlick="
                                <button id="Stop" onlick="st
                                <button>Start Again</button>
                                <button>Restart</button>
                            </div>
                        </div>
                        <div id="varian-">
                        <p>Varian Controls</p>
                    </div>



            </div>
            <p >Varian Controls</p>
    <script src="js/sides2.js"></script>
    <script src="js/vect3.js"></script>
    <script src="js/victor.min.js"></script>
    <script src="js/script.js"></script>
    <script src="js/boid.js"></script>
    <script src="js/gambarBoids.js"></script>

</body>
</html>
```

A.5 Kode Program boids.js

```
class Boid {
```

```
//constructor boids
 constructor(boid) {

   // Initial Properties
   this.id = boid.id;

   this.position = new Victor( boid.x, boid.y );
   this.positionI = new Victor();

   // for loop
   this.k1 = new Victor();
   this.k2 = new Victor();
   this.k3 = new Victor();
   this.k4 = new Victor();
   this.bun = new Victor();
   this.h = new Victor(1,1);
   this.half = new Victor(0.5,0.5);
   this.two = new Victor(2,2);
   this.Six = new Victor(6,6);
   //

   this.positionTheta = Math.atan2();
   this.distanceFromCenter0 = this.position.distance(center);
   this.coordinateO = new Victor (0,0);
   this.x ;
   this.y ;

   this.radius = boid.radius * radiusCoefficients[ boid.radiusCoef
   this.introversionCoefficient = boid.introversionCoefficient;
   this.introversion = boid.introversion * this.introversionCoeffi
   this.quicknessCoefficient = boid.quicknessCoefficient;
   this.quickness = boid.quickness * this.quicknessCoefficient;
   this.racismCoefficient = boid.racismCoefficient;
   this.racism = boid.racism * boid.racismCoefficient;
   this.color = boid.color;
   this.volume = (4/3) * Math.PI * Math.pow( this.radius,3 );
```

```
;
    this.angle;
    this.angle2 = 3/2*Math.PI;


    this.distanceFromCenter = this.position.clone().distance(center
    this.distanceFromCenter = this.position.distance(center);

    this.positionC = new Victor( this.distanceFromCenter*Math.cos(t
    this.theta = 0;
    this.frequency = 0 ;
    this.radians =  Math.PI*2;//*Math.random();

    // Speed & Velocity & Force
    this.maxSpeed = speedIndex * this.quickness;
    this.speed = this.maxSpeed * .5;
    this.speedangular = this.speed/this.distanceFromCenter;

    this.angularVelocity = this.speed/this.distanceFromCenter;

    var radians = Math.PI; //* getRandomInt(-99,100) / 100;

    this.velocity = new Victor( this.speed * Math.cos( radians ), t
    this.velocity0 = new Victor(Math.cos(this.radians)*this.distance
  //Force and Accel
    this.maxForce = 0.25//.5;
    this.maxForce2 = .1;
    this.acceleration = new Victor(0,0);
    this.angularAcceleration;
    this.desiredPosition;
    this.target;
  }
////////////// meta transform
    transform() {
    var realPosition = transform({x:this.position.x ,y:this.positio
```

```
  }
////////////////

  seek( target ){
    var targetposition = target.clone();
    var diff = targetposition.subtract(this.position);
    var desired = new Victor(diff.x,diff.y);
    this.desiredPosition = desired;
    this.target = diff;

    //area buffer seek biar seaknya beda
    if (target.radius) {
      var buffer = target.radius + this.radius + 1;
    } else {
      var buffer = this.radius * 2 + 1;
    }
    //
    var dist = diff.magnitude();
    if (dist < buffer) {
      desired.x = 0;
      desired.y = 0;
    } else if ( dist <= 100 ) {
      //normalize and multiply
      desired.normalize();
      desired.divide({x:this.maxSpeed * dist / 100,y:this.maxSpeed
    } else {
      desired.limitMagnitude(this.maxSpeed);
    }
    desired.subtract(this.velocity);
    desired.limitMagnitude(this.maxForce);
    return desired;
  }

  //this must edited to seek center canvas target position
   centripetal( boids ){
```

```
var sum = new Victor();
var steer = new Victor();
var u = new Victor(100,100);
var t = 0,dt=1;
for (var i = 0; i < boids.length; i++) {
  var desiredtoCenter = Math.pow(this.velocity,2)/distanceFromC
  var nowPosition = this.position.clone();
  var vecKaaba = Victor.fromObject(wallsKaaba);
  var distanceFromCenter =this.position.clone().distance(center

  var centerKaaba = new Victor(surf.center().x,surf.center().y)
  var distanceFromCenter2 = this.position.clone().distance(cente
  var pathCircular = 2*Math.PI* distanceFromCenter;
  var theta = 1;


    var N = sides.length;

    if ( (distanceFromCenter2 > 0 &&distanceFromCenter2 < 50) )


    var thisposition = this.position.clone();
    var diff = thisposition.subtract(centerKaaba);
    var diffVelocity = diff;
    diff.normalize();
    diff.multiply({x:distanceFromCenter,y:distanceFromCenter});

    sum.add(diff);
    t+=dt
    }
    //force to area that make boids circular path
    if ( (distanceFromCenter2 > 50 && distanceFromCenter2 <= 50

    var thisposition = this.position.clone();
    var diff = thisposition.subtract(centerKaaba);
```

```
        const diffVelocity = diff;

        diff.normalize();
        diff.multiply({x:distanceFromCenter,y:distanceFromCenter});

        sum.subtract(diff);

        t+=dt
        }else{

        }
        //
      //}

    }
    if (t > 0) {
      sum.divide({x:t,y:t});
      sum.normalize();
      sum.multiply({x:this.maxSpeed,y:this.maxSpeed});
      steer = sum.add(this.velocity);
      steer.limitMagnitude(this.maxForce);
      return steer;
    } else {
      return steer;
    }
  }


    circularMotion( boids ){
    var sum = new Victor();
    var steer = new Victor();
    var t = 0,dt=1;
    for (var i = 0; i < boids.length; i++) {

      var desiredtoCenter = Math.pow(this.velocity,2)/distanceFromC
      var distanceFromCenter =this.position.clone().distance(center
```

```
    var pathCircular = 2*Math.PI* distanceFromCenter;
    var theta = 1;

    if ( (distanceFromCenter < 50 && distanceFromCenter < 300 ) )
      var thisposition = this.position.clone();
      var thisvelocity = this.velocity.clone();
      var thispositionX = thisposition.x + (distanceFromCenter*Ma
      var thispositionY = thisposition.y + (distanceFromCenter*Ma
      var diff = thisposition.add(new Victor(thispositionX,thispo
      diff.normalize();

      sum.add(diff);
      t+=dt
    }
  }
  if (t > 0) {
    sum.divide({x:t,y:t});
    sum.normalize()
    sum.multiply({x:this.maxSpeed,y:this.maxSpeed});
    steer = sum.subtract(this.velocity);
    steer.limitMagnitude(this.maxForce);
    return steer;
  } else {
    return steer;
  }
}


separate( boids ){
  var sum = new Victor();
  var t = 0,dt=1;

  for (var j = 0; j < boids.length; j++) {
    if ( this.color != boids[j].color ) {
      var racismMultiplier = this.racism;
```

