TESIS – TE142599

NAVIGASI MULTIAGEN MENGGUNAKAN RECIPROCAL VELOCITY OBSTACLES BERELASI LEADER-FOLLOWER DENGAN TINGKAT PRIORITAS

MOCH FACHRI
NRP 07111650050011

DOSEN PEMBIMBING
Mochammad Hariadi, S.T.,M.Sc.,Ph.D.
Dr. Supeno Mardi Susiki Nugroho, ST., MT.

PROGRAM MAGISTER
BIDANG KEAHLIAN JARINGAN CERDAS MULTIMEDIA
DEPARTEMEN TEKNIK ELEKTRO
FAKULTAS TEKNOLOGI ELEKTRO
INSTITUT TEKNOLOGI SEPULUH NOPEMBER
SURABAYA
2018

TESIS – TE142599

NAVIGASI MULTIAGEN MENGGUNAKAN RECIPROCAL VELOCITY OBSTACLES BERELASI LEADER-FOLLOWER DENGAN TINGKAT PRIORITAS

MOCH FACHRI
07111650050011

DOSEN PEMBIMBING
Mochammad Hariadi, S.T.,M.Sc.,Ph.D.
Dr. Supeno Mardi Susiki Nugroho, ST., MT.

PROGRAM MAGISTER
BIDANG KEAHLIAN JARINGAN CERDAS MULTIMEDIA
DEPARTEMEN TEKNIK ELEKTRO
FAKULTAS TEKNOLOGI ELEKTRO
INSTITUT TEKNOLOGI SEPULUH NOPEMBER
SURABAYA
2018

LEMBAR PENGESAHAN

Tesis disusun untuk memenuhi salah satu syarat memperoleh gelar
Magister Teknik (M.T)
di
Institut Teknologi Sepuluh Nopember

oleh:

Moch Fachri
NRP. 07111650050011

Tanggal Ujian : 19 Desember 2017
Periode Wisuda : Maret 2018

Disetujui oleh:

1. **Mochammad Hariadi, S.T., M.Sc., Ph.D.** **(Pembimbing I)**
   NIP: 196912091997031003

2. **Dr. Supeno Mardi Susiki Nugroho, ST., MT.** **(Pembimbing II)**
   NIP: 19700313199512100l

3. **Dr. Surya Sumpeno , ST., M.Sc.** **(Penguji)**
   NIP: 196906131997021003

4. **Dr. Eko Mulyanto Yuniarno, S.T., M.T.** **(Penguji)**
   NIP: 196806011995121009

Dekan Fakultas Teknologi Elektro

Dr. Tri Arief Sardjono, S.T., M.T.
NIP. 197002121995121001

Halaman ini sengaja dikosongkan

PERNYATAAN KEASLIAN TESIS

Dengan ini saya menyatakan bahwa isi keseluruhan Tesis saya dengan judul **NAVIGASI MULTIAGEN MENGGUNAKAN RECIPROCAL VELOCITY OBSTACLES BERELASI LEADER-FOLLOWER DENGAN TINGKAT PRIORITAS** adalah benar-benar hasil karya intelektual mandiri, diselesaikan tanpa menggunakan bahan-bahan yang tidak diijinkan dan bukan merupakan karya pihak lain yang saya akui sebagai karya sendiri.

Semua referensi yang dikutip maupun dirujuk telah ditulis secara lengkap pada daftar pustaka. Apabila ternyata pernyataan ini tidak benar, saya bersedia menerima sanksi sesuai peraturan yang berlaku.

Surabaya, Januari 2018

Moch Fachri
07111650050011

Halaman ini sengaja dikosongkan

NAVIGASI MULTIAGEN MENGGUNAKAN RECIPROCAL VELOCITY OBSTACLES BERELASI LEADER-FOLLOWER DENGAN TINGKAT PRIORITAS

Nama Mahasiswa	:	Moch Fachri
NRP	:	07111650050011
Pembimbing	:	1. Mochamad Hariadi, ST., M.Sc., Ph.D.
		2. Dr. Supeno Mardi Susiki Nugroho, ST., MT.

ABSTRAK

Pada simulasi kerumunan, kepadatan agen dalam kerumunan menimbulkan masalah dalam pergerakan agen. Hal ini mengharuskan adanya metode navigasi khusus yang dapat memastikan manuver penghindaran yang tidak mengakibatkan kekacauan massal tanpa adanya komunikasi eksplisit antar agen. Untuk simulasi kerumunan dengan relasi *leader-follower*, agen *follower* diharuskan mampu mengikuti *leader* kelompoknya. Agen *follower* juga perlu memberikan agen *leader*-nya kebebasan bernavigasi, yang berarti gerakan agen *follower* sebisa mungkin tidak mengganggu *leader*-nya. Hal ini dikarenakan navigasi agen *follower* yang bergantung terhadap *leader*-nya. *Reciprocal Velocity Obstacle* (RVO) sebagai salah satu metode navigasi multiagen tidak memerlukan komunikasi eksplisit dengan agen lain dalam penghindaran, dan bebas dari gerak osilasi yang terjadi pada *Velocity Obstacle* biasa. Diharapkan dengan memakai RVO sebagai navigasi multiagen yang memiliki variasi tingkat prioritas, didapatkan simulasi kerumunan dengan relasi *leader-follower* yang bebas dari tabrakan, osilasi, dan kehilangan arah.

Kata Kunci : RVO (*Reciprocal Velocity Obstacle*), Simulasi Kerumunan, Navigasi Multi-agen

Halaman ini sengaja dikosongkan

MULTI-AGENTS NAVIGATION USING RECIPROCAL VELOCITY OBSTACLES AND LEADER-FOLLOWER RELATIONSHIP WITH LEVEL OF PRIORITY

By	:	Moch Fachri
Student Identity Number	:	07111650050011
Supervisor(s)	:	1. Mochamad Hariadi, ST., M.Sc., Ph.D.
		2. Dr. Supeno Mardi Susiki Nugroho, ST., MT.

ABSTRACT

In crowd simulation, agents density on crowds cause problem in agents movement. This oblige a particular navigation method that can ensure the avoidance maneuver that does not result in mass chaos without any explicit communication between agents. To simulate the crowd with leader-follower relationships, agents follower is expected to follow the group leader. followers agent need to give their leader freedom of navigation, which is mean that followers movement shall not make any disturbance to their leader. This is because the navigation of agent follower depends on the leader. Reciprocal Velocity Obstacle(RVO) as one method of multiagent navigation does not require explicit communication with other agents in avoidance, and free from oscillation movement that occur on regular Velocity Obstacle. RVO usage with variant inlevel of priority as multi-agents navigation is expected to bring out crowd simulation with leader-follower relationship that is free of collisions, oscillation, and loss direction.

Key words : RVO (*Reciprocal Velocity Obstacle*), Crowd Simulation, Multi-agents Navigation

Halaman ini sengaja dikosongkan

KATA PENGANTAR

Puji dan syukur kehadirat Allah Subhanahu wa ta'ala atas segala limpahan berkah, rahmat, serta hidayah-Nya, penulis dapat menyelesaikan penelitian dengan judul **Navigasi Multi-agen Menggunakan *Reciprocal Velocity Obstacles* Berelasi *Leader-Follower* Dengan Tingkat Prioritas**. Tanpa kehendak-Nya tidak mungkin penulis mampu mewujudkan penelitian ini.

Penelitian ini disusun dalam rangka pemenuhan bidang riset di Departemen Teknik Elektro ITS, bidang studi Jaringan Cerdas Multimedia, konsentrasi Teknologi Permainan, serta digunakan sebagai persyaratan menyelesaikan pendidikan S2. Selain itu penelitian ini juga ditujukan untuk memajukan peradaban manusia di bidang teknologi komputer terutama bidang teknologi permainan dan simulasi komputer. Penelitian ini dapat terselesaikan tidak lepas dari bantuan berbagai pihak. Oleh karena itu, penulis mengucapkan terima kasih kepada:

1. Keluarga, Ibu, Bapak dan Saudara tercinta yang telah memberikan dorongan spiritual dan material dalam penyelesaian buku penelitian ini.
2. Bapak Dr. Tri Arief Sardjono, ST., MT. selaku Dekan Fakultas Teknologi Elektro, Institut Teknologi Sepuluh Nopember.
3. Bapak Dr. Ardyono Priyadi, ST., M.Eng. selaku Ketua Departemen Teknik Elektro, Fakultas Teknologi Elektro, Institut Teknologi Sepuluh Nopember.
4. Secara khusus penulis mengucapkan terima kasih yang sebesar-besarnya kepada Bapak Mochamad Hariadi, ST., M.Sc., Ph.D. dan Bapak Dr. Supeno Mardi Susiki Nugroho, ST., MT. atas bimbingan selama mengerjakan penelitian hingga memungkinkan terwujudnya hasil penelitian ini.
5. Ibu Susi Juniastuti, ST., M.Eng. atas bantuannya dalam penelitian ini.
6. Bapak-ibu dosen pengajar Bidang Studi Jaringan Cerdas Multimedia beserta dosen bidang studi lainnya, atas pengajaran, bimbingan, serta perhatian yang diberikan kepada penulis selama ini.
7. Seluruh teman-teman *B201-crew* Laboratorium Telematika atas berbagai macam saran dan masukannya beserta berbagai motivasi yang membangun dan menghancurkan.
8. Segenap civitas akademik Departemen Teknik Elektro beserta Departemen Teknik Komputer ITS untuk berbagai dukungannya secara langsung maupun tidak langsung.
9. Para anggota forum.unity3d.com yang tidak bisa saya sebut satu-persatu. Terima kasih atas bantuannya dalam penelitian ini.

Selama pengerjaan hingga terselesaikannya buku ini, banyak sekali masukan dari berbagai pihak dalam rangka menyempurnakan buku ini. Segala kritik dan saran setelah selesainya buku ini sangat diharapkan terutama untuk penelitian kedepannya. Sebagaimana dikatakan oleh bapak Prof. Dr. Ir. Mauridhi Hery Purnomo, M.Eng. "Penelitian yang baik adalah penelitian yang berkelanjutan", segala isi dari buku ini diharapkan dapat membantu pengem-

bangan penelitian lain demi keberlanjutan pengembangan ilmu pengetahuan yang membawa maslahat kepada umat manusia.

Surabaya, Januari 2018

Moch Fachri

DAFTAR ISI

Pernyataan Keaslian **vii**

Abstrak **ix**

Abstract **xi**

Kata pengantar **xiii**

DAFTAR ISI **xv**

DAFTAR GAMBAR **xvii**

DAFTAR TABEL **xix**

NOMENKLATUR **xxi**

1 PENDAHULUAN **1**
- 1.1 Latar belakang . . . 1
- 1.2 Rumusan Masalah . . . 2
- 1.3 Tujuan . . . 2
- 1.4 Batasan masalah . . . 3
- 1.5 Kontribusi . . . 3

2 KAJIAN PUSTAKA **5**
- 2.1 *Reciprocal Velocity Obstacles* . . . 5
- 2.2 Simulasi Kerumunan . . . 10
- 2.3 *Leader Following Behaviors* . . . 12

3 METODOLOGI PENELITIAN **13**
- 3.1 Desain Navigasi . . . 15
  - 3.1.1 *Seeking* . . . 15
  - 3.1.2 *Steering* . . . 16
  - 3.1.3 *Collision Avoidance* . . . 19
  - 3.1.4 *Collision detection* . . . 23
- 3.2 Pengambilan data . . . 24
- 3.3 Desain Simulasi . . . 25

4 HASIL DAN PEMBAHASAN **31**
- 4.1 Sistem Navigasi Agen . . . 31
  - 4.1.1 Proses *Seeking* . . . 31
  - 4.1.2 Proses *Steering* . . . 32
  - 4.1.3 Proses *Collision Avoidance* . . . 33
  - 4.1.4 Proses *Collision Detection* . . . 35
- 4.2 Sistem ekstraksi data . . . 36
- 4.3 Hasil Simulasi . . . 36

4.3.1 Skenario Pertama 37
4.3.2 Skenario Kedua 41

5 Kesimpulan **51**

DAFTAR PUSTAKA **53**

LAMPIRAN **55**

Biografi Penulis **59**

DAFTAR GAMBAR

2.1 Konsep penghindaran tabrakan dengan *Velocity Obstacle* 5
2.2 Osilasi yang terjadi pada penghindaran dengan *Velocity Obstacle* 7
2.3 Konsep penghindaran tabrakan dengan *Reciprocal Velocity Obstacle* 8
2.4 Gerak penghindaran tabrakan dengan RVO 9
2.5 RVO dengan penambahan pembobotan penghindaran 10
2.6 *Navigasi multiagen dengan RVO* 11
2.7 *Leader Following Behavior* 12

3.1 *Flowchart* proses pergerakan agen setiap cycle 14
3.2 *Flowchart seeking* 16
3.3 *Diagram alir proses steering pada [1]* 17
3.4 *Diagram Alir proses steering setelah diubah* 17
3.5 Proses *collision avoidance* pada [2] 20
3.6 *Diagram alir proses collision avoidance dengan penambahan paramater prioritas penghindaran* 21
3.7 Ilustrasi *leader following behavior* pada 22
3.8 *Diagram alir tahap pengecekan terjadinya tabrakan* 23
3.9 Skenario simulasi 26
3.10 Contoh rintangan statis pada lingkungan dekat area A simulasi 27
3.11 Contoh rintangan statis yang berada pada area tengah lingkungan simulasi 28
3.12 Contoh rintangan statis pada lingkungan dekat area B simulasi . 28

4.1 Hasil *seeking* agen *leader & follower* 32
4.2 Pengejaran oleh agen *follower* yang tertinggal 33
4.3 *Steering* agen *follower* yang berada di depan *leader*-nya 34
4.4 Hasil penghindaran tabrakan dua agen dengan tingkat prioritas yang sama 34
4.5 Hasil penghindaran tabrakan dua agen dengan tingkat prioritas yang berbeda 35
4.6 Hasil pengujian deteksi tabrakan 35
4.7 Contoh UI ekstraksi data 36
4.8 Salah satu cuplikan simulasi skenario pertama 38
4.9 Grafik perbandingan waktu simulasi skenario pertama 39
4.10 Grafik waktu simulasi skenario pertama dengan prioritas yang sama 39
4.11 Grafik waktu simulasi skenario pertama dengan prioritas yang sama 40
4.12 Contoh kejadian *deadlock* yang terjadi pada simulasi 41
4.13 Jumlah tabrakan yang terjadi pada skenario pertama dengan tingkat prioritas yang sama 41

4.14 Jumlah tabrakan yang terjadi pada skenario pertama dengan tingkat prioritas yang berbeda 42
4.15 Grafik perbandingan jumlah tabrakan yang terjadi pada simulasi skenario pertama 42
4.16 Salah satu cuplikan simulasi skenario kedua 43
4.17 Cuplikan lain simulasi skenario kedua 44
4.18 Grafik perbandingan waktu simulasi skenario pertama 45
4.19 Grafik waktu simulasi skenario pertama dengan prioritas yang sama 45
4.20 Grafik waktu simulasi skenario pertama dengan prioritas yang sama 45
4.21 Contoh kejadian *deadlock* yang terjadi pada simulasi skenario kedua 46
4.22 Jumlah tabrakan yang terjadi pada skenario kedua dengan tingkat prioritas yang sama 46
4.23 Jumlah tabrakan yang terjadi pada skenario kedua dengan tingkat prioritas yang berbeda 47
4.24 Grafik perbandingan jumlah tabrakan yang terjadi pada simulasi skenario kedua 48
4.25 Grafik penurunan *frame rate* simulasi skenario pertama 48
4.26 Grafik penurunan *frame rate* simulasi skenario kedua 49

DAFTAR TABEL

4.1 Hasil pengujian fungsi pengambilan data 36

A1 Data hasil simulasi skenario satu kelompok kerumunan dengan tingkat prioritas penghindaran sama 55
A2 Data hasil simulasi skenario satu kelompok kerumunan dengan tingkat prioritas penghindaran berbeda 55
A3 Data hasil simulasi skenario dua kelompok kerumunan dengan tingkat prioritas penghindaran sama 56
A4 Data hasil simulasi skenario dua kelompok kerumunan dengan tingkat prioritas penghindaran berbeda 56

Halaman ini sengaja dikosongkan

NOMENKLATUR

A	= Posisi agen A.
B	= Posisi agen B.
$\mathbf{a}$	= arah vektor agen A.
$\mathbf{b}$	= arah vektor agen B.
p	= Titik referensi.
v	= Arah *cast*.
t	= Waktu.
VO	= *Velocity Obstacle*.
$\mathbf{v}$	= kecepatan (*velocity*).
RVO	= *Reciprocal velocity Obstacle*.
α	= Bobot penghindaran.
$V_{follower}$	= Kelajuan agen *follower*.
V_{leader}	= Kelajuan agen *leader*.
SM	= Pengali kecepatan.
D_{agent}	= Jarak agen *follower* terhadap posisi *leader*.
D_{eq}	= Jarak yang diinginkan untuk agen *follower* memiliki kelajuan yang sama dengan agen *leader*.
D_{LoS}	= LoS agen *follower*.
n	= Konstanta.

Halaman ini sengaja dikosongkan

BAB 1

PENDAHULUAN

1.1 Latar belakang

Simulasi kerumunan merupakan proses permodelan gerakan dari sekumpulan entitas bergerak yang disebut agen dalam jumlah besar dengan kepadatan tinggi. Secara umum, simulasi kerumunan memiliki kerumunan agen dengan masing-masing agen pergi ke titik tujuan tertentu pada dunia simulasi. Pergerakan agen dalam simulasi kerumunan tidak terlepas dari navigasi multi-agen.

Masalah yang muncul dalam navigasi multi-agen berelasi *leader-follower* pada simulasi kerumunan sulit dihindari. Masalah ini muncul akibat dari ketergantungan gerak *follower* kepada agen *leader*-nya sehingga diperlukan metode navigasi khusus. Navigasi multi-agen ini perlu memastikan manuver penghindaran agen *follower* yang memberikan keleluasaan navigasi agen *leader*. Navigasi agen *leader* berperan penting, karena navigasi agen *leader* menentukan gerakkan keleuruhan kelompok menuju titik tujuannya dengan jarak terpendek secara aman, dalam artian tidak terjadi kemacetan antar agen maupun rintangan yang menghadang. Kesalahan navigasi satu agen *follower* saja bisa mengakibatkan navigasi agen *leader* terganggu sehingga seluruh kerumunan tidak dapat bergerak. Karenanya dikembangkan berbagai macam metode navigasi multi-agen agar agen-agen dapat bergerak dalam kerumunan berkepadatan tinggi dengan aman.

Reciprocal Velocity Obstacle (RVO) sebagai salah satu metode navigasi multi-agen memiliki banyak fitur yang masih dapat dikembangkan. Salah satunya adalah penggunaan variasi tingkat prioritas penghindaran untuk navigasi agen-agen yang memiliki relasi pemimpin dan pengikut (*leader* & *follower*). Penelitian mengenai RVO dengan relasi *leader follower* pada penelitian[2]

menggunakan *leader following behaviour*. Pada penelitian tersebut, *steering* penghindaran agen *follower* yang berada didepan *leader* memakai konsep yang dikemukakan pada [1] dengan prioritas penghindaran yang homogen. Oleh karena itu diperlukan penelitian keandalan metode RVO pada navigasi multi-agen dengan varian tingkat pembebanan prioritas untuk meningkatkan performansi navigasi dengan *leader following behaviour*. Hasil penelitian ini bisa diterapkan pada simulasi kerumunan yang membutuhkan skenario kerumunan agen *follower* mengikuti seorang agen *leader*.

1.2 Rumusan Masalah

Pada simulasi kerumunan dengan *leader following behavior*, agen *follower* tidak memiliki informasi koridor lingkungan simulasi sehingga perlu mengikuti agen *leader* yang mampu menuntun kelompok kerumunan menuju titik tujuan kelompok. Agen *leader* ini memiliki informasi koridor lingkungan simulasi. Terganggunya navigasi agen *leader* yang disebabkan agen *follower* menyebabkan pergerakan kelompok kerumunan mengalami kemacetan. Kegagalan navigasi multi-agen berelasi *leader* & *follower* diakibatkan agen *follower* tidak mampu memberikan navigasi yang bebas kepada *leader*-nya. Kesalahan navigasi tersebut dapat menyebabkan kerumunan terjebak dalam kemacetan (*deadlock*). Semua itu mengakibatkan kelompok kerumunan kesulitan dalam mencapai titik tujuannya.

1.3 Tujuan

Navigasi multi-agen menggunakan RVO tidak memerlukan komunikasi eksplisit dengan agen lain dalam penghindaran, dan bebas dari gerak osilasi seperti yang terjadi pada *Velocity Obstacle* biasa[3]. Karena itu RVO dapat mengatasi masalah penghindaran rintangan tanpa koordinasi terpusat di simulasi kerumunan. Penelitian ini bertujuan untuk memodifikasi navigasi multi-agen berelasi *leader-follower* berbasis RVO menggunakan prioritas penghindaran. Modifikasi dengan penambahan prioritas penghindaran dimaksudkan untuk memberikan hasil penghindaran yang lebih baik antara agen *follower*

dengan *leader*-nya Dari hasil simulasi yang didapat, diharapkan penggunaan prioritas penghindaran ini dapat meningkatkan performansi navigasi multi-agen dengan RVO terutama pada *leader following behaviour*.

1.4 Batasan masalah

Batasan masalah pada penelitian ini adalah:

1. Pergerakan agen hanya pada bidang planar dimensi dua.
2. Jumlah agen dibatasi hingga 500 agen.
3. Ukuran dan tipe agen tidak divariasikan.

1.5 Kontribusi

Penelitian mengenai navigasi multi-agen merupakan penelitian di beberapa bidang keilmuan. Dalam teknologi permainan (*game technology*), navigasi multi-agent dipakai untuk mengatur pergerakan *virtual agent* dalam permainan dengan jumlah agen besar di ruang terbatas(dunia permainan). Selain dalam bidang teknologi permainan, simulasi ini dapat diimplementasikan dalam bidang robotika untuk navigasi robot. Perkembangan penelitian ini dapat digunakan untuk riset dan penelitian mengenai alternatif lain dalam navigasi multi-agen terutama untuk simulasi kerumunan.

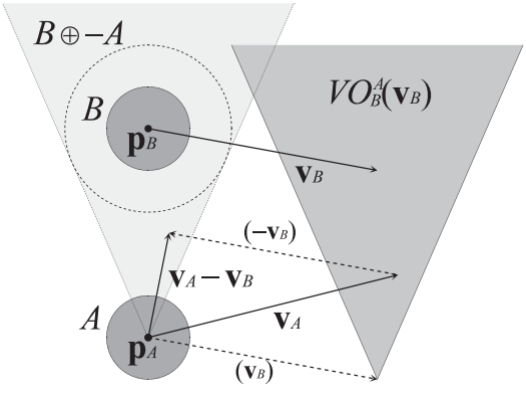
Halaman ini sengaja dikosongkan

BAB 2
KAJIAN PUSTAKA

Demi mendukung penelitian ini, dibutuhkan beberapa teori penunjang sebagai bahan acuan dan referensi. Dengan demikian penelitian ini menjadi lebih terarah. Berikut ini akan dipaparkan teori dasar mengenai RVO, simulasi kerumunan, dan *leader following behaviour*.

2.1 *Reciprocal Velocity Obstacles*

Reciprocal Velocity Obstacles (RVO) adalah konsep untuk *local reactive collision avoidance* yang digunakan dalam navigasi multiagen yang tiap agen melakukan navigasi secara independen tanpa ada komunikasi secara eksplisit[3]. RVO memperbaiki konsep Velocity Obstacle[4] yang sebelumnya dengan tujuan menghasilkan gerakan agen yang aman dari tabrakan dengan agen lain dan rintangan, serta bebas dari osilasi yang terjadi pada *Velocity Obstacle*.



Gambar 2.1: Konsep penghindaran tabrakan dengan *Velocity Obstacle*[3]

Gambar 2.1 menunjukkan konsep penghindaran tabrakan dengan *Velocity Obstacle* yang dijelaskan oleh Jur van der Berg, dkk[3] sebagai berikut. Misalkan agen A adalah agen sedang bertranslasi di bidang dengan titik referensinya pada $\mathbf{P}_A$,dan B adalah rintangan planar(bergerak) dengan titik referensi pada $\mathbf{P}_B$. *Velocity Obstacle* $VO_B^A(\mathbf{v}_B)$ dari rintangan B ke agent A adalah himpun-

an yang terdiri dari semua kecepatan $\mathbf{v}_A$ untuk A yang akan menghasilkan tabrakan pada suatu momen waktu bersamaan dengan rintangan B bergerak dengan kecepatan $\mathbf{v}_B$.

Velocity Obstacle dari agen A terhadap rintangan B tersebut dapat didefinisikan secara geometri sebagaimana pada persamaan 2.1

$$A \oplus B = \{\mathbf{a} + \mathbf{b} | \mathbf{a} \in A, \mathbf{b} \in B\}, -A = \{-\mathbf{a} | \mathbf{a} \in A\} \tag{2.1}$$

A = Posisi agen A.
B = Posisi agen B.
$\mathbf{a}$ = arah vektor agen A.
$\mathbf{b}$ = arah vektor agen B.

$A \oplus B$ adalah penjumlahan Minkowski dari dua objek A dan B dimana -A adalah refleksi objek A terhadap titik referensinya. Misal $\lambda(\mathbf{p}, \mathbf{v})$ menyatakan *ray* yang bermula di $\mathbf{p}$ dan bergerak ke arah $\mathbf{v}$, maka secara matematis dapat dinyatakan sebagaimana persamaan 2.2.

$$\lambda(\mathbf{p}, \mathbf{v}) = \{\mathbf{p} + t\mathbf{v} | t \geq 0\} \tag{2.2}$$

p = Titik referensi.
v = Arah *cast*.
t = Waktu.

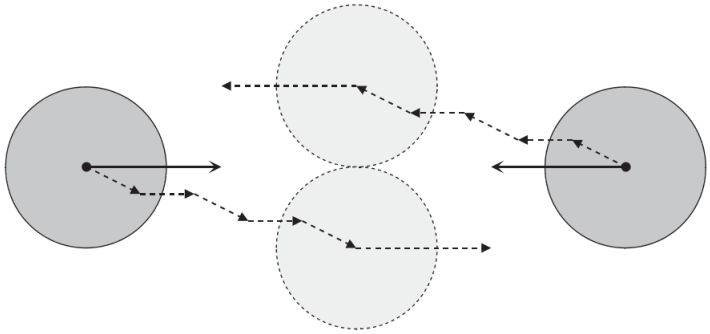
Jika *ray* yang bermula pada $\mathbf{p}_A$ dan mengarah ke kecepatan relatif dari A dan B($\mathbf{v}_A - \mathbf{v}_B$) memotong jumlah Minkowski dari B dan A yang berpusat di $\mathbf{p}_B$, Velocity Obstacle $\mathbf{v}_A$ berada didalam *Velocity Obstacle* dari B. Sehingga Velocity Obstacle B ke A didefinisikan sebagaimana persamaan 2.3

$$VO_B^A(\mathbf{v}_B) = \{\mathbf{v}_A | \lambda(\mathbf{p}_A, \mathbf{v}_A - \mathbf{v}_B) \cap B \oplus -A \neq \emptyset\} \tag{2.3}$$

VO = *Velocity Obstacle*.
$\mathbf{v}$ = kecepatan (*velocity*).

Ini berarti jika $\mathbf{v}_A \in VO_B^A(\mathbf{v}_B)$, A dan B akan bertabrakan pada suatu titik waktu. Jika $\mathbf{v}_A$ diluar *Velocity Obstacle* dari B, kedua objek tiada akan pernah bertabrakan. Jika $\mathbf{v}_A$ ada pada perbatasan *Velocity Obstacle*, maka A akan bersentuhan dengan B pada suatu momen waktu.

Konsep *Velocity Obstacle* dapat digunakan untuk navigasi multiagen dengan tiap agen menganggap agen lain sebagai rintangan bergerak dan memilih kecepatan untuk dirinya sendiri yang berada diluar *Velocity Obstacle* yang diinduksikan dari agen lain. Namun, pendekatan ini menghasilkan pergerakan osilasi yang tidak diinginkan.

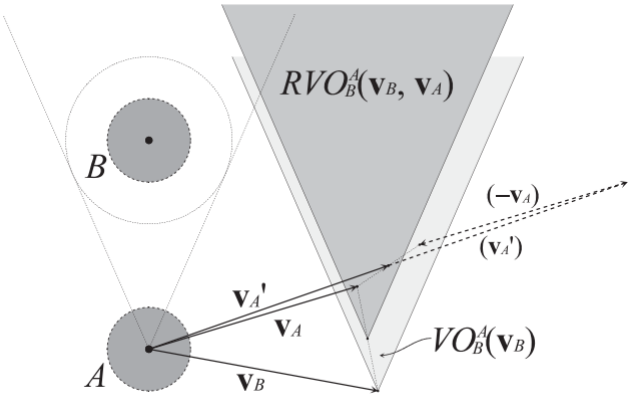


Gambar 2.2: Osilasi yang terjadi pada penghindaran dengan *Velocity Obstacle*[3]

Misalkan pada situasi yang ditunjukkan gambar 2.2, dua agen A dan B bergerak dengan kecepatan $\mathbf{v}_A$ dan $\mathbf{v}_B$ bersamaan sehingga $\mathbf{v}_A \in VO_B^A(\mathbf{v}_B)$ dan $\mathbf{v}_B \in VO_A^B(\mathbf{v}_A)$, Karenanya, jika kedua agen tetap melanjutkan kecepatan yang telah ada akan berakibat pada tabrakan antara keduanya. Untuk menghindarinya, agen A akan mengubah kecepatannya menjadi $\mathbf{v}_A$ yang berada diluar *Velocity Obstacle* $B(\mathbf{v}_A \notin VO_A^B(\mathbf{v}_B))$. Disaat bersamaan agen B juga mengubah kecepatannya menjadi $\mathbf{v}_B$ yang berada diluar *Velocity Obstacle* $A(\mathbf{v}_B \notin VO_A^B(\mathbf{v}_B))$.