```
    } else {
      var racismMultiplier = 0;
    }
    var desiredSeparation = this.radius + boids[j].radius + ( 25
    var sep = this.position.clone().distance(boids[j].position);
    if ( (sep > 0) && (sep < desiredSeparation) ) {
      var thisposition = this.position.clone();
      var diff = thisposition.subtract(boids[j].position);
      diff.normalize();
      diff.divide({x:sep,y:sep});
      sum.add(diff);
      t+=dt
    }
  }
  if (t > 0) {
    sum.divide({x:t,y:t});
    sum.normalize();
    sum.multiply({x:this.maxSpeed,y:this.maxSpeed});
    sum.subtract(this.velocity);
    sum.limitMagnitude(this.maxForce);
  }
  return sum;
}


align( boids ) {
  var neighborDist = 50;//50
  var sum = new Victor();
  var steer = new Victor();
  var t = 0,dt=1;
  for (var i = 0; i < boids.length; i++) {
    var dist = this.position.distance(boids[i].position);
    if ( dist > 0 && dist < neighborDist ) {
      sum.add(boids[i].velocity);
      t+=dt;
    }
```

```
  }
  if (t > 0) {
    sum.divide({x:t,y:t});
    sum.normalize()
    sum.multiply({x:this.maxSpeed,y:this.maxSpeed});
    steer = sum.subtract(this.velocity);
    steer.limitMagnitude(this.maxForce);
    return steer;
  } else {
    return steer;
  }
}


cohesion( boids ) {
  var neighborDist = 50;//50
  var sum = new Victor();
  var t = 0,dt=1;
  for (var i = 0; i < boids.length; i++) {
    var dist = this.position.distance(boids[i].position);
    if ( dist > 0 && dist < neighborDist ) {
      sum.add(boids[i].position);
      t+=dt;
    }
  }
  if (t > 0) {
    sum.divide({x:t,y:t});
    return this.seek(sum);
  } else {
    return sum;
  }
}


avoidWalls() {
```

```
  var buffer = mobile ? 5 : 15;

  if ( this.distanceFromHorWall() < this.radius * buffer || this.
    return this.seek(center);
  } else { return false; }

}


flock() {

  // Get Forces

  var alignForce = this.align(boids);
  //console.log(alignForce);

  var circularMotionForce = this.circularMotion(boids);
  if ( mouseSeek ){
    var mouseForce = this.seek(center);//mouse.position
    var x = new Victor(0,0 );
    var mouseForce2 = this.seek(x);
  }
  //if ( mouseSeek ) var mouseForce = this.seek(mouse.position);/
  var separateForce = this.separate(boids);
  var cohesionForce = this.cohesion(boids);
  ///
  var centripetalForce = this.centripetal(boids);
  ////
  if ( walls ) var avoidWallsForce = this.avoidWalls();


  // Weight Forces
  var centripetalWeight = 1;
  var circularWeight = 1;
  //
  var alignWeight = 1; //1.2
```

```
  if ( mouseSeek ) var mouseWeight = 0.5; //.2
  var separateWeight = 1;
  var cohesionWeight = 1;
  if ( walls ) var avoidWallsWeight = 1.2;

  // Apply forces

  this.applyForce( alignForce, alignWeight );
  if ( mouseSeek ){
    this.applyForce( mouseForce, mouseWeight );
    this.applyForce( mouseForce2, mouseWeight );
  }
  this.applyForce( separateForce, separateWeight );
  this.applyForce( cohesionForce, cohesionWeight );
  //
  //this.applyForce(centrifugalForce, centrifugalWeight );
  this.applyForce(centripetalForce, centripetalWeight );

  //this.applyForce(circularMotionForce, circularWeight );
  //
  var distanceFromCenter =this.position.clone().distance(center);

  //
  if ( walls && avoidWallsForce ) this.applyForce( avoidWallsForce

}



//acceleration
applyForce( force, coefficient ) {
  if ( ! coefficient ) { var coefficient = 1; }
  force.divide({x:coefficient,y:coefficient});
  this.velocity.add(force);
  this.velocity.limitMagnitude( this.maxSpeed );
}
```

```
  //calculate
nextPosition() {

  // Update position
  let k1,k2,k3,k4,bun;
  var positiont= this.position.clone();
  var velocityt = this.velocity.clone();
  var position2= this.position.clone();
  var velocityt2 = this.velocity.clone();

  function u(p,v){
    var uv = p.add(v);
    return uv
  }

  const two = new Victor(2,2);
  const h = new Victor(0.5,0.5);
  const p0 = this.position

  var vecP = this.position.toArray();
  var vecV = this.velocity.toArray();
  var fLen=1,jLen=1;
  var upS =[,];
  for (let i = 0; i < vecP.length; i++) {
    for (let j = 0; j < vecV.length; j++){
      upS[i]= vecP[i]+vecV[i];
    }
  }




  this.velocity = this.velocity.add(this.acceleration);
```

```
this.positionI = this.position.add(this.velocity);//.multiply(t
//console.log(this.positionI);
////////////////////////////////////////////////////////////////
function rk4(x,v){
  var x1 = x;
  var v1 = v;
  //var h1 = h;

  var two = new Victor(2,2);
  var h = new Victor(0.5,0.5);
  var Six = new Victor(6,6);

  function f(x0,v0){
    x0.add(v0);
  }
  console.log("TEST"+f(x1.clone(),v1.clone()));
  var k1 = f(x1.clone(),v1.clone()).multiply(h);
  //console.log("k1:"+k1);

  var k2 = f(x1.clone().add(k1.clone().divide(two)),v1.clone().
  var k3 = f(x1.clone().add(k2.clone().divide(two)),v1.clone().
  var k4 = f(x1.clone().add(k3),v1.clone.add(h)).multiply(h);

  var bun = k1.clone().add(k2.clone().multiply(two)).add(k3.clo
  //var bun = k1.clone().add(k2.clone().multiply(two)).add(k3.c
  //var vN = x1.clone().add(bun.clone().multiply(h).divide(Six)
  var vN = bun
  console.log("V"+vN);
  return(vN);
}

//rk4(this.position,this.velocity);
//console.log("apakah ini:"+rk4(this.position,this.velocity));
//  this.velocity.add(this.h)
//  this.positionI = this.positionI.add(rk4(this.position,this.v
//  console.log("POS"+this.positionI);
```

```
//////////////////////////////////////////////////////////////
// k1 = (this.position.add(this.velocity)).multiply(this.h);
// this.k1.copy(k1);
// console.log("k1"+this.k1);
// // k2 = positiont.add(k1.divide(this.two)).add(velocityt2).a
// console.log("k2"+k2);
// k3 = positiont.add(k2.divide(this.two)).add(velocityt).add(t
// k4 = positiont.add(k3).add(velocityt).add(this.h).multiply(t
// this.bun = k1.add(k2.multiply(this.two)).add(k3.multiply(this
// this.positionI = this.position.add(this.bun);
// //console.log(this.positionI)
// console.log(this.position);
// console.log(this.velocity);
// this.k1 = this.position.add(this.velocity).multiply(this.h);
// this.k2 = this.position.add(this.k1.divide(this.two)).add(th
// this.k3 = this.position.add(this.k2.divide(this.two)).add(th
// this.k4 = this.position.add(this.k3).add(this.velocity).add(
//console.log(this.position);

////////////////////////////////////////////////
// this.k1 = this.k1.add(u(this.position,this.velocity)).multip
// console.log("k1"+this.k1);
// this.k2 = this.k2.add(u(this.position.add(this.k1.clone().di
// console.log("k2"+this.k2);
// // this.k3 = this.k3.add(u(this.position.add(this.k2.divide(
// console.log("k3"+this.k3);
// this.k4 = this.k4.add(u(this.position.add(this.k3.clone()),t
// console.log("k4"+this.k4);
// this.bun = this.k1.clone().add(this.k2.clone().multiply(this
// console.log("bundle="+this.bun)
// this.positionI = this.position.add(this.bun);
////////////////////////////////////////////////
// ////////////////////////////////////////////////
// this.k1 = u(this.position.clone(),this.velocity.clone()).mul
// console.log("k1"+this.k1);
// this.k2 = u(this.position.clone().add(this.k1.clone().divide
```

```
// this.k3 = u(this.position.clone().add(this.k2.clone().divide
// this.k4 = u(this.position.clone().add(this.k3.clone()),this.

// this.bun = this.k1.clone().add(this.k2.clone().divide(two)).
// console.log("test"+bun);
// this.positionI = this.position.add(this.bun);
// ////////////////////////////////////////////////////
// console.log("p"+this.positionI);
// // this.velocity.add(this.h);
// console.log(this.positionI);
//console.log(this.k1);
// // this.positionI = this.position.add()
// var test = new Victor();
// var u = new Victor(1,3);

// test = test.add(u).add(u).multiply(h);
// console.log(test);

//this.positionI = this.position + 1/6*(k1+(2*k2)+(2*k3)+k4);
//this.position.y = this.position.y.add(this.velocity.y);
// k1 = this.position.add(this.velocity).multiply(0.1);
// k2 = this.position.add(this.velocity.add(0.1/2)).multiply(k1
// k3 = this.position.add(this.velocity.add(0.1/2)).multiply(k2
// k4 = this.position.add(this.velocity.add(0.1)).multiply(k3/2
// this.position = this.position + 1/6*(k1+(2*k2)+(2*k3)+k4);

// function rungeKutta(){
// }
// Loop through behaviors to apply forces

this.flock();


// Collision detection if enabled
//console.log(this.position)
if ( collisions ) { this.detectCollision(); }
```

```
  // Check edges for walls or overruns
  this.edgeCheck();
  //this.kaabaCheck()

}


edgeCheck() {
  if (walls) {
    this.wallBounce();
    //this.kaabaBounce();
  } else {
    this.borderWrap();
  }
}


borderWrap() {
  if (this.position.x < 0) {
    this.position.x = document.body.clientWidth;
  } else if ( this.position.x > document.body.clientWidth ) {
    this.position.x = 0;
  }
  if (this.position.y < 0) {
    this.position.y = document.body.clientHeight;
  } else if ( this.position.y > document.body.clientHeight ) {
    this.position.y = 0;
  }
}




wallBounce() {
```

```
  if (this.position.x <= this.radius) {
    this.position.x = this.radius;
  } else if ( this.position.x >= document.body.clientWidth - this
    this.position.x = document.body.clientWidth - this.radius;
  }
  if (this.position.y <= this.radius) {
    this.position.y = this.radius;
  } else if ( this.position.y >= document.body.clientHeight - thi
    this.position.y = document.body.clientHeight - this.radius;
  }
  if ( this.distanceFromHorWall() <= this.radius  ) {
    this.velocity.invertY();
  }
  if ( this.distanceFromVertWall() <= this.radius  ) {
    this.velocity.invertX();
  }

}


distanceFromVertWall() {
  if (this.velocity.x > 0) {
    return document.body.clientWidth - ( this.position.x );
  } else {
    return this.position.x;
  }

}


distanceFromHorWall() {
  if (this.velocity.y > 0) {
    return document.body.clientHeight - ( this.position.y );
  } else {
    return this.position.y;
  }
```

```
}

///tambahan ikhsan
distanceFromHorKaaba(){
  if (this.velocity.y > 0){
    return wallskaaba.y - (this.position.y);
  } else {
    return this.position.y
  }
}
distanceFromVertKaaba(){
  if (this.velocity.x > 0) {
    return wallsKaaba.x - (this.position.x);
  } else {
    return this.position.x;
  }
}

wallKaabaBounce(){
  if (this.position.x <= this.radius) {
    this.position.x = this.radius;
  } else if ( this.position.x >= document.body.clientWidth - this
    this.position.x = document.body.clientWidth - this.radius;
  }
  if (this.position.y <= this.radius) {
    this.position.y = this.radius;
  } else if ( this.position.y >= document.body.clientHeight - thi
    this.position.y = document.body.clientHeight - this.radius;
  }
  if ( this.distanceFromHorWall() <= this.radius  ) {
    this.velocity.invertY();
  }
  if ( this.distanceFromVertWall() <= this.radius  ) {
    this.velocity.invertX();
  }
```

```
}



draw(){
  //c.beginPath();
  //transform the position
  var rr = transform({x:this.positionI.x, y:this.positionI.y}); /
  var theta = -1*Math.atan2(this.velocity.y,this.velocity.x); + M
  var vr = transform({x:this.velocity.x, y:this.velocity.y}); //
  var t,deltat;

  let r = this.radius;
  let sudut1=Math.PI*(0);
  let sudut2=Math.PI*((2/3)-(1/3));
  //let theta =-1*Math.atan2(this.velocity.x,this.velocity.y) + M
  c.save();
  c.beginPath();


  c.arc(rr.x, rr.y, this.radius, 0, Math.PI * 2);
  c.fillStyle = this.color;
  c.fill();
  c.closePath();
  c.restore();


  draw(canvas,rr.x,rr.y,vr.x,vr.y,1,sudut1,sudut2,this.color);
  t=+deltat
```

```
}

startLine(){
  c.strokeStyle = "#ff0000";
  c.beginPath();
  c.moveTo(center.x-50,center.y-50);
  c.lineTo(center.x,center.y);
  c.moveTo(center.x,center.y);
  c.lineTo(1020,820);
  c.moveTo(center.x-10,center.y);
  c.lineTo(988,820);
  c.moveTo(center.x,center.y-10);
  c.lineTo(1050,820);
  //c.fillStyle = "#ff0000";
  c.fill();
  c.stroke();
}

drawArrow(){
  c.strokeStyle = "#ff0000";
  c.beginPath();
  c.moveTo(this.position.x,this.position.y);
  c.lineTo(center.x,center.y);
  c.stroke();
}

update() {
  //this.transform();// tambahan ikhsan
  var sudut1=Math.PI*(0);
  var sudut2=Math.PI*((2/3)-(1/3));

  this.draw();
  ///draw all boundary except kaaba
  //this.startLine();
```

```
  //this.circularPath();
  ///
  this.nextPosition();
  c.beginPath();

  c.closePath();