Masalah muncul ketika 2 agen berada pada kecepatan $\mathbf{v}_A$ dan $\mathbf{v}_B$, kecepatan lama($\mathbf{v}_A$ dan $\mathbf{v}_B$) sudah berada diluar *Velocity Obstacle* A dan B. Kedua agen akan kembali pada kecepatan yang lama karena dengan kecepatan yang lama, jarak menuju titik tujuan masing-masing agen adalah yang terpendek.

Pada siklus selanjutnya kedua agen dengan kecepatan lama akan bertabrakan sehingga kecepatan mereka berdua akan berubah lagi menjadi $\mathbf{v}_A$ dan $\mathbf{v}_B$. Hal ini berulang hingga kedua agen berpapasan. Perubahan kecepatan terus menerus dari $\mathbf{v}_A$ dan $\mathbf{v}_B$ ke $\mathbf{v}_A$ dan $\mathbf{v}_B$ kemudian sebaliknya inilah yang menimbulkan gerak osilasi yang ditunjukkan di gambar 2.2.



Gambar 2.3: Konsep penghindaran tabrakan dengan *Reciprocal Velocity Obstacle*. Sumber gambar:[3]

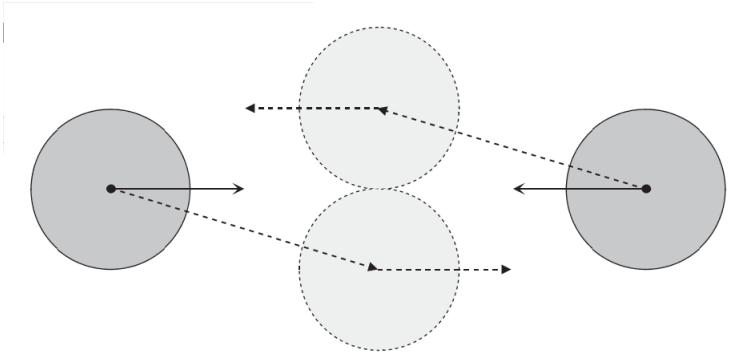
Untuk menghindari gerak osilasi inilah, Jur van der Berg, dkk[3] merumuskan metode baru yang dikembangkan dari *Velocity Obstacle* yaitu *Reciprocal Velocity Obstacle*(RVO) seperti yang diilustrasikan pada gambar 2.3. Ide dasarnya sederhana, Kecepatan(*velocity*) baru yang dipilih untuk menghindari tabrakan merupakan rerata dari kecepatan agen sekarang dan kecepatan yang berada diluar *Velocity Obstacle* agen lain. RVO didefinisikan sebagai persamaan 2.4.

$$RVO^A_B(\mathbf{v}_B, \mathbf{v}_A) = \{\mathbf{v'}_A | \mathbf{v'}_A - \mathbf{v}_A \in VO^A_B(\mathbf{v}_B)\} \tag{2.4}$$

$RVO = Reciprocal\ velocity\ Obstacle.$

Gambar 2.3 menunjukkan *Reciprocal Velocity Obstacle* $RVO^A_B\ (\mathbf{v}_B, \mathbf{v}_A)$ dari agen B ke agen A berisi semua kecepatan agen A yang merupakan rerata kecepatan $\mathbf{v}_A$ dan kecepatan yang berada didalam *Velocity Obstacle* $VO^A_B(\mathbf{v}_B)$

agen B. Kemudian RVO_B^A secara geometris dapat diintepretasikan sebagai *Velocity Obstacle* VO_B^A yang ditranslasikan supaya nilai puncaknya berada di $\frac{\mathbf{v}_A+\mathbf{v}_B}{2}$. Gerakan penghindaran yang dihasilkan dapat dilihat pada gambar 2.4.



Gambar 2.4: Gerak penghindaran tabrakan dengan RVO[3]

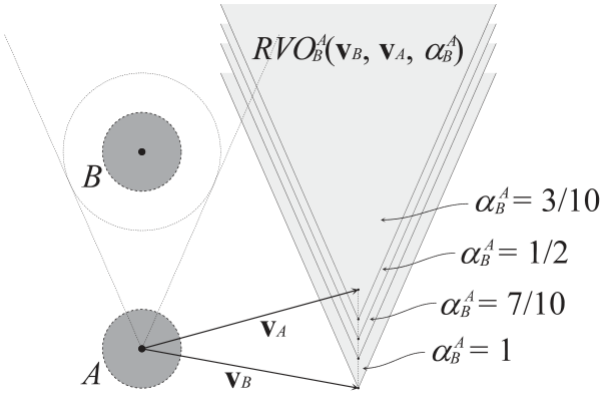
Hal tersebut dengan asumsi bahwa tiap agen melakukan beban penghindaran yang sama satu sama lain. Tiap agen dapat diberi level prioritas natural untuk membedakan beban penghindaran antar agen. Beban ini didefinisikan sebagai nilai α, dimana beban penghindaran agen A terhadap agen B didefinisikan sebagai α_B^A, maka sebaliknya nilai $\alpha_B^A = 1 - \alpha_A^B$. untuk nilai beban penghindaran dengan prioritas sama antar keduanya, maka $\alpha_A^B = \alpha_B^A = \frac{1}{2}$.

Prinsip dasarnya adalah pembobotan dipakai dalam memilih velocity baru, dimisalkan pada agen A, velocity terbaru merupakan pembobotan rerata dari nilai $1 - \alpha_B^A$ terhadap velocity saat itu ($\boldsymbol{v}_a$) dan α_B^A yang berada diluar nilai $VO_B^A(\boldsymbol{v}_b)$ dari agen B. Agen B juga melakukan hal yang sama dengan nilai bobot penghindaran yang berlawanan, sehingga nilai yang diambil berasal dari nilai $1 - \alpha_A^B$ terhadap velocity saat itu ($\boldsymbol{v}_b$) dan α_A^B yang berada diluar nilai $VO_A^B(\boldsymbol{v}_a)$ dari agen A. Sehingga formulasi RVO untuk agen A terhadap B didefinisikan pada persamaan 2.5

$$RVO_B^A(\boldsymbol{v}_B, \boldsymbol{v}_A) = \left\{ \boldsymbol{v'}_A | \frac{1}{\alpha_B^A} \boldsymbol{v'}_A + (1 - \alpha_B^A)\boldsymbol{v}_A \in VO_B^A(\boldsymbol{v}_b) \right\} \tag{2.5}$$

α = Bobot penghindaran.

Konsekuensi dari pembobotan ini adalah pergeseran *collision cone* tiap agen yang mengakibatkan nilai *velocity* terpilih juga ikut tergeser sebagaimana ditunjukkan pada gambar 2.5.



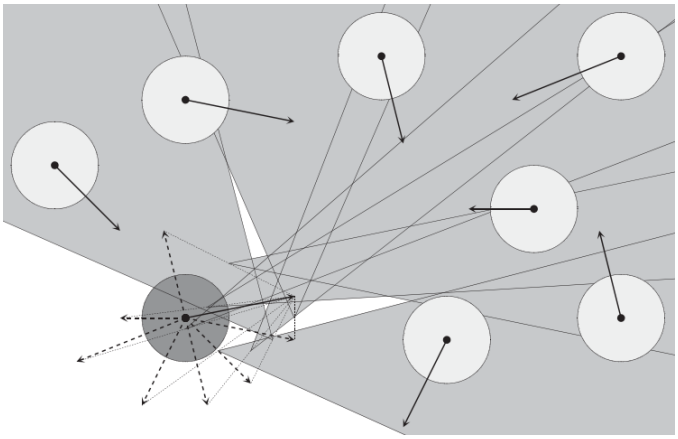
Gambar 2.5: RVO dengan penambahan pembobotan penghindaran[3]

Untuk navigasi multiagen, agen akan memilih *velocity* berdasarkan *valid velocity* yang dihasilkan dari RVO yaitu *velocity* yang berada diluar gabungan seluruh area *reciprocal velocity obstacles* dan yang paling mendekati velocity yang diinginkan. Dapat dilihat pada gambar 2.6, area gerak yang diarsir gelap menunjukkan *invalid velocity* untuk agen berwarna gelap yang merupakan *union* dari RVO yang dihasilkan agen berwarna cerah. *Velocity* yang dipilih oleh agen tersebut adalah paling mendekati velocity yang diinginkan(*velocity* referensi) dan masih berada pada *velocity* yang valid.

2.2 Simulasi Kerumunan

Kerumunan adalah sekumpulan individu di dalam lingkungan fisik yang sama, berbagi tujuan yang sama[5]. Simulasi kerumunan dipakai untuk memproduksi gerakan dan menampilkan perilaku sekumpulan kerumunan yang alami[6]. Ada tiga pendekatan yang luas untuk simulasi kerumunan [5] yaitu:

1. Pendekatan fluida. Dapat diobservasi bahwa gerakan kerumunan pada level makroskopik mirip dengan aliran fluida. Telah ada percobaan yang



Gambar 2.6: *Navigasi multiagen dengan RVO*[3]

berhasil memodelkan agen menggunakan hukum fisika seputar dinamika fluida.

2. Pendekatan otomata seluler yaitu suatu pendekatan sistem diskret dan dinamik yang memodelkan suatu struktur sel yang mendasarkan status suatu sel pada status sel yang berhubungan langsung dengannya.

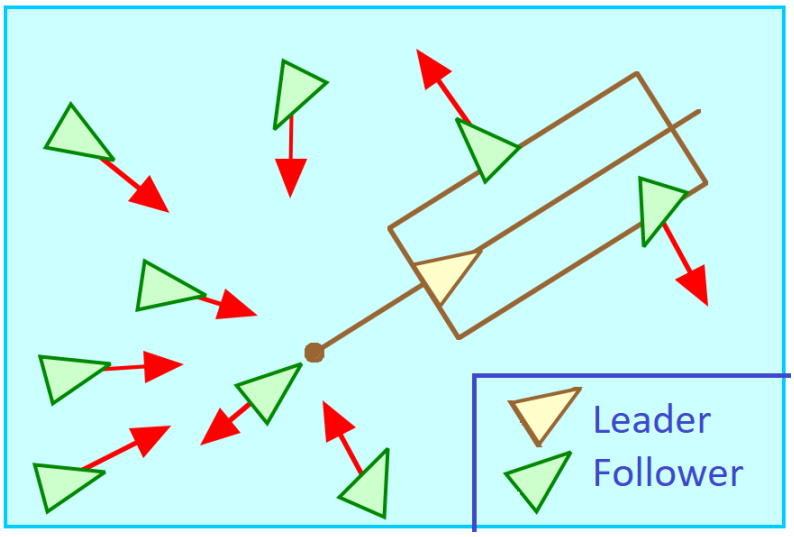
3. Pendekatan Partikel, kemungkinan pendekatan ini yang paling sering digunakan pada mayoritas percobaan yang dijelaskan di berbagai literatur. Sering disebut sebagai pendekatan partikulat atau atomik. Dalam skema ini setiap agen dianggap sebagai satu entitas dan interaksi antar agen dimodelkan secara individual berdasarkan hukum fisik atau sosial.

Simulasi kerumunan ini dapat didekati dalam level makroskopik dan mikroskopik. Pada level makroskopik, fokus utama berada di kerumunan massa orang-orang bergerak dengan gerakan yang realistis, tanpa perlu terfokus oleh gerakan individual. Namun pada level mikroskopik, pada tiap-tiap individu agen bergerak dalam pergerakan yang meyakinkan perlu difokuskan. Pendekatan pada level makroskopik maupun mikroskopik tergantung pada aplikasi dari simulasi yang akan dibuat. Secara umum pada level makroskopik metode pendekatan berbasis fluida adalah yang paling baik, sedangkan pada level mikroskopis metode yang sering dipakai adalah metode pendekatan berbasis partikel.

2.3 *Leader Following Behaviors*

Leader following behavior adalah *steering behavior* untuk gerak *boids* dimana *boids* diharuskan bergerak mengikuti *leader* yang telah ditentukan. *Boids* ini disebut sebagai *follower*. Berdasarkan *steering behavior* untuk *boids* yang didesain oleh Craig W. Reynolds[1], *Leader Following Behaviors* dibentuk dari dua macam *behavior* yaitu *arrival* dan *separation*. Kedua behavior tersebut terbentuk karena secara umum, *follower* ingin tetap bergerak dekat dengan *leader* tanpa menghalangi pergerakannya disertai dengan penghindaran tabrakan dengan *follower* lain seperti pada gambar 2.7.