  //this.draw(); //awalnya disini
}




detectCollision(){

  for (var i = 0; i < boids.length; i++) {
    if ( this === boids[i] ) { continue; }
    if ( getDistance( this.position.x, this.position.y, boids[i].
      this.resolveCollision( this, boids[i]);
    }
  }
}


rotate(velocity, angle) {
  return {
      x: velocity.x * Math.cos(angle) - velocity.y * Math.sin(ang
      y: velocity.x * Math.sin(angle) + velocity.y * Math.cos(ang
  };
}


 resolveCollision(boid, otherBoid) {

    var xVelocityDiff = boid.velocity.x - otherBoid.velocity.x;
    var yVelocityDiff = boid.velocity.y - otherBoid.velocity.y;
```

```
var xDist = otherBoid.position.x - boid.position.x;
var yDist = otherBoid.position.y - boid.position.y;

// Prevent accidental overlap of boids
if ( xVelocityDiff * xDist + yVelocityDiff * yDist >= 0 ) {

  // Grab angle between the two colliding boids
  var angle = -Math.atan2(otherBoid.position.y - boid.positio

  // Store mass in var for better readability in collision eq
  var m1 = boid.mass;
  var m2 = otherBoid.mass;

  // Velocity before equation
  var u1 = this.rotate(boid.velocity, angle);
  var u2 = this.rotate(otherBoid.velocity, angle);

  // Velocity after 1d collision equation
  var v1 = { x: u1.x * (m1 - m2) / (m1 + m2) + u2.x * 2 * m2
  var v2 = { x: u2.x * (m1 - m2) / (m1 + m2) + u1.x * 2 * m2

  // Final velocity after rotating axis back to original posi
  var vFinal1 = this.rotate(v1, -angle);
  var vFinal2 = this.rotate(v2, -angle);

  // Swap boid velocities for realistic bounce effect
  boid.velocity.x = vFinal1.x;
  boid.velocity.y = vFinal1.y;
  boid.velocity.limitMagnitude(boid.maxSpeed);

  otherBoid.velocity.x = vFinal2.x;
  otherBoid.velocity.y = vFinal2.y;
  otherBoid.velocity.limitMagnitude(otherBoid.maxSpeed);
}
```

```
    }

}
```

A.6 Kode Program script.js

```
/*---- Global Setup ----*/

// Set up canvas
const canvas = document.getElementById('boids');
const c = canvas.getContext('2d');

//Get Firefox
var browser=navigator.userAgent.toLowerCase();
if(browser.indexOf('firefox') > -1) {
  var firefox = true;
}

// Detect Mobile
var mobile = ( /Android|webOS|iPhone|iPad|iPod|BlackBerry|IEMobile|

// Set Size
var size = {
  width: window.innerWidth || document.body.clientWidth,
  height: window.innerHeight || document.body.clientHeight
}
console.log("ini bodi client"+document.body.clientWidth+"dan"+docum
canvas.width = size.width;
canvas.height = size.height;
console.log("besar canvas");
console.log(size.width);
console.log(size.height);

var center = new Victor( size.width / 2 ,size.height / 2 );
```

```
console.log("pusat"+center.x);


// Initialize Mouse
var mouse = {
  position: new Victor( innerWidth / 2, innerHeight / 2 )
};

/*---- end Global Setup ----*/

/*---- Helper Functions ----*/

/**
 * Returns a random int between a min and a max
 *
 * @param  int | min | A minimum number
 * @param  int | max | A maximum number
 * @return int | The random number in the given range
 */
function getRandomInt(min, max) {
  return Math.floor(Math.random() * (max - min + 1)) + min;
}

/**
 * Returns the distance between two coordinates
 *
 * @param  int | x1 | Point 1's x coordinate
 * @param  int | y1 | Point 1's y coordinate
 * @param  int | x2 | Point 2's x coordinate
 * @param  int | y2 | Point 2's y coordinate
 * @return int | The distance between points 1 and 2
 */
function getDistance(x1, y1, x2, y2) {
  var xDist = x2 - x1;
  var yDist = y2 - y1;
  return Math.sqrt( Math.pow(xDist, 2) + Math.pow(yDist, 2) );
```

```
}

/**
 * Returns a random color from the colors array
 *
 * @param  array | colors | An array of color values
 * @return string | The random color value
 */
function randomColor(colors) {
  return colors[ Math.floor( Math.random() * colors.length) ];
}

/**
 * Get coefficients based on normal distribution
 *
 * @param  int | mean | The mean value of the data set
 * @param  int | stdev | The standard deviation for the data set
 * @return int | A number from the data set
 */
function gaussian(mean, stdev) {
    var y2;
    var use_last = false;
    return function() {
        var y1;
        if(use_last) {
           y1 = y2;
           use_last = false;
        }
        else {
            var x1, x2, w;
            do {
                 x1 = 2.0 * Math.random() - 1.0;
                 x2 = 2.0 * Math.random() - 1.0;
                 w  = x1 * x1 + x2 * x2;
            } while( w >= 1.0);
            w = Math.sqrt((-2.0 * Math.log(w))/w);
```

```
            y1 = x1 * w;
            y2 = x2 * w;
            use_last = true;
        }

        var retval = mean + stdev * y1;
        if(retval > 0)
            return retval;
        return -retval;
        console.log("tes retval"+retval);
    }
}

var getCoefficient = gaussian(50, 9); //(50,9)
var getQuicknessCoefficient = gaussian(75,7.5); //(75,7.5)

/**
 * Add Limit Magnitude function to Victor objects
 *
 * @param  int | max | The limit magnitude for the vector
 */
Victor.prototype.limitMagnitude = function (max) {

  if (this.length() > max) {
    this.normalize();
    this.multiply({x:max,y:max});
  }

};
/**
 * Convert angle to radian
 *
 * @param  int | sudut | angle that convert
 */

function toRadian (sudut) {
```

```
  return sudut * (Math.PI / 180);
}

/*--- end Helper Functions ----*/

/*---- Loop and Initializing ----*/

// Checkbox Options
var walls = true;
var mouseSeek = false;
var collisions = false;

/*---- How much Boids ----*///120
var allBoids = 100;//1000;
var numAllBoids = allBoids;

var minBoids = 60;//1000;
var numBoids = minBoids;

var agroBoids = 20;//1000;//20
var numAgBoids = agroBoids;

var blackBoids = 20;//20
var numBlBoids =blackBoids;

// Set possible radii  based on screen size
var radius;
if ( size.width / 288 > 5 ) {
  radius = 5;
} else if ( size.width / 288 < 3) {
  radius = 3;
} else {
  radius = size.width / 288;
}
var radiusCoefficients = [.5,.6,.7,.8,.9,1];
```

```
// Boid Attributes
var colors = [
  '#4286f4',
  '#7df442',
  '#41f4a0',
  '#f9f9f9',
  '#a341f4',
  '#f48341',
  '#f4e841',
  '#42ebf4'
];

var colorsBlack = [
  '#000000'
];

var colorsRed = [
  '#FF0000'
];

var colorsYellow = [
  ' #FFFF00'
];

var diversity = 8;

var quickness = 1;
var agroQuickness = 1.25;
var blackQuickness = 0.5;

var introversion = .5;
var racism = 1; // 0 awalnya coba 5
var speedIndex;
if ( size.width / 160 < 5 ) {
  speedIndex = 1.25;//5 DEFAULT x
} else if ( size.width / 180 > 8 ) {
```

```
  speedIndex = 2.25;//9
} else {
  speedIndex = size.width / 180;
}
var maxForceAggro = 0.4;

// Create Boids Array
var boids = [];

// Other
var trigger = true;
/**
 * Create Boids Array
 *
 */
function createBoids() {

  // Instantiate all Boids
  for ( i = 0; i < numBoids; i++ ) {

    // Generate introversion coefficient
    var introversionCoefficient = getCoefficient() / 100;
    var quicknessCoefficient = getQuicknessCoefficient() / 100;
    var racismCoefficient = getCoefficient() / 100;
    var radiusCoefficient = Math.floor(Math.random() * radiusCoeffi

    // Generate random coords MUST FROM 50 TO SIZE CANVAS OR WALL M
    // if ( distanceFromCenter > 300 && distanceFromCenter < 1500){
    //   var x = Math.ceil(Math.random()* ( getRandomInt(center.x+5
    //   var y = Math.ceil(Math.random()* ( getRandomInt(center.y+5
    // }
    var x = Math.ceil(Math.random()* ( getRandomInt(center.x+50,size
    var y = Math.ceil(Math.random()* ( getRandomInt(center.y+50,size
    // For subsequent boids, check for collisions and generate new
    if ( i !== 0 ) {
      for (var j = 0; j < boids.length; j++ ) {
```

```
      if ( getDistance(x, y, boids[j].x, boids[j].y) - ( radius +
        x = Math.ceil(Math.random()* ( getRandomInt(center.x+50,s
        y = Math.ceil(Math.random()* ( getRandomInt(center.y+50,s
        j = -1;
      }
    }
  }

  // Add new Boid to array
  boids.push( new Boid( {
    id: i,
    x: x,
    y: y,
    speedIndex: speedIndex,
    radius: radius,
    radiusCoefficient: radiusCoefficient,
    quickness: quickness,
    quicknessCoefficient: quicknessCoefficient,
    color: randomColor(colors),
    racism: racism,
    racismCoefficient: racismCoefficient,
    introversion: introversion,
    introversionCoefficient: introversionCoefficient
    //maxForce: maxForce
  } ) );


  // Add new black Boid to array
  // boids.push( new Boid( {
  //   id: i,
  //   x: x,
  //   y: y,
  //   speedIndex: speedIndex,
  //   radius: radius,
  //   radiusCoefficient: radiusCoefficient,
  //   quickness: quickness,
```

```
    //    quicknessCoefficient: quicknessCoefficient,
    //    color: colorsBlack,
    //    racism: racism,
    //    racismCoefficient: ,
    //    introversion: introversion,
    //    introversionCoefficient: introversionCoefficient
    // } ) );
  }

}

function agressiveBoids() {

  // Instantiate all Boids
  for ( i = 0; i < numAgBoids; i++ ) {

    // Generate introversion coefficient
    var introversionCoefficient = getCoefficient() / 100;
    var quicknessCoefficient = getQuicknessCoefficient() / 100;
    var racismCoefficient = getCoefficient() / 100;
    var radiusCoefficient = Math.floor(Math.random() * radiusCoeffi

    // Generate random coords MUST FROM 50 TO SIZE CANVAS OR WALL M
    // if ( distanceFromCenter > 300 && distanceFromCenter < 1500){
    //   var x = Math.ceil(Math.random()* ( getRandomInt(center.x+5
    //   var y = Math.ceil(Math.random()* ( getRandomInt(center.y+5
    // }
    var x = Math.ceil(Math.random()* ( getRandomInt(center.x+50,size
    var y = Math.ceil(Math.random()* ( getRandomInt(center.y+50,size
    // For subsequent boids, check for collisions and generate new
    if ( i !== 0 ) {
      for (var j = 0; j < boids.length; j++ ) {
        if ( getDistance(x, y, boids[j].x, boids[j].y) - ( radius +
          x = Math.ceil(Math.random()* ( getRandomInt(center.x+50,s
          y = Math.ceil(Math.random()* ( getRandomInt(center.y+50,s
          j = -1;
```

```
        }
      }
    }

    // Add new Boid to array
    boids.push( new Boid( {
      id: i,
      x: x,
      y: y,
      speedIndex: speedIndex,
      radius: radius,
      radiusCoefficient: radiusCoefficient,
      quickness: agroQuickness,
      quicknessCoefficient: quicknessCoefficient,
      color: colorsRed,
      racism: racism,
      racismCoefficient: racismCoefficient,
      introversion: introversion,
      introversionCoefficient: introversionCoefficient,
      //maxForce: maxForceAggro
    } ) );

  }

}

function slowBoids() {

  // Instantiate all Boids
  for ( i = 0; i < numBlBoids; i++ ) {

    // Generate introversion coefficient
    var introversionCoefficient = getCoefficient() / 100;
    var quicknessCoefficient = getQuicknessCoefficient() / 100;
    var racismCoefficient = getCoefficient() / 100;
    var radiusCoefficient = Math.floor(Math.random() * radiusCoeffi
```

```
// Generate random coords MUST FROM 50 TO SIZE CANVAS OR WALL M
// if ( distanceFromCenter > 300 && distanceFromCenter < 1500){
//   var x = Math.ceil(Math.random()* ( getRandomInt(center.x+5
//   var y = Math.ceil(Math.random()* ( getRandomInt(center.y+5
// }
var x = Math.ceil(Math.random()* ( getRandomInt(center.x+50,size
var y = Math.ceil(Math.random()* ( getRandomInt(center.y+50,size
// For subsequent boids, check for collisions and generate new
if ( i !== 0 ) {
  for (var j = 0; j < boids.length; j++ ) {
    if ( getDistance(x, y, boids[j].x, boids[j].y) - ( radius +
      x = Math.ceil(Math.random()* ( getRandomInt(center.x+50,s
      y = Math.ceil(Math.random()* ( getRandomInt(center.y+50,s
      j = -1;
    }
  }
}

// Add new Boid to array
boids.push( new Boid( {
  id: i,
  x: x,
  y: y,
  speedIndex: speedIndex,
  radius: radius,
  radiusCoefficient: radiusCoefficient,
  quickness: blackQuickness,
  quicknessCoefficient: quicknessCoefficient,
  color: colorsBlack,
  racism: racism,
  racismCoefficient: racismCoefficient,
  introversion: introversion,
  introversionCoefficient: introversionCoefficient
  //maxForce: maxForce
} ) );
```

```
  }

}



/**
 * function draw walls kaaba
 *
 */

////////////////////////////
 //bikin sekat area luar
 //outer walls
  //1 banding 3 scalar
  function Vect2() {
    this.x = 0;
    this.y = 0;
  }

  var sx = center.x;
  var sy = center.y;
  var s11 = 219; // 73
  var s12 = 216; // 72
  var s21 = 219; // 73
  var s22 = 225; // 75
  var s31 = 219; // 73
  var s32 = 216; // 72
  var s41 = 225; // 75
  var s42 = 219; // 73
  var wA = new Vect3(-s11,s12,0);
  var wB = new Vect3(s21, s22,0);
  var wC = new Vect3(s31,-s32,0);
  var wD = new Vect3(-s41, -s42,0);
  //console.log(rA);
```