Arrival behavior mengakibatkan *follower* berusaha bergerak ke titik tujuannya yaitu posisi *leader* dan berusaha melambat ketika jarak dengan leader semakin mendekat. Sebaliknya, semakin tertinggal agen tersebut dengan *leader* maka kelajuan agen tersebut akan meningkat untuk mengejar ketertinggalannya. Jika *follower* berada di area rectangular di depan *leader*, maka follower akan melakukan *streering* menghindari jalur *leader* sebelum melanjutkan *arrival behavior*. Adapun *separation* diimplementasikan untuk mencegah para *follower* bergerombol satu sama lain. Hal ini sebagai tindakan pencegahan terjadinya *collision* antar *follower*



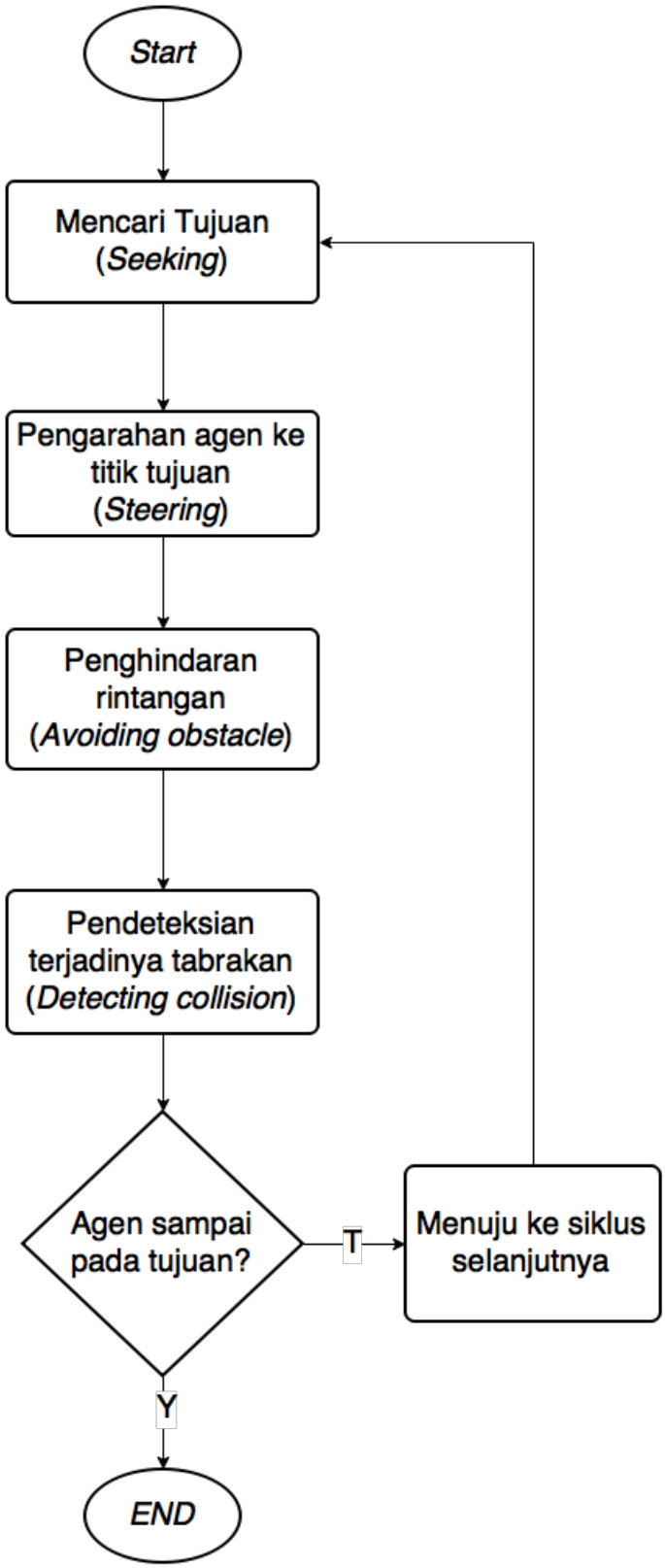
Gambar 2.7: *Leader Following Behavior*[1]

BAB 3

METODOLOGI PENELITIAN

Penelitian ini dilaksanakan sesuai dengan desain sistem berikut dengan implementasinya. Tujuan yang ingin dicapai dari penelitian ini adalah memperbaiki navigasi multiagen berelasi *leader-follower* berbasis RVO menggunakan prioritas penghindaran. Sebelum melakukan pengukuran performa, infrastuktur dan sistem dari simulasi kerumunan harus dibangun terlebih dahulu. Karena itulah pada bab ini akan dipaparkan metodologi yang digunakan pada penelitian ini.

Infrastruktur yang pertama dibuat adalah desain simulasi. Desain simulasi dibuat berdasarkan salah satu skenario yang dipakai dalam pengujian performansi navigasi agen kerumunan[2]. Skenario simulasi yang dipakai untuk pengujian memakai skenario seperti yang ditunjukkan pada subbab 3.3. Pada penelitian ini akan ditambahkan relasi *leader & follower* sehingga agen *follower* perlu mengikuti masing-masing *leader*-nya, dan *leader* tersebut yang akan bergerak ke titik tujuan kelompok kerumunan. Selain itu pada skenario yang dibuat, terdapat banyak rintangan statis yang menjadi tantangan sistem navigasi yang didesain dengan memakai tingkat prioritas. Skenario simulasi didesain sedemikian rupa sehingga menyebabkan agen *follower* berada di depan agen *leader*. Tantangan dalam memberikan keleluasan navigasi inilah yang akan menjadi bagian utama yang paling berpengaruh dalam hasil performansi yang diberikan. Sebelum membahas lebih lanjut mengenai skenario simulasi, terlebih dahulu akan dibahas lebih dalam tentang desain *inner working* sistem navigasi multiagen.



Gambar 3.1: Flowchart proses pergerakan agen setiap cycle

3.1 Desain Navigasi

Rancangan dasar dari pergerakan agen di simulasi ini ditunjukkan pada gambar 3.1. Kalkulasi pergerakan agen terdiri dari empat tahap untuk setiap siklus berjalannya simulasi yang pada Unity dikalkulasi setiap *frame*. Pergerakan agen dibangun menggunakan sistem navigasi yang telah disediakan Unity. Dalam pergerakan agen ini ada dua hal yang menjadi perhatian utama yaitu cara agen menemukan destinasinya dan cara agen menuju destinasinya.

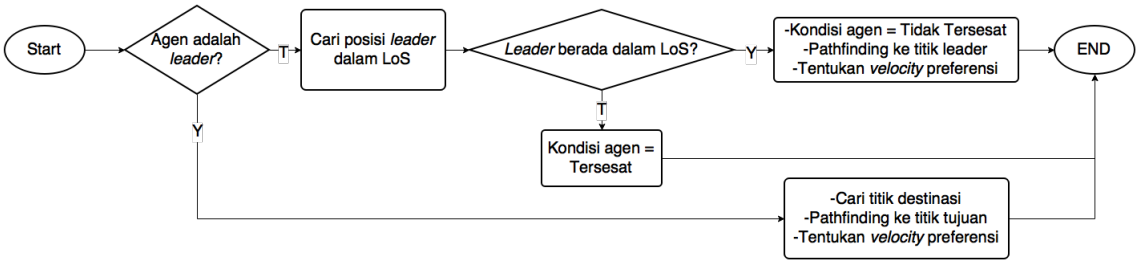
3.1.1 *Seeking*

Bagian ini adalah bagian dimana agen pada simulasi menentukan titik tujuan pergerakannya(destinasi). Agen *leader* dan *follower* memiliki jenis tujuan yang berbeda. Sebagaimana ditunjukkan pada *flowchart* di gambar 3.2 agen *leader* memiliki titik tujuan yang telah ditentukan sedangkan *follower* memiliki titik tujuan pada *leader* tiap *follower*. Pada salah satu skenario pengujian, agen *leader* memiliki beberapa titik tujuan dengan *waypoint* tertentu. Agen *leader* akan mencari jalur menuju destinasinya. Pencarian jalur dari titik agen menuju titik tujuan memakai algoritma A*.

Untuk agen *follower*, setiap agen memiliki seorang *leader* yang telah ditentukan dan posisi *leader*. Agen *follower* pertama kali akan mencari *leader* dalam LoS(*Line of Sight*). Bila pada siklus tersebut, agen tidak dapat menemukan *leader*, maka status agen tersebut dianggap tersesat. Agen *follower* yang telah menemukan *leader*-nya di dalam LoS akan dianggap sebagai agen yang tidak tersesat. Selanjutnya agen *follower* akan mencari jalur menuju posisi *leader* dengan algoritma A*. Ketika agen *follower* tersebut kehilangan *leader*-nya dari LoS, agen *follower* tersebut akan menjadikan titik terakhir *leader* berada dalam LoS sebagai titik tujuan agen.

Hasil pencarian jalur ketitik tujuan masing-masing agen akan menghasilkan parameter jarak dan arah yang akan disimpan untuk proses selanjutnya dalam bentuk *velocity* preferensi. Nilai preferensi ini tidak bisa menghasilkan kecepatan karena masih tidak mencakup nilai kelajuan. Nilai kelajuan dari

velocity preferensi diambil dari nilai kelajuan hasil *velocity* dari siklus sebelumnya. Pada siklus pertama, nilai kelajuan dari velocity preferensi ini masih bernilai nol. Velocity preferensi telah menyertakan nilai jarak agen terhadap titik tujuannya, nilai nantinya yang akan digunakan pada proses selanjutnya yaitu proses steering pada subbab 3.1.2 untuk menentukan kelajuan referensi agen.



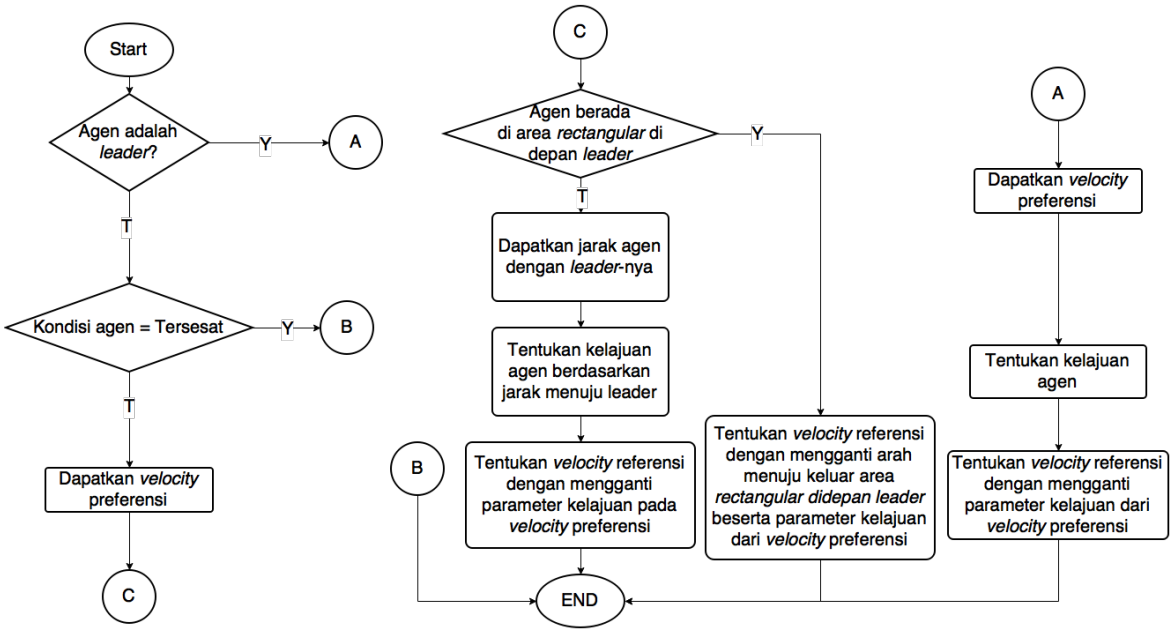
Gambar 3.2: *Flowchart seeking*

3.1.2 *Steering*

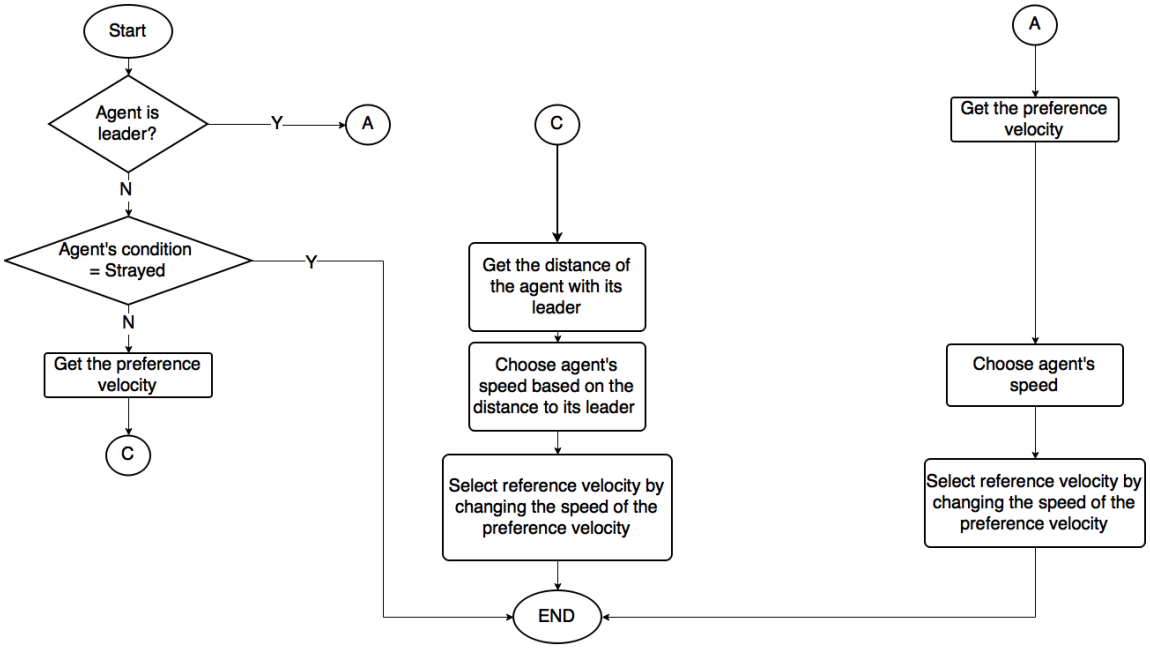
Steering behavior yang digunakan dalam penelitian ini adalah *leader following behavior*[[1]. Namun proses *steering* dari agen *follower* yang berada pada area *rectangular* didepan *leader*-nya tidak dilakukan. Hal ini dikarenakan proses *steering* tersebut yang ditunjukkan pada gambar 3.3 digantikan dengan pemberian prioritas penuh pada agen *leader* pada penghindaran yang akan dijelaskan pada bagian *collision avoidance*.

Setelah agen menentukan titik tujuannya serta mendapatkan *velocity preferensi* dari proses *seeking* pada subbab 3.1.1 maka proses selanjutnya adalah pengarahan agen menuju destinasinya yaitu *steering*. Dapat dilihat pada *flowchart* 3.4, agen *follower* bila tidak tersesat akan mengambil parameter *velocity* preferensi dari proses *seeking*. Parameter yang dipakai yaitu jarak dari agen tersebut ke *leader*-nya. Dari parameter ini akan ditentukan kelajuan(*speed*) agen dimana semakin jauh jaraknya akan semakin cepat kelajuannya agar bisa mengejar *leader*.