```
// Define wall
//var surf = new Sides2(); //barrier
var sides = [];
surf = new Sides2(wA,wB);
sides.push(surf);
surf = new Sides2(wB,wC);
sides.push(surf);
surf = new Sides2(wC,wD);
sides.push(surf);
surf = new Sides2(wD,wA);
sides.push(surf);


 //kaaba walls
var s = 50
var sx = center.x;
var sy = center.y;
var rA = new Vect3(sx,sy,0);
var rB = new Vect3(sx-s,sy,0);
var rC = new Vect3(sx-s,sy-s,0);
var rD = new Vect3(sx,sy-s  ,0);


// var s = 100
// var s2 = 150
// var sx = center.x;
// var sy = center.y;
// var rA = new Vect3(sx-s,sy-s2,0);
// var rB = new Vect3(sx+s,sy+s2  ,0);
// var rC = new Vect3(sx+s,sy-s2,0);
// var rD = new Vect3(sx-s,sy+s2  ,0);
```

```
  // Define kaaba //change from sides to walls
  var surf = new Sides2();
  var wallsKaaba = [];
  surf = new Sides2(rA,rB);
  wallsKaaba.push(surf);
  surf = new Sides2(rB,rC);
  wallsKaaba.push(surf);
  surf = new Sides2(rC,rD);
  wallsKaaba.push(surf);
  surf = new Sides2(rD,rA);
  wallsKaaba.push(surf);
  surf = new Sides2(rA,rC);
  var vecKaaba = Victor.fromArray(wallsKaaba);
  console.log("tes"+vecKaaba);

  //make center var surf


  console.log("x="+surf.center().x);
  console.log("y="+surf.center().y);

  console.log("koordinat tengah"+surf.center());
  console.log(surf.strval());

  console.log("arrayof kaaba"+wallsKaaba);
  console.log("tes"+wallsKaaba[0].p[0].x);
  console.log();
  // console.log("tes"+walls[0].p[0].x.length);
  // console.log(surf.p[0].x);



  //console.log(sides);
function drawWalls(id,surfs,color){
  var cx = document.getElementById("boids").getContext("2d");
    cx.strokeStyle = color;
```

```
    var N = surfs.length;
    //console.log(N);
    for(var i = 0; i < N; i++) {
      var M = surfs[i].p.length;
      //console.log(M);
      cx.beginPath();
      for(var j = 0; j < M; j++) {
        var s = surfs[i];
        var rr = transform({x: s.p[j].x, y: s.p[j].y});
        if(j == 0) {
          cx.moveTo(rr.x, rr.y);
        } else {
          cx.lineTo(rr.x, rr.y);
        }
      }
      cx.stroke();
    }
    c.strokeStyle = "#ff0000";
    c.beginPath();
    // c.moveTo(center.x-50,center.y-50);
    // c.lineTo(center.x,center.y);
    c.moveTo(center.x,center.y);
    c.lineTo(1020,820);
    c.moveTo(center.x-10,center.y);
    c.lineTo(988,820);
    c.moveTo(center.x,center.y-10);
    c.lineTo(1050,820);
    //c.fillStyle = "#ff0000";
    c.fill();
    c.stroke();

}
// Define world coordinate
  var xmin = 0;//-1*(size.width/2);//0//-1*(size.width/2);//360
  var ymin = 0;//-1*(size.width/2);//0//-1*(size.height/2);//
  var xmax = size.width//size.width//size.width/2;//
```

```
  var ymax = size.height//size.height//size.height/2;//

  // Define canvas size
  // var canvasWidth = 720;
  // var canvasHeight = 720;

  // Define canvas coordinate
  var XMIN = 0;
  var YMIN = 0;//
  var XMAX = size.width;//
  var YMAX = size.height;

function transform(r) {
    var X = (r.x - xmin) / (xmax - xmin) * (XMAX - XMIN);
    X += XMIN;
    var Y = (r.y - ymin) / (ymax - ymin) * (YMAX - YMIN);
    Y += YMIN;
    return {x: X, y: Y};
  }

////////////////////////////////////////////////////

/**
 * Setup and call animation function
 *
 */
  var t=0,dt=1;
  //var arr=[]; //y
  //var arr1=[];//x

  var arrt=[];
  var arr =[];

  var arry=[];
  var arry1=[];
  var arry2=[];
```

```
  var arry3=[];
  var arry4=[];
  var arry5=[];

  var Vb1,Vb2,Vb3,Vb4,Vb5;

  var aB1,aB2;

  var verR1;

function animate() {
  requestAnimationFrame(animate);

  // Calc elapsed time since last loop
  now = Date.now();
  elapsed = now - then;
  //console.log("elapsed"+elapsed);

  // FPS Reporting
  fpsReport++;
  if (fpsReport > 60) {
    fpsNum.innerHTML = Math.floor(1000/elapsed);
    fpsReport = 0;
  }

  // If enough time has elapsed, draw the next frame
  if (elapsed > fpsInterval) {
      // Get ready for next frame by setting then=now, but also adj
      // specified fpsInterval not being a multiple of RAF's interv
      then = now - (elapsed % fpsInterval);
      // Drawing Code
      t+=dt
    function simulate() {
      c.clearRect(0, 0, canvas.width, canvas.height);
      var arr=[]; //y
      var arr1=[];//x
```

```
  var arr2=[];
  var arr3=[];
  var verR;
  // Update all boids
function test(){
  if(trigger== true){
    for (var i = 0; i < boids.length; i++ ) {
    //arr.push(boids[i].velocity.length());
    boids[i].update();
    arr.push(boids[i].velocity.length());
    arr2.push(boids[i].position.x);
    arr3.push(boids[i].position.y);
    drawWalls("walls", wallsKaaba, "#f00");//
    //drawWalls("walls", sides, "#f00");//
      }
}
  console.log(arr2);
  console.log(arr3);
  //console.log(n[1].x);



  //arr1.push(t);
  // t+=dt
  console.log(arr1);
  }
  console.log(arr1);
  test();
  console.log(arr[1]);
  Vb1 = arr[99];
  Vb2 = arr[98];
  Vb3 = arr[97];
  Vb4 = arr[96];
  Vb5 = arr[95];

  aB1 =arr2;
```

```
aB2 =arr3;
console.log(aB1);



console.log(Vb1);
//arr1.push(t);
//console.log(arr);
// console.log(
// arr.reduce((a, b) => a + b, 0)
// )
verR = arr.reduce((a, b) => a + b, 0);
//verR1 = verR/boids[i].length();
verR1 = verR/numAllBoids;
const verR2 = verR1
//console.log(verR1);
//console.log(arr1);
//console.log(verR2);
// for(let i = 0; i < 2; i++  ){

//    arrt.push(verR2);
// }



// //arrt.push(verR2);
// console.log(arrt);
//
const N = 10, t_0 = 0, t_1 = 1, y_0 = 0
const h = (t_1 - t_0) / N  //time step size

//var ts = Array.from(Array(N+1), (_, k) => k * h + t_0)
//var ys = Array(N+1).fill(0)  //empty array for the results
var ts = [];
//var ys = Array.from(verR1+1).fill(0)
var ys =[];
ys[0] = y_0  //initial conditions
```