Dapat dilihat pada persamaan 3.1, kelajuan agen *follower* untuk *velocity* referensi bergantung pada kelajuan agen *leader* dan pengali kecepatan. Kela-



Gambar 3.3: *Diagram alir proses steering pada [1]*



Gambar 3.4: *Diagram Alir proses steering setelah diubah*

juan pengejaran sendiri diberikan pada persamaan 3.2. Kelajuan pengejaran bertambah secara eksponensial berbanding lurus dengan penambahan jarak dari agen *follower* ke *leader*. Hal ini berarti semakin jauh agen *follower* dari *leader*, maka penambahan kelajuan akan bertambah tinggi agar bisa mengimbangi ketertinggalan dengan agen *leader*. Persamaan kelajuan agen *follower*

secara keseluruhan dapat dilihat di persamaan 3.3.

$$V_{follower} = V_{leader} \times SM \tag{3.1}$$

$$SM = exp\left(\frac{D_{agent} - D_{eq}}{D_{LoS}}\right)^{n} \tag{3.2}$$

$$V_{follower} = V_{leader} \times exp\left(\frac{D_{agen} - D_{eq}}{D_{LoS}}\right)^{n} \tag{3.3}$$

$V_{follower}$ = Kelajuan agen *follower*.
V_{leader} = Kelajuan agen *leader*.
SM = Pengali kecepatan.
D_{agent} = Jarak agen *follower* terhadap posisi *leader*.
D_{eq} = Jarak yang diinginkan untuk agen *follower* memiliki kelajuan yang sama dengan agen *leader*.
D_{LoS} = LoS agen *follower*.
n = Konstanta, semakin rendah nilainya akan membuat penambahan nilai mendekati linier.

Untuk agen *leader*, parameter kelajuan yang diubah dari *velocity* preferensi ditentukan secara konstan. Hal ini mengakibatkan kelajuan pada *velocity* referensi *leader* adalah *velocity* ideal. Sehingga perubahan *velocity* hanya tergantung dari proses selanjutnya. Kelajuan agen yang dipilih akan mengubah parameter kelajuan yang berada *velocity* preferensi. *Velocity* ini disimpan sebagai *velocity* referensi.

Perbedaan utama pemilihan *velocity* referensi pada penelitian ini dengan

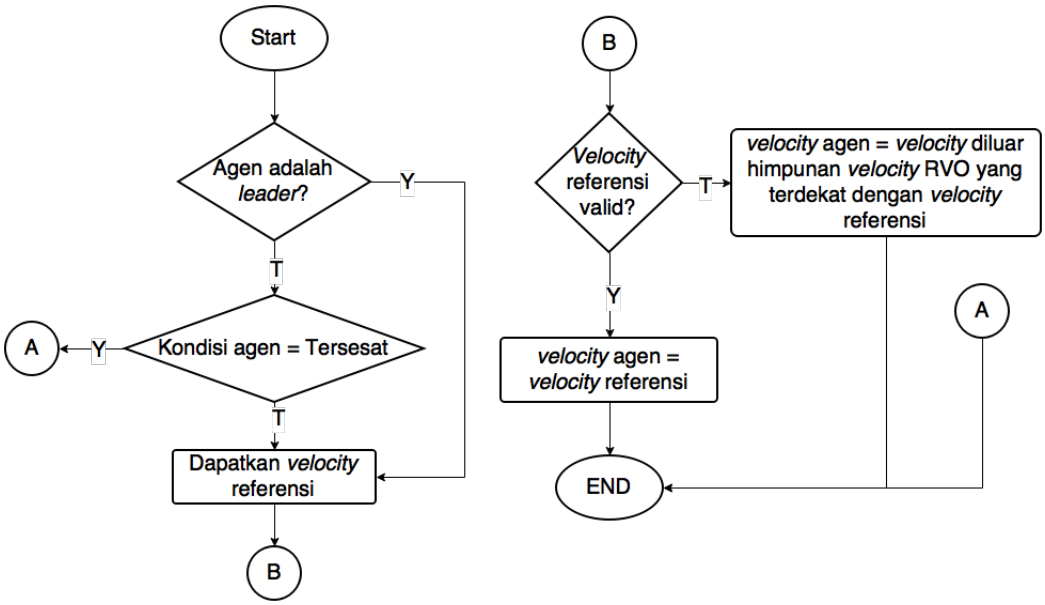
penelitian sebelumnya[2] terletak pada *steering* dari agen *follower* yang berada di depan agen *leader*. Pada metode yang dipakai di [1] dan [2], agen *follower* diberi nilai tambahan vektor keluar area *rectangular* yang berada di depan agen *leader* seperti pada gambar 2.7. Nilai vektor ini ditambahkan pada nilai *velocity* preferensi yang mengarahkan *follower* pada *leader*-nya. Hal yang demikian secara umum mengakibatkan nilai *velocity* referensi dari agen *follower* tidak bersebrangan dengan agen *leader*-nya. Namun pada kasus-kasus tertentu dimana terdapat banyak rintangan statis, agen *follower* tidak mampu mengosongkan area rektangular di depan *leader*. Dari masalah yang terjadi pada kasus tersebut, diadakan perubahan dengan menambahkan tingkat prioritas dimana terdapat pembedaan prioritas pembobotan penghindaran antara *leader*dengan *follower*.

3.1.3 *Collision Avoidance*

Collision Avoidance adalah penghindaran tabrakan baik itu rintangan statis maupun agen yang bergerak. Pengindaran dilakukan dengan menentukan gerak agen yang valid melalui pemilihan *valid velocity* berdasarkan algoritma RVO. Keluaran dari proses ini adalah penentuan *velocity* agen yang akan dipakai untuk siklus gerak saat itu.

Alur proses yang ditunjukkan pada *flowchart* di gambar 3.5 menggambarkan alur penentuan *velocity* agen. Untuk seluruh agen baik itu *leader* maupun *follower* (yang tidak tersesat), *Velocity* referensi dari proses 3.1.2 akan diambil untuk pengecekan validitas dari *velocity* referensi tersebut. *Velocity* referensi valid bila *velocity* tersebut tidak berada dalam *collision cone* dari RVO yang artinya, *velocity* referensi tidak mengakibatkan *collision*.

Velocity referensi sendiri berasal dari proses sebelumnya yaitu proses *steering* yang telah dijelaskan pada subbab 3.1.2. *Velocity* tersebut masih berupa referensi kemana perginya suatu agen sehingga belum dipilih untuk *velocity* agen pada siklus navigasi tersebut. Referensi ini merupakan nilai optimum bagi agen untuk bernavigasi pada suatu siklus. Nilai ini dimaksudkan untuk



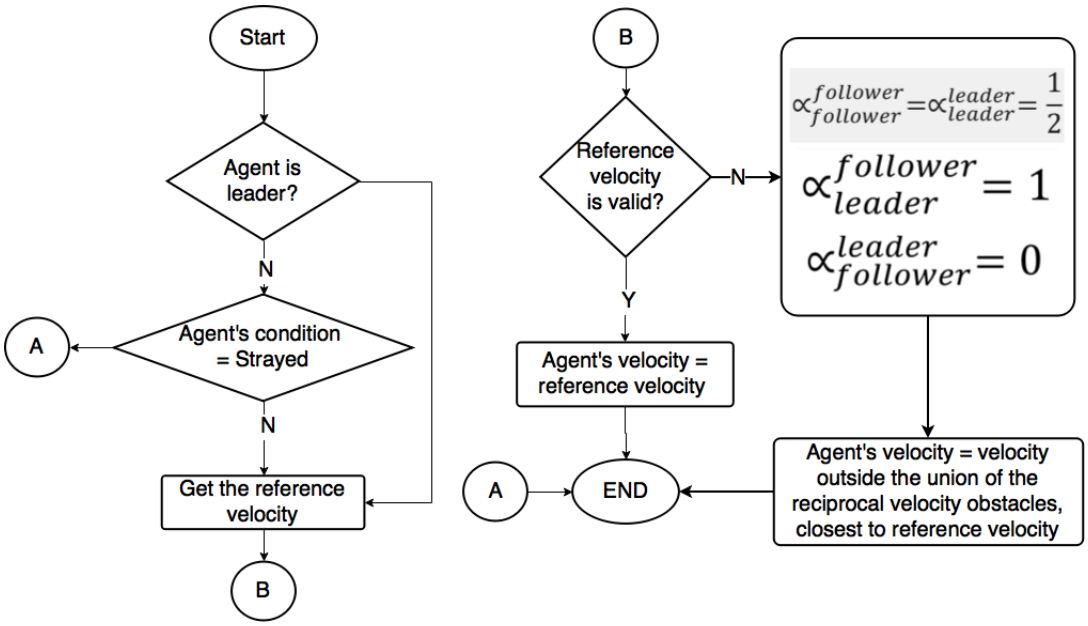
Gambar 3.5: Proses *collision avoidance* pada [2]

memenuhi tuntutan agen tersebut dalam megikuti aturan *steering bahavior* yang dipakai. Karena itulah proses *collision avoidance* menggunakan *velocity* dari proses *steering* sebagai referensi, dan memfinalisasi kevalidan dari *velocity* referensi tersebut.

Bila *velocity* referensi dinyatakan valid maka *velocity* referensi telah sah menjadi *velocity* agen untuk siklus saat itu. Sebaliknya bila *velocity* referensi tersebut tidak valid yang artinya *velocity* tersebut akan mengakibatkan tabrakan, maka sebagaimana pada penjelasan RVO dalam penghindaran *collision*[3], *velocity* agen akan dipilih dari *velocity* diluar himpunan *velocity* RVO yang terdekat dengan *velocity* referensi. Dalam simulasi ini terdapat tambahan *invalid velocity* yaitu *invalid velocity* untuk separasi antar agen. Dengan begini, proses *collision avoidance* mengkonsiderasi *velocity* referensi agar sebisa mungkin navigasi agen mendekati referensinya untuk bisa memenuhi *behaviour*-nya tanpa mengakibatkan *collision*.

Pada penelitian ini, RVO yang dipakai tidak diasumsikan memiliki beban penghindaran yang sama pada seluruh agen. Hal ini dikarenakan penggunaan beban prioritas yang dibedakan antara agen *leader* dan *follower* pada penghindaran baik antar agen maupun antar kelompok. Dengan ini, agen *leader* lebih

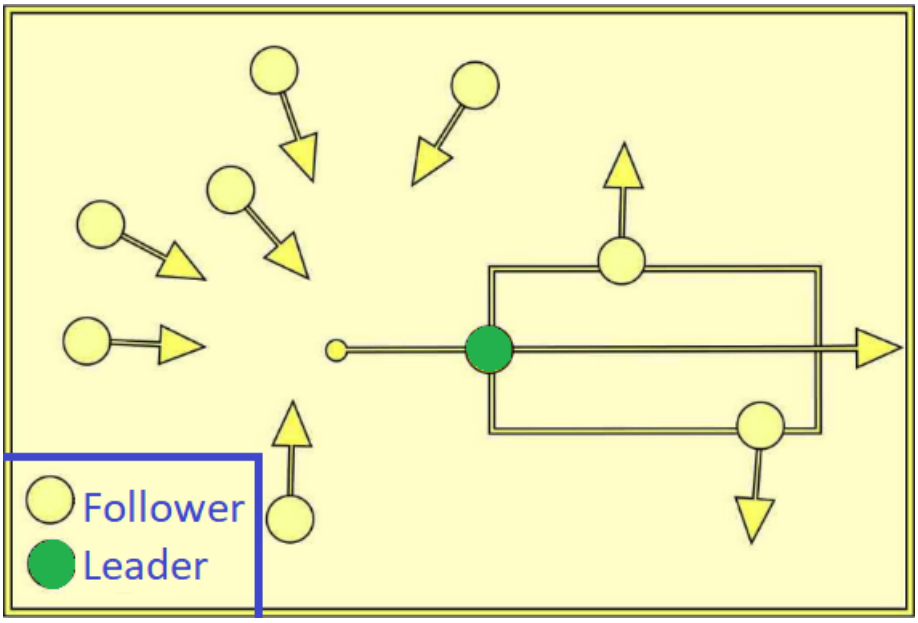
diprioritaskan terhadap agen *follower*. Sehingga agen *follower* yang berada di depan agen *leader*, akan menghindari agen *leader* tersebut. Penghindaran ini membuat agen *leader* tidak perlu mengubah *velocity* referensinya terhadap *follower* kelompoknya. Penghindaran inilah yang menggantikan proses *steering* untuk agen *follower* yang berada di area *rectangular* di depan *leader*-nya. Sehingga proses *collision avoidance* yang ditunjukkan pada gambar 3.5, diganti dengan proses pada gambar 3.6.



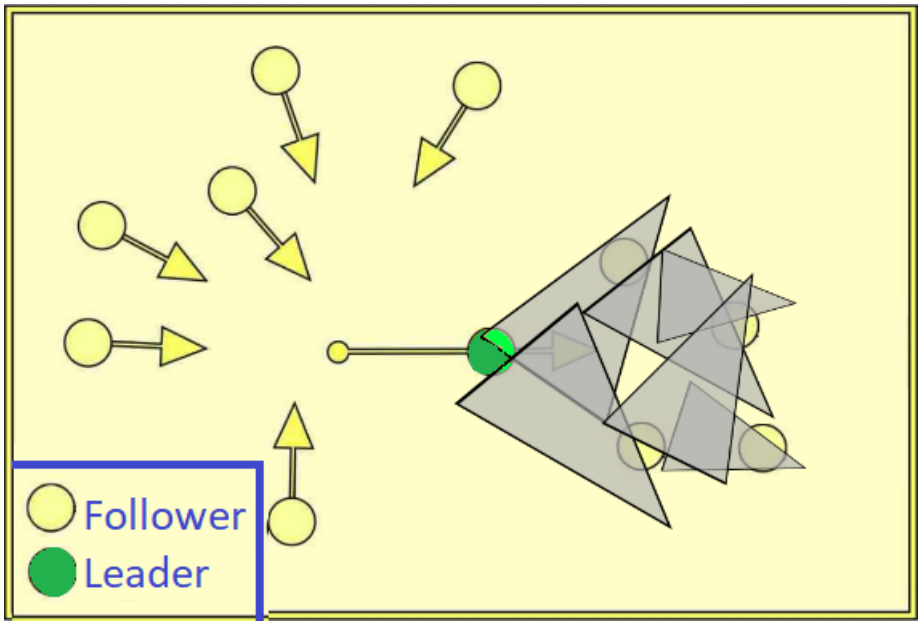
Gambar 3.6: *Diagram alir proses collision avoidance dengan penambahan paramater prioritas penghindaran*

Pada *flowchart* yang ditunjukkan pada gambar 3.6, terdapat penambahan proses pembobotan penghindaran yang notasikan dengan α. Sebagaimana telah diilustrasikan, $\alpha^{follower}_{leader}$ bernilai satu yang berarti bobot penghindaran yang dilakukan agen *follower* terhadap *leader* bernilai maksimal. Hal sebaliknya pada bobot $\alpha^{leader}_{follower}$ yang bernilai nol. Sehingga set dari valid *velocity* agen *leader* tidak dipengaruhi oleh agen *follower*-nya.

Perubahan pembobotan ini beserta pengubahan proses *steering behavior* yang telah dijelaskan pada subbab sebelumnya yaitu pada subbab 3.1.2, mengakibatkan *leader following behavior* sebelumnya yang diilustrasikan pada



(a) *leader following behavior*[1]



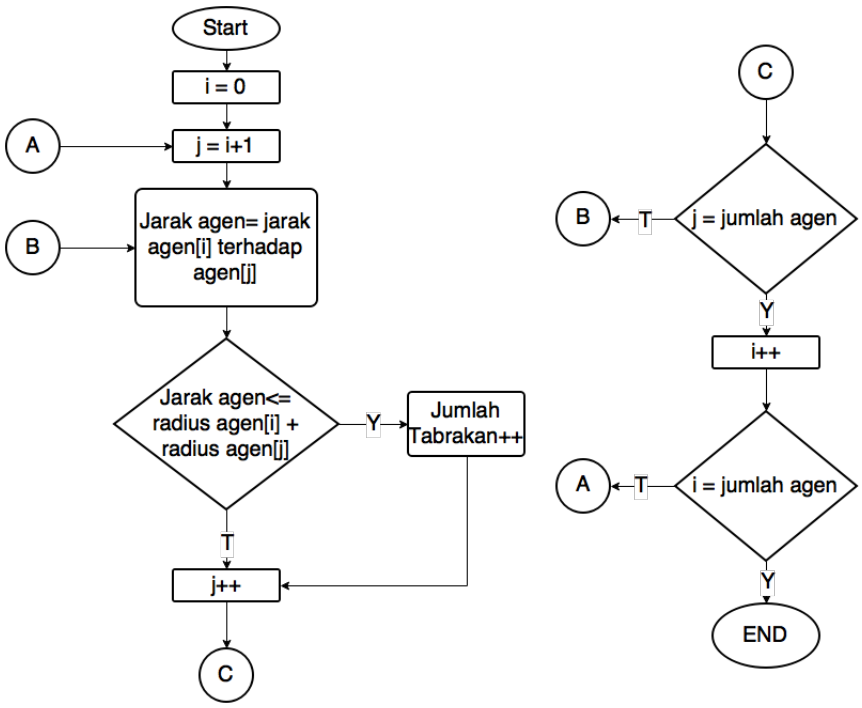
(b) *leader following behavior* setelah diubah

Gambar 3.7: Ilustrasi *leader following behavior* pada a.[10], b. *leader following behavior* setelah diubah dengan mengganti pembobotan prioritas penghindaran

gambar 3.7a berubah menjadi seperti yang diilustrasikan pada gambar 3.7b. Sebagaimana ditunjukkan pada ilustrasi tersebut, terdapat proses yang dihilangkan. Proses yang dihilangkan yaitu pengarahan agent *follower* untuk keluar dari area rektangular didepan *leader*-nya sehingga agen leader dapat leluasa bernavigasi. Hal ini karena proses ini telah digantikan dengan penghindaran

dengan pembobotan penghindaran penuh oleh agen *follower*.

3.1.4 *Collision detection*



Gambar 3.8: *Diagram alir tahap pengecekan terjadinya tabrakan*

Tahap terakhir setelah gerakan agen ditentukan adalah *collision detection* untuk mendeteksi terjadinya tabrakan. Tabrakan ini dideteksi dengan menggunakan *collider* yang ada pada unity. *Collider* yang dipakai adalah *capsule collider* dan bekerja sebagaimana algoritma deteksi tabrakan *circle to circle detection* yang ditunjukkan pada diagram alir di gambar 3.8. *Collider* ini akan mendeteksi tabrakan bila *collider* yang satu dengan yang lainnya terjadi kontak fisik. Karena *collider* bisa didekati sebagai lingkaran. Maka tabrakan terjadi bila jarak kedua agen lebih rendah atau sama dengan nilai jumlah radius keduanya.

Proses ini merupakan proses yang memvalidasi keberhasilan sistem navigasi agen dalam memperkirakan terjadinya tabrakan, dan juga merupakan ekstraksi data performansi pada simulasi kerumunan ini. Semakin rendah nilai tabrakan yang terjadi pada tiap simulasi yang dilakukan, semakin tinggi per-

formansi navigasi multiagen dengan teorema RVO dan *leader following steering behavior*.

3.2 Pengambilan data

Dalam simulasi ini ada data yang akan diambil dari *game engine* yang digunakan. Data ini dipakai untuk menganalisa performa dari simulasi kerumunan yang memiliki relasi *leader & follower* dengan penggunaan RVO sebagai navigasi multiagennya. Data tersebut adalah:

1. *Total frame*

   Merupakan data yang menunjukkan jumlah *frame* yang ada selama simulasi. Tiap *frame* menunjukkan *cycle* perhitungan pergerakan agen. Total *frame* simulasi didapat dengan menghitung pemanggilan fungsi *update per frame* yang selalu dipanggil pada Unity setiap *frame game* berjalan. data ini dipakai untuk menghitung nilai *frame rate*.

$$\Sigma Frame = \Sigma Function_{update()} \tag{3.4}$$

2. *Simulation time*

   Data ini merepresentasikan waktu yang diperlukan untuk melakukan simulasi. Simulasi berhenti ketika semua agen telah sampai pada titik tujuannya atau terjadi *deadlock* sehingga mengakibatkan agen yang belum mencapai titik tujuan tidak dapat melanjutkan perjalanan. Data *simulation time* didapatkan dengan mengambil data pewaktu internal Unity, dimana waktu simulasi disamakan dengan waktu berjalananya permainan.

$$\Sigma Simulation\ Time = \Sigma Time_{\Delta Time} \tag{3.5}$$

3. *Total collision*

   Sebagaimana telah disebutkan pada bagian 3.1.3 , terdapat data jumlah

terjadinya tabrakan agen selama simulasi. Dari data ini dapat dianalisis keefektifan RVO dalam penghindaran tabrakan pada berbagai kepadatan kerumunan. Data ini diambil dari akumulasi *collision* yang dideteksi oleh *collider* agen.

$$\Sigma Collision = \Sigma Function_{OnCollisionEnter()} \tag{3.6}$$

4. *Average frame rate*

   Merupakan nilai yang menunjukkan rerata dari jumlah *frame* persatuan detik. Didapatkan dengan membagi jumlah *frame* selama simulasi berjalan dengan *simulation time* dan memiliki satuan FPS(*Frame per-second*).

$$\widetilde{FPS} = \frac{\Sigma Frame}{\Sigma Simulation\ Time} \tag{3.7}$$

5. *Total loss agents*

   Merupakan data yang menunjukkan jumlah agen *follower* yang telah kehilangan arah, yang berarti agen tersebut tidak dapat menemukan leadernya.

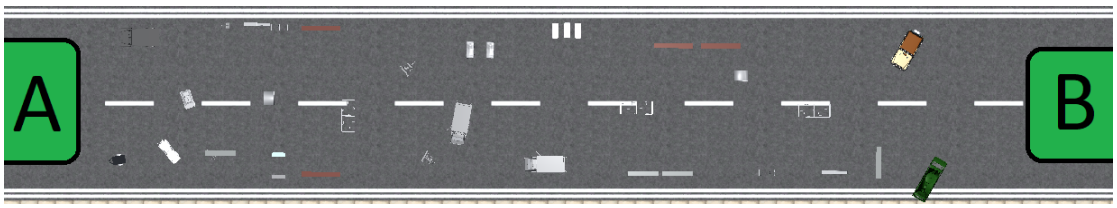
$$\Sigma Loss\ Agents = \Sigma Follower_{loss=true} \tag{3.8}$$

3.3 Desain Simulasi

Untuk mendapatkan data keefektifan navigasi multiagen menggunakan RVO pada simulasi kerumunan yang memiliki relasi *leader* & *follower*, maka perlu didesain skenario simulasi untuk menguji algoritma navigasi multiagen. Dalam penelitian ini dipakai salah satu skenario simulasi yang digunakan pada salah satu penelitian RVO [2]

Skenario ini merupakan cuplikan dari skenario tur pada penelitian [2], dimana bagian yang dicuplik adalah bagian dimana terdapat banyak rintangan

statis sebagaimana pada gambar3.9. Rintangan inilah yang menjadi masalah pada navigasi dengan relasi *leader-follower*. Masalah ini muncul karena agen *follower* tidak mampu mengosongkan area rektangular didepan *leader* yang mengakibatkan navigasi leader terganggu.



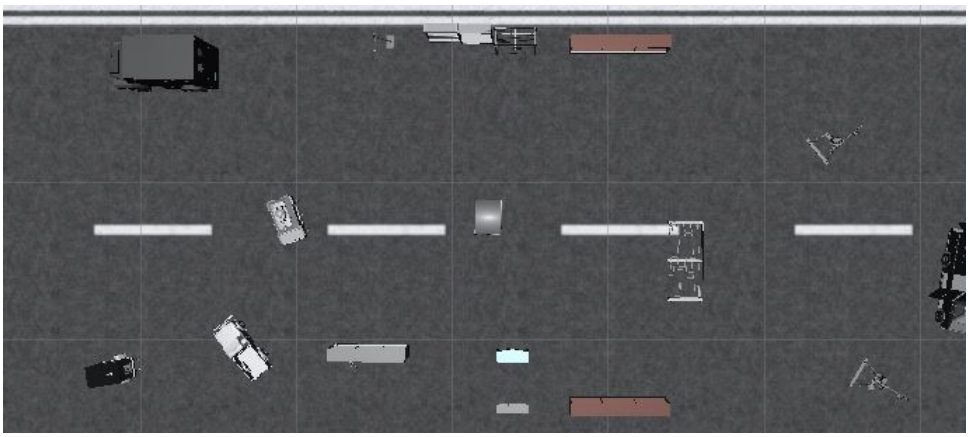
Gambar 3.9: Skenario simulasi

Seluruh lingkungan dibuat dengan kondisi yang sama pada semua skenario pengujian. Seluruh area navigasi lingkungan tidak memiliki variasi friksi yang dapat memengaruhi kelajuan agen. Tidak ada *environmental hazard* seperti hujan, badai dan semacamnya. Dengan begitu pengaruh lingkungan terhadap navigasi agen hanya pada pengaruh rintangan statis yang tersebar pada lingkungan simulasi.

Lingkungan simulasi kerumunan yang dibuat disesuaikan dengan skenario simulasi. Secara garis besar objek lingkungan simulasi kerumunan dihimpun menjadi dua jenis berdasarkan kemampuan agen bernavigasi yaitu *walkable* dan *unwalkable*. Sehingga perlu ada pendefinisian pada lingkungan simulasi bagian mana yang dapat dilalui agen dan tidak dapat dilalui untuk masing-masing scenario. area *unwalkable* yang dimaksud adalah area rintangan statis itu sendiri, dimana rintangan statis merupakan bagian yang menjadi penghalang gerakan agen simulasi

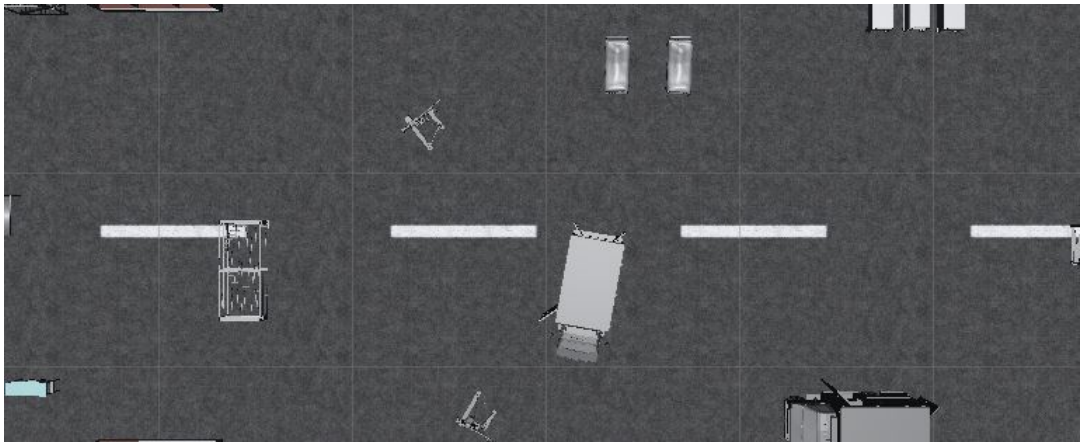
Simulasi pada lingkungan yang ditunjukkan pada gambar 3.9 menggunakan dua skenario macam skenario di lingkungan tersebut. Skenario pertama menskenariokan satu kerumunan. Agen *leader* akan muncul dari area A yang ditunjuk pada gambar 3.9 menuju area B kemudian kembali lagi ke area A. Sedangkan agen *follower* muncul dari area B dan bernavigasi ke arah area A hingga menemukan *leader*-nya jika berada dalam LoS agen *follower*. Skenario ini mengakibatkan agen *follower* akan berada didepan agen *follower* ketika

bertemu ditengah area, dan ketika agen *leader* berbalik arah ketika sampai pada area B dan kembali ke area A. Agen *follower* harus mampu keleluasaan navigasi sebisa mungkin pada *leader*-nya ditengah banyaknya rintangan statis pada skenario ini.



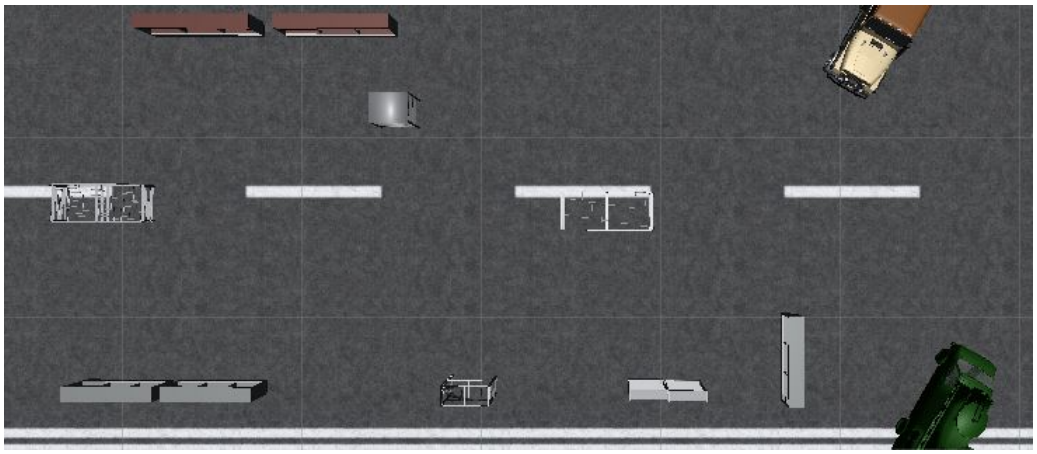
Gambar 3.10: Contoh rintangan statis pada lingkungan dekat area A simulasi

Skenario kedua juga dilakukan pada lingkungan yang sama sebagaimana skenario pertama, hanya saja terdapat dua kerumunan pada skenario ini. Masing-masing agen *leader* akan muncul dari area A dan area B yang ditunjuk pada gambar 3.9. Agen *leader* kelompok pertama yang muncul di area A akan menuju area B kemudian kembali lagi ke area A. Sedangkan agen *leader* kelompok kedua dimunculkan di area B akan menuju area A kemudian kembali lagi ke area B berlawanan dengan *leader* kelompok A. Adapun agen *follower* agen *follower*, masing-masing muncul dari area B dan area A. Agen *follower* kelompok pertama muncul dari area B dan bernavigasi ke arah area A hingga menemukan *leader*-nya jika berada dalam LoS agen *follower*. Sedangkan untuk agen *follower* kelompok kedua, mereka dimunculkan di area A dan bernavigasi ke arah area B hingga menemukan *leader*-nya jika berada dalam LoS agen *follower*. Skenario ini mengakibatkan agen *follower* akan berada didepan agen *follower* ketika bertemu ditengah area, dan ketika agen *leader* berbalik arah ketika sampai pada area B dan kembali ke area A. Agen *follower* harus mampu keleluasaan navigasi sebisa mungkin pada *leader*-nya ditengah banyaknya rintangan statis dan kelompok kerumunan lain pada skenario ini.