```
function f(v,ts){
  var u,v,ts;
  u=v*t;
  return u;
}
var resolution = 100;
var y=[],s=[],yts;
for (let i = 0; i < resolution; i++) {
   ts[i]=i;
   ys[i]=verR1;
   const k1 = f(ts[i], ys[i])

   const s1 = ys[i] + k1 * h/2
   const k2 = f(ts[i] + h/2, s1)

   const s2 = ys[i] + k2 * h/2
   const k3 = f(ts[i] + h/2, s2)

   const s3 = ys[i] + k3 * h
   const k4 = f(ts[i] + h, s3) // f(t + h, y_n + k3*h)
   ys[i + 1] = ys[i] + (k1/6 + k2/3 + k3/3 + k4/6) * h
   //yts.copy(ys[i+1]);
   //y.push(yts.clone());
   y.push(ys[i+1])
   //console.log(ys[i+1]);
   s.push(ts[i]);

}
console.log(y);
console.log(s);
//
//console.log(arrt);
// var tes= [1,2,3,4,5];
// var testwo= [6,7,4,3,1];
```

```
//openGraph(s,y);
}
simulate();



function openGraph(tes,testwo1,testwo2,testwo3,testwo4,testwo5)
  var tes;
  var testwo;
  var testwo1;
  var testwo2;
  var testwo3;
  var testwo4;
  var testwo5;
  var graph1={
  x: tes,
  y: testwo1,
  mode: 'lines+markers',
  type: 'scatter',
  name: 'dots1'
  };
  var graph2={
  x: tes,
  y: testwo2,
  mode: 'lines+markers',
  type: 'scatter',
  name: 'dots2'
  };
  var graph3={
  x: tes,
  y: testwo3,
  mode: 'lines+markers',
  type: 'scatter',
  name: 'dots3'
  };
  var graph4={
```

```
x: tes,
y: testwo4,
mode: 'lines+markers',
type: 'scatter',
name: 'dots4'
};
var graph5={
x: tes,
y: testwo5,
mode: 'lines+markers',
type: 'scatter',
name: 'dots5'
};
var data1 =[graph1,graph2,graph3,graph4,graph5];
var gambar1= {
title: {
  text:'Grafik 1',
  font: {
  family: 'Courier New, monospace',
  size: 24
  },
  xref: 'paper',
  x: 0.05,
},
xaxis: {
 title: {
 text: 'T(waktu)',
 font: {
 family: 'Courier New, monospace',
 size: 18,
 color: '#7f7f7f'
 }
 },
},
yaxis: {
 title: {
```

```
    text: 'V(kecepatan)',
    font: {
    family: 'Courier New, monospace',
    size: 18,
    color: '#7f7f7f'
    }
    }
  }
  };
;
  // Opens a new window
  var myWindow = window.open("", "myWindow", "width=1080,height
  myWindow.document.createElement("div");
  const graph = document.createElement("div");
  graph.setAttribute('id', 'area3');
  myWindow.document.body.appendChild(graph);
  Plotly.newPlot(graph, data1, gambar1,"")
 }
function openGraph1(tes,testwo){
  var tes;
  var testwo;

  var graph1={
  x: tes,
  y: testwo,
  mode: 'lines+markers',
  type: 'scatter',
  name: 'dots1'
  };

  var data1 =[graph1];
  var gambar1= {
  title: {
    text:'Grafik 1',
    font: {
    family: 'Courier New, monospace',
```

```
        size: 24
        },
        xref: 'paper',
        x: 0.05,
      },
      xaxis: {
       title: {
       text: 'T(waktu)',
       font: {
       family: 'Courier New, monospace',
       size: 18,
       color: '#7f7f7f'
       }
       },
      },
      yaxis: {
       title: {
       text: 'V(kecepatan)',
       font: {
       family: 'Courier New, monospace',
       size: 18,
       color: '#7f7f7f'
       }
       }
      }
      };
     ;
      // Opens a new window
      var myWindow = window.open("", "myWindow", "width=1080,height
      myWindow.document.createElement("div");
      const graph = document.createElement("div");
      graph.setAttribute('id', 'area3');
      myWindow.document.body.appendChild(graph);
      Plotly.newPlot(graph, data1, gambar1,"")
     }
}
```

```
    // console.log(y);
    // console.log(s);
arrt.push(t);
//arry.push(verR1);
arry1.push(Vb1);
arry2.push(Vb2);
arry3.push(Vb3);
arry4.push(Vb4);
arry5.push(Vb5);
// console.log(arrt);
 // console.log(arry);
 // console.log(arry1);
 // console.log(arry2);
 // console.log(arry3);
//openGraph(arrt,arry1,arry2,arry3,arry4,arry5);
//openGraph1(aB1,aB2);


}
// arrt.push(t);
// arry.push(verR1);
// console.log(arrt);
// console.log(verR1);



function clear(){
    var ca = arguments[0];
    var ctx = ca.getContext("2d");
    ctx.beginPath();
    ctx.clearRect(0,0,size.width,size.height);
    ctx.closePath();
}


// Setup animation
```

```
var stop = false;
var frameCount = 0;
var fps, fpsInterval, startTime, now, then, elapsed;
var fpsNum = document.getElementById('fps-number');
var fpsReport = 58;
/**
 * Start Animation of Boids
 *
 */
function startAnimating() {
  if(fps == null) { var fps = 60; }
  fpsInterval = 1000 / fps;
  then = Date.now();
  startTime = then;
  animate();
}
/**
 * Stop Animation of Boids
 *
 */

 function stopAnimating() {
  if(fps != null) { var fps = null; }
  fpsInterval = 1000 / fps;
  then = Date.now();
  startTime = then;
  //animate();
}



/*---- end Loop and Initializing ----*/

/*---- Event Listeners ----*/

/**
```

```
 * Update mouse positions on mousemove
 *
 */
addEventListener('mousemove', function(event){
  mouse.position.x = event.clientX;
  mouse.position.y = event.clientY;
});

/**
 * Update boundary sizes on window resize
 *
 */
addEventListener('resize', function(){
  size.width = innerWidth;
  size.height = innerHeight;
  canvas.width = innerWidth;
  canvas.height = innerHeight;
  center.x = size.width/ 2;
  center.y = size.height / 2;
  if ( innerWidth >= 1000 && ! mobile ) {
    document.getElementById('mobile-boids-controls').style.display =
  } else {
    document.getElementById('mobile-boids-controls').style.display =
  }
});

/*---- end Event Listeners ----*/

/*---- Inputs ----*/

var buttonStart = document.getElementById('Start');
buttonStart.onclick = function(){
//Initalize program
var id = event.target.id;
createBoids();
agressiveBoids();
```

```
slowBoids();
startAnimating(60);
}

// var buttonStart2 = document.getElementById('Start2');
// buttonStart.onclick = function(){
// //Initalize program
// var id = event.target.id;
// createBoids();
// startAnimating(60);
// }


var buttonStop = document.getElementById('Stop');
buttonStop.onclick = function(){
 c.clearRect(0, 0, canvas.width, canvas.height);
 stopAnimating(0);
}

// Hide Elements on Mobile
document.getElementById('collisions-mobile').style.display = 'none'
document.getElementById('mouse-seek-mobile').style.display = 'none'