Gambar 3.11: Contoh rintangan statis yang berada pada area tengah lingkungan simulasi

Kedua skenario ini sekalipun di buat di tempat yang sama, masing-masing memiliki tantangannya masing-masing. Skenario pertama dimaksudkan untuk pengamatan pada perilaku agen *follower* ketika berada didepan agen *leader* mereka. Pada skenario pertama ini, tantangan yang dihadapi oleh navigasi multi-agen terdapat pada kemampuan memberikan keleluasan navigasi kepada agen *leader* sekalipun banyak rintangan yang menghadang agen *follower* dalam prosesnya memberi jalan kepada *leader* dan kembali mengikuti agen leader tersebut.



Gambar 3.12: Contoh rintangan statis pada lingkungan dekat area B simulasi

Pada skenario kedua memiliki kemiripan dengan skenario *crossroad* yang juga digunakan pada penelitian sebelumnya [2]. Hanya saja pada skenario yang digunakan pada penelitian ini, terdapat banyak rintangan statis, kemu-

dian agen *follower* masing masing kelompok ketika dimunculkan, tidak berada langsung dibelakang *leader*-nya. Pada skenario ini, tantangan yang dihadapi oleh navigasi multi-agen terdapat pada kemampuan memberikan keleluasan navigasi kepada agen *leader* sembari menghindari rintangan lain baik berupa gerakan agen dari kelompok lain, serta gerakan rintangan statis yang telah ada di lingkungan simulasi.

Halaman ini sengaja dikosongkan

BAB 4

HASIL DAN PEMBAHASAN

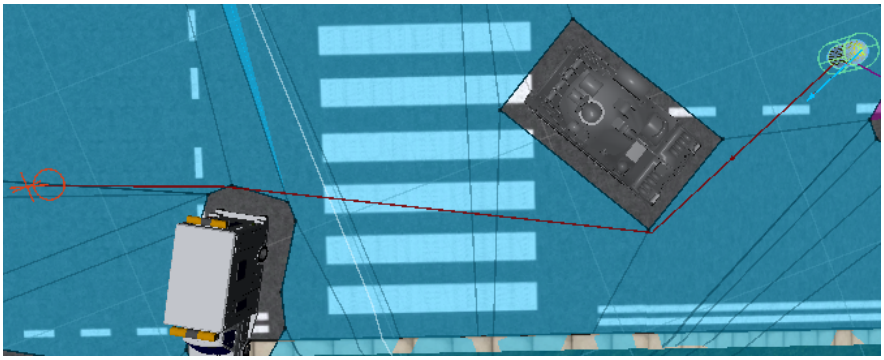
Pada bab ini akan dipaparkan hasil dan analisis dari simulasi yang dilakukan beserta data yang didapat. Pertama-tama akan dipaparkan hasil pengujian sistem navigasi serta ekstraksi data sebelum hasil dari skenario simulasi yang telah dilakukan.

4.1 Sistem Navigasi Agen

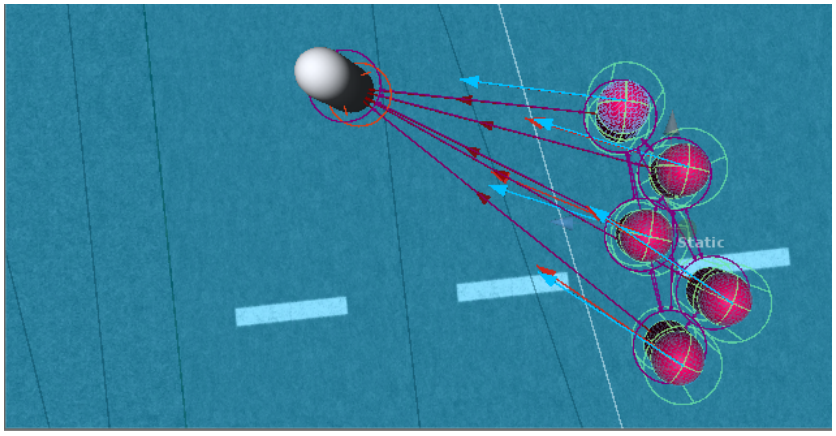
Pada bagian ini akan dipaparkan sistem navigasi agen yang telah dibuat yaitu sistem navigasi yang berdasarkan pada penelitian [2]. Pada bagian ini juga akan dilakukan pemaparan hasil dari sistem navigasi yang telah dibuat yaitu *seeking*, *steering*, *collision avoidance* dan *collision detection*. Kemudian dilanjutkan dengan pengujian ekstraksi data.

4.1.1 Proses *Seeking*

Proses pertama pada penentuan pergerakan agen disetiap siklusnya adalah *seeking*. Pada proses ini akan diperlihatkan kemampuan agen dalam mencari titik destinasinya. Pada gambar 4.1 diperlihatkan hasil dari proses *seeking* yang telah dibuat dimana 4.1(a) merupakan hasil *seeking* oleh agen *leader* 4.1(b) oleh agen *follower*. Agen *leader* dapat menemukan titik destinasinya yaitu rute yang telah ditentukan. Agen *follower* menemukan titik destinasinya yaitu agen *leader* yang harus dia ikuti. Jalur yang didapat ditunjukkan oleh garis merah. Jalur tersebut merupakan hasil algoritma A*. Hasil ini disimpan dalam bentuk *velocity preferensi* yang berisi arah serta jarak ke titik tujuan.



(a) Agen *leader*(putih)



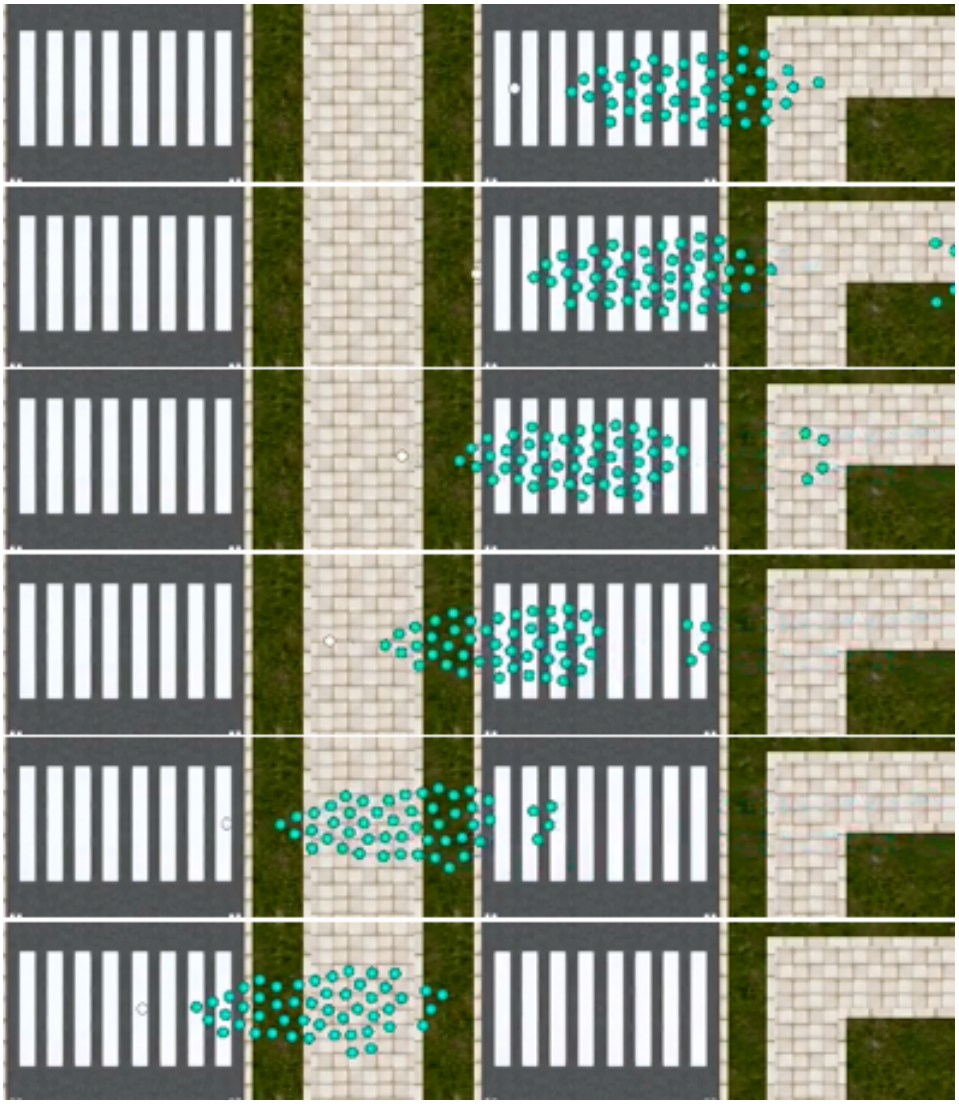
(b) Agen *follower* (merah)

Gambar 4.1: Hasil *seeking* agen *leader & follower*

4.1.2 Proses *Steering*

Setelah agen menentukan titik tujuannya serta mendapatkan *velocity preferensi* dari proses *seeking* maka proses selanjutnya yang diujikan adalah pengarahan agen menuju destinasinya yaitu *steering*. Proses pertama adalah pengecekan pemilihan kelajuan agen dalam pengejaran terhadap leader *leader*. Dapat dilihat pada gambar 4.2, terdapat empat agen *follower* yang tertinggal dari kelompoknya sehingga agen tersebut menambah kelajuan untuk mengejar ketertinggalannya.

Proses selanjutnya yang telah dibuat yaitu untuk tahapan *steering* adalah pengarahan agen *follower* untuk memberikan jalan kepada *leader*-nya bila agen *follower* berada di depan *leader*. Hasilnya ditunjukkan pada gambar 4.3 dimana agen *follower* diarahkan keluar dari area *rectangular* yang berada didepan *leader*. Parameter *velocity* hasil dari *steering* ini disimpan sebagai

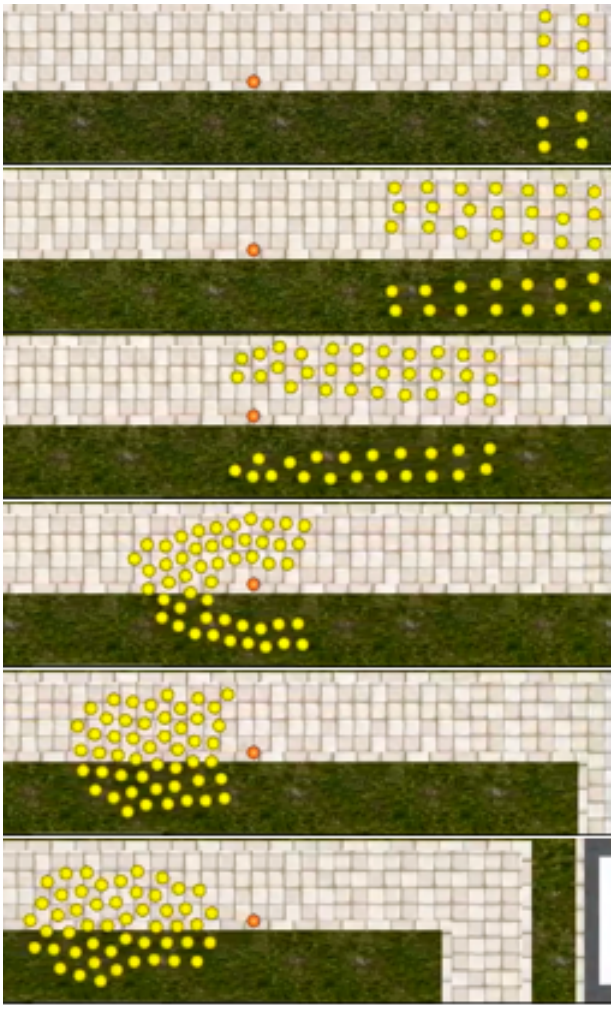


Gambar 4.2: Pengejaran oleh agen *follower* yang tertinggal

velocity referensi.

4.1.3 Proses *Collision Avoidance*

Pada tahap ini, agen simulasi yang telah mendapatkan *velocity* referensinya akan melakukan proses pengecekan validitas dari *velocity* referensi tersebut dengan algoritma RVO. Sistem yang telah dibuat ini diujikan dengan menggerakkan dua agen simulasi yang memiliki arah gerak berlawanan sehingga akan mengakibatkan tabrakan bila kedua agen tersebut bergerak dengan tetap menggunakan *velocity* referensi. Proses navigasi penghindaran agen dengan RVO akan mengubah *velocity* agen diluar *velocity obstacle* yang mengakibat-



Gambar 4.3: *Steering* agen *follower* yang berada di depan *leader*-nya

kan tabrakan. Bila penghindaran terjadi antara agen dengan tingkat prio-

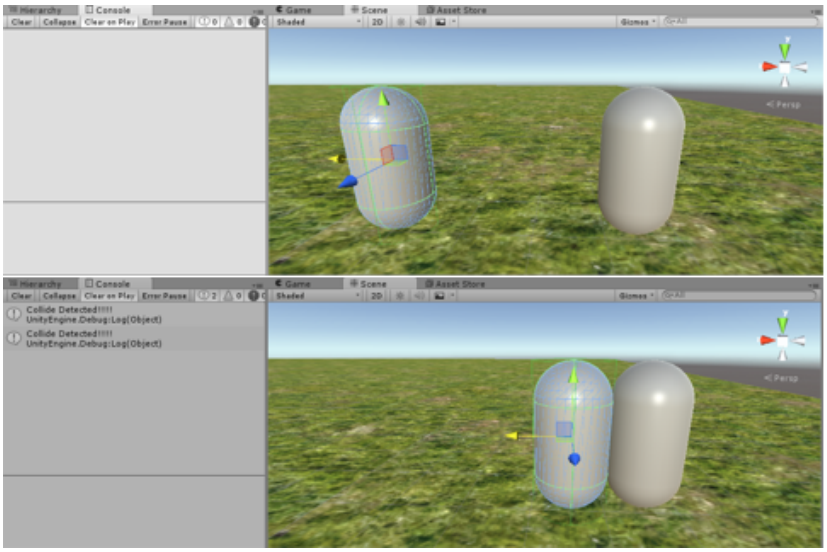
Gambar 4.4: Hasil penghindaran tabrakan dua agen dengan tingkat prioritas yang sama

ritas yang sama, maka bobot penghindaran terbagi antara agen tersebut dan menghasilkan gerak penghindaran seperti pada gambar 4.4. Sedangkan pada

gambar 4.5 adalah penghindaran antara agend dengan tingkat prioritas berbeda. Perbedaan yang diperlihatkan dari garis jejak pergerakan kedua agen menunjukan bahwa agen dengan prioritas yang lebih tinggi tidak melakukan penghindaran karena bobot penghindaran diserahkan seluruhnya pada agen dengan tingkat prioritas lebih rendah.