// Mobile Closers
var mobileClosers = document.getElementsByClassName('boids-control-
for (var i = 0; i < mobileClosers.length; i++) {
  mobileClosers[i].onclick = function() {
    this.parentNode.classList.toggle('show');
    document.getElementById('mobile-boids-controls').style.display =
  }
}

// Walls
var wallsInput = document.getElementById('walls');
wallsInput.checked = true;
wallsInput.onclick = function() {
```

```
  if ( !this.checked ) {
    this.checked = false;
    wallsMobile.dataset.checked = false;
    wallsMobile.classList.toggle('boids-checkbox-on');
    walls = false;
  } else {
    this.checked = true;
    wallsMobile.dataset.checked = true;
    wallsMobile.classList.toggle('boids-checkbox-on');
    walls = true;
  }
}
var wallsMobile = document.getElementById('walls-mobile');
wallsMobile.dataset.checked = true;
wallsMobile.onclick = function() {
  if ( this.dataset.checked == 'false') {
    this.dataset.checked = true;
    wallsInput.checked = true;
    this.classList.toggle('boids-checkbox-on');
    walls = true;
  } else {
    this.dataset.checked = false;
    wallsInput.checked = false;
    this.classList.toggle('boids-checkbox-on');
    walls = false;
  }
}

// Collision Detection
var collisionDetectionInput = document.getElementById('collision-de
collisionDetectionInput.checked = false;
collisionDetectionInput.onclick = function() {
  if ( !this.checked ) {
    this.checked = false;
    collisionDetectionMobile.dataset.checked = false;
    collisionDetectionMobile.classList.toggle('boids-checkbox-on');
```

```
    collisions = false;
  } else {
    this.checked = true;
    collisionDetectionMobile.dataset.checked = true;
    collisionDetectionMobile.classList.toggle('boids-checkbox-on');
    collisions = true;
  }
}
var collisionDetectionMobile = document.getElementById('collisions-
collisionDetectionMobile.dataset.checked = false;
collisionDetectionMobile.onclick = function() {
  if ( this.dataset.checked == 'false') {
    this.dataset.checked = true;
    collisionDetectionInput.checked = true;
    this.classList.toggle('boids-checkbox-on');
    collisions = true;
  } else {
    this.dataset.checked = false;
    collisionDetectionInput.checked = false;
    this.classList.toggle('boids-checkbox-on');
    collisions = false;
  }
}

// Mouse Seek
var mouseSeekInput = document.getElementById('mouse-seek');
mouseSeekInput.checked = false;
mouseSeekInput.onclick = function() {
  if ( !this.checked ) {
    this.checked = false;
    mouseSeekMobile.dataset.checked = false;
    mouseSeekMobile.classList.toggle('boids-checkbox-on');
    mouseSeek = false;
  } else {
    this.checked = true;
    mouseSeekMobile.dataset.checked = true;
```

```
    mouseSeekMobile.classList.toggle('boids-checkbox-on');
    mouseSeek = true;
  }
}
var mouseSeekMobile = document.getElementById('mouse-seek-mobile');
mouseSeekMobile.dataset.checked = false;
mouseSeekMobile.onclick = function() {
  if ( this.dataset.checked == 'false') {
    this.dataset.checked = true;
    mouseSeekInput.checked = true;
    this.classList.toggle('boids-checkbox-on');
    mouseSeek = true;
  } else {
    this.dataset.checked = false;
    mouseSeekInput.checked = false;
    this.classList.toggle('boids-checkbox-on');
    mouseSeek = false;
  }
}

// Introversion
var introversionControlContainer = document.getElementById('introver
var introversionInput = document.getElementById('introversion');
introversionInput.onchange = function() {
  introversion = this.value / 10;
  updateIntroversion(introversion);
}
var introversionMobile = document.getElementById('introversion-mobi
introversionMobile.onclick = function() {
  document.getElementById('mobile-boids-controls').style.display =
  introversionControlContainer.classList.toggle('show');
}
function updateIntroversion(value) {
  for (var i=0; i<boids.length; i++) {
    boids[i].introversion = value * boids[i].introversionCoefficien
  }
```

```
}

// Speed
var speedControlContainer = document.getElementById('speed-control-
var speedInput = document.getElementById('speed');
speedInput.onchange = function() {
  quickness = this.value / 10 + .5;
  updateQuickness(quickness);
}
var speedMobile = document.getElementById('speed-mobile');
speedMobile.onclick = function() {
  document.getElementById('mobile-boids-controls').style.display =
  speedControlContainer.classList.toggle('show');
}
function updateQuickness(value) {
  for (var i=0; i<boids.length; i++) {
    boids[i].quickness = value * boids[i].quicknessCoefficient;
    boids[i].maxSpeed = speedIndex * boids[i].quickness;
  }
}

// //Red Speed
// var speedControlContainer = document.getElementById('Rspeed-cont
// var speedInput = document.getElementById('Rspeed');
// speedInput.onchange = function() {
//   quickness = this.value / 10 + .5;
//   updateQuickness(quickness);
// }
// var speedMobile = document.getElementById('Rspeed-mobile');
// speedMobile.onclick = function() {
//   document.getElementById('mobile-boids-controls').style.display
//   speedControlContainer.classList.toggle('show');
// }
// function updateQuickness(value) {
//   for (var i=0; i<boids.length; i++) {
//     boids[i].quickness = value * boids[i].quicknessCoefficient;
```

```
//      boids[i].maxSpeed = speedIndex * boids[i].quickness;
//    }
// }

// //Black Speed
// var speedControlContainer = document.getElementById('speed-contro
// var speedInput = document.getElementById('speed');
// speedInput.onchange = function() {
//   quickness = this.value / 10 + .5;
//   updateQuickness(quickness);
// }
// var speedMobile = document.getElementById('speed-mobile');
// speedMobile.onclick = function() {
//   document.getElementById('mobile-boids-controls').style.display
//   speedControlContainer.classList.toggle('show');
// }
// function updateQuickness(value) {
//   for (var i=0; i<boids.length; i++) {
//     boids[i].quickness = value * boids[i].quicknessCoefficient;
//     boids[i].maxSpeed = speedIndex * boids[i].quickness;
//   }
// }



// Racisms
var racismControlContainer = document.getElementById('racism-control
var racismInput = document.getElementById('racism');
racismInput.onchange = function() {
  racism = this.value / 5;
  updateRacism(racism);
}
var racismMobile = document.getElementById('racism-mobile');
racismMobile.onclick = function() {
  document.getElementById('mobile-boids-controls').style.display =
  racismControlContainer.classList.toggle('show');
```

```
}
function updateRacism(value) {
  for (var i=0; i<boids.length; i++) {
    boids[i].racism = value * boids[i].racismCoefficient;
  }
}

// Diversity
var diversityControlContainer = document.getElementById('diversity-
var diversityInput = document.getElementById('diversity');
diversityInput.onchange = function() {
  diversity = this.value;
  updateDiversity(diversity);
}
var diversityMobile = document.getElementById('diversity-mobile');
diversityMobile.onclick = function() {
  document.getElementById('mobile-boids-controls').style.display =
  diversityControlContainer.classList.toggle('show');
}
function updateDiversity(value) {
  for (var i=0; i<boids.length; i++) {
    boids[i].color = colors[ i % value ];
  }
}
```

Lampiran B

Riwayat Hidup

make as simple as possible