Gambar 4.5: Hasil penghindaran tabrakan dua agen dengan tingkat prioritas yang berbeda

4.1.4 Proses *Collision Detection*



Gambar 4.6: Hasil pengujian deteksi tabrakan

Collision detection merupakan proses terakhir pada satu siklus kalkulasi pergerakan agen. Pada proses ini akan dilakukan pengujian deteksi tabrakan oleh *collider* yang ditanam pada tubuh agen. Pada gambar 4.6 kedua agen yang awalnya tidak bertabrakan di translasikan sehingga menabrak satu sama lain. Ketika terjadi tabrakan, akan muncul *log* peringatan "*collide*

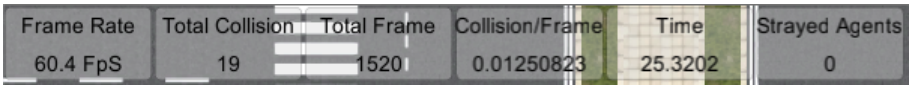
detected!!!!!" yang berarti kedua agen mendeteksi terjadinya tabrakan.

4.2 Sistem ekstraksi data

Tabel 4.1: Hasil pengujian fungsi pengambilan data

No.	Nama data	Fungsi dapat berjalan
1	Jumlah *frame*	Ya
2	Jumlah *tabrakan*	Ya
3	Waktu simulasi	Ya
4	*Collision per frame*	Ya
5	Rata-rata *frame rate*	Ya
6	Jumlah agen tersesat	Ya

Sistem ini dibuat dengan maksud dan tujuan untuk mendapatkan data dari simulasi. Data hasil ekstraksi inilah yang akan digunakan sebagai bahan analisa dalam penelitian nantinya. Terdapat enam macam data sebagaimana ditunjukkan pada tabel 4.1. Pada tabel 4.1 tersebut juga ditunjukkan hasil pengujian untuk memeriksa sistem pengambilan data dan memastikannya telah berjalan sesuai yang diinginkan. Pengujian ini dilakukan dengan mencoba seluruh fungsi dalam ekstraksi data dan memastikan apakah telah berjalan dengan baik atau tidak. Data yang di ekstraksi ditampilkan pada *user interface* simulasi seperti pada gambar 4.7. Hasil yang didapatkan dari pengujian kesesuaian sistem permainan ini terdapat pada Tabel 4.1.



Gambar 4.7: Contoh UI ekstraksi data. Informasi yang ditampilkan dari kiri ke kanan: *frame rate*,jumlah tabrakan, jumlah frame, *collision per-frame*, waktu simulasi, jumlah agen tersesat

4.3 Hasil Simulasi

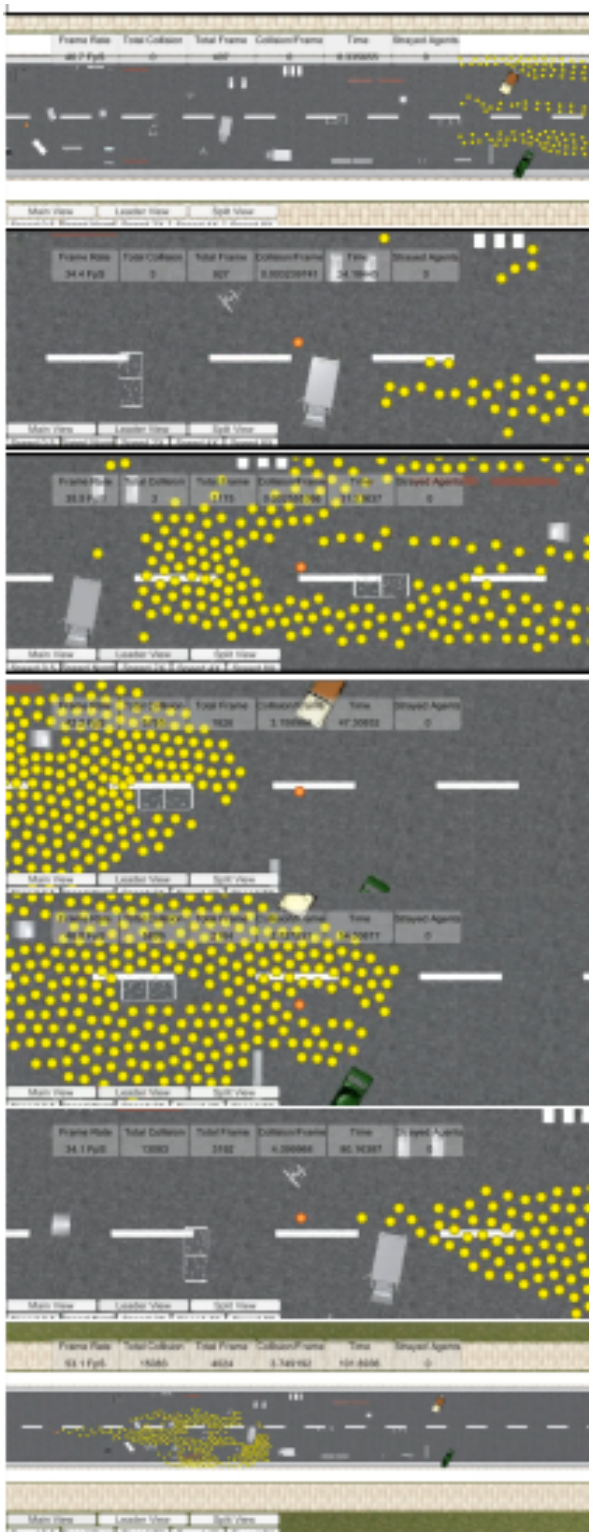
Pada bagian ini akan ditampilkan data yang didapat dari simulasi kerumunan pada skenario simulasi yang telah dijelaskan pada bagian 3.3. Simulasi dijalankan pada komputer dengan spesifikasi prosesor Intel Core i5-3470 3.20

GHZ dengan RAM sebesar 8 GB serta GPU Radeon HD7700 VRAM 1GB. Setiap skenario simulasi dijalankan dengan jumlah agen bervariasi dari 25 hingga 500 agen. Setiap skenario dijalankan dengan dua macam kondisi yaitu dengan menggunakan perbedaan tingkat prioritas antara *leader* dan *follower* dan dengan tingkat prioritas yang sama. Dari kedua hasil ini didapatkan data keefektifan Navigasi dengan RVO yang memakai tingkat prioritas dengan tanpa tingkat prioritas.

Parameter performansi yang didapat dari simulasi adalah waktu simulasi dan jumlah tabrakan. Waktu simulasi menunjukkan kemampuan navigasi dalam membawa kerumunan menuju tujuannya. Semakin sedikit waktu yang dibutuhkan menunjukkan navigasi kerumunan dapat mengatasi tantangan yang ada pada simulasi seperti rintangan statis, serta kondisi gerakan agen *follower* yang tidak mengganggu navigasi agen *leader*. Secara umum pada setiap skenario simulasi, akan terjadi tabrakan pada tingkat keramaian tertentu. Tabrakan ini diakibatkan karena tingkat kerumunan yang tinggi mengakibatkan kalkulasi perhitungan yang lebih kompleks. Hal ini terjadi akibat semakin banyaknya agen yang harus ditentukan pergerakannya secara lokal tanpa ada koordinasi. Selain itu pada penggunaan relasi *leader & follower*, gerak agen *follower* semakin terikat pada gerakan *leader*-nya. Perhitungan gerak agen dilakukan pada tiap *frame*. Sehingga bila pada suatu *frame* telah ditentukan *velocity* seluruh agen, maka perubahan *velocity* selanjutnya harus menunggu *frame* selanjutnya. Bila sebelum *frame* selanjutnya tercapai, *velocity* agen berada dalam *collision cone*, agen bisa bertabrakan. Hal ini diakibatkan belum ada perubahan pada *velocity* agen dikarenakan *frame* selanjutnya belum tercapai.

4.3.1 Skenario Pertama

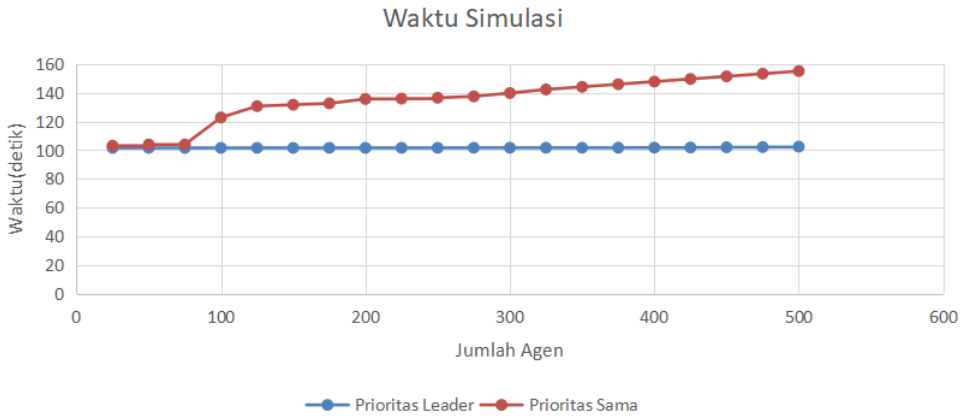
Skenario simulasi pertama melibatkan simulasi satu kerumunan dengan agen *follower* dan *leader* ditempatkan di titik kemunculan yang berbeda sebagaimana telah dijelaskan lebih detail pada subbab 3.3. Gambar 4.8 menunjukkan cuplikan dari simulasi kerumunan, data dari hasil kerumunan tersebut



Gambar 4.8: Salah satu cuplikan simulasi skenario pertama

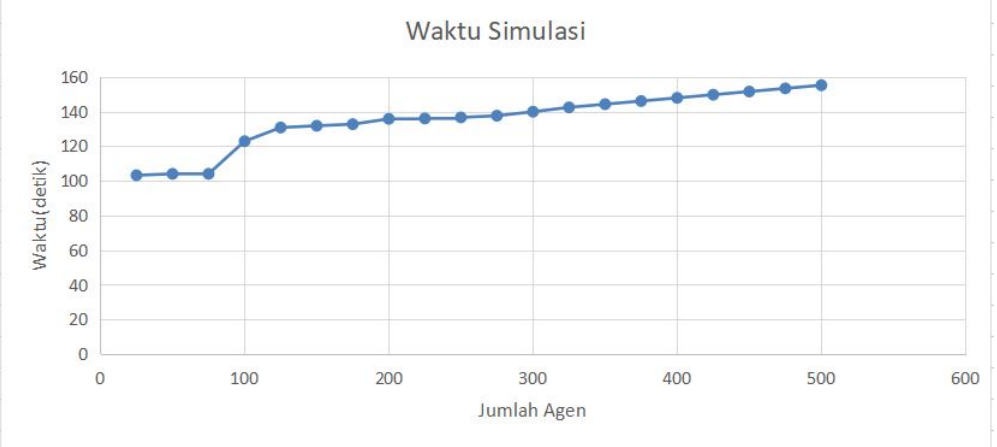
disajikan pada grafik digambar 4.9 dan 4.15. Dari data yang didapat, terjadi peningkatan tabrakan pada simulasi baik dengan penggunaan tingkat prioritas

penghindaran maupun dengan yang tidak menggunakan.

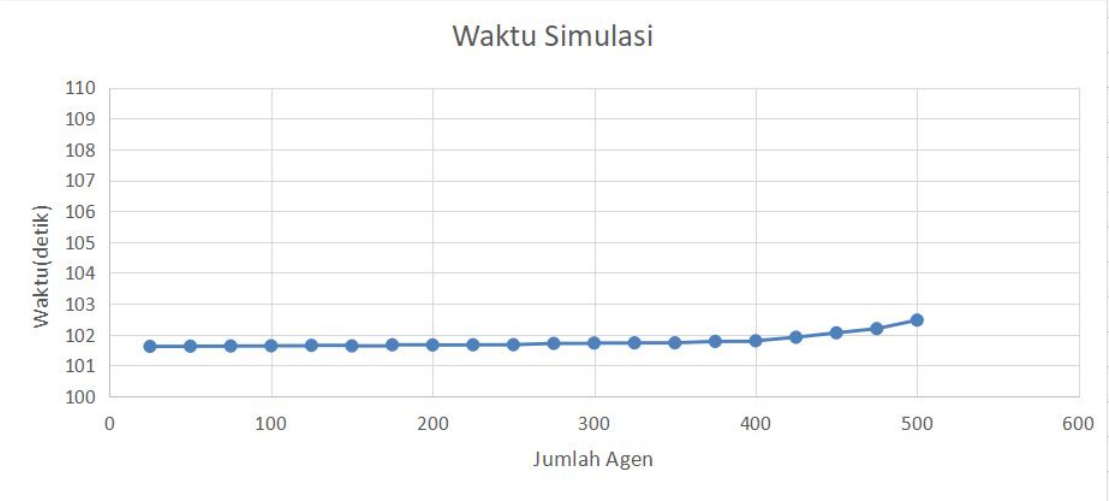


Gambar 4.9: Grafik perbandingan waktu simulasi skenario pertama

Grafik waktu simulasi pada gambar 4.9 memperlihatkan kenaikan waktu yang dibutuhkan untuk agen *leader* menuju titik tujuannya. Data pada grafik 4.10 menunjukkan simulasi dengan prioritas sama, dimana simulasi ini menghasilkan kenaikan waktu yang cukup signifikan dengan rata-rata kenaikan 50,43%. Pada penggunaan pembedaan prioritas, kenaikan hanya terjadi dengan rata-rata kenaikan 0,84% dengan rerata waktu yang dibutuhkan sebanyak 101,78 detik. kestabilan ini dikarenakan kasus *deadlock* yang terjadi pada simulasi dengan prioritas yang tidak dibedakan seperti ditunjukkan pada gambar 4.12.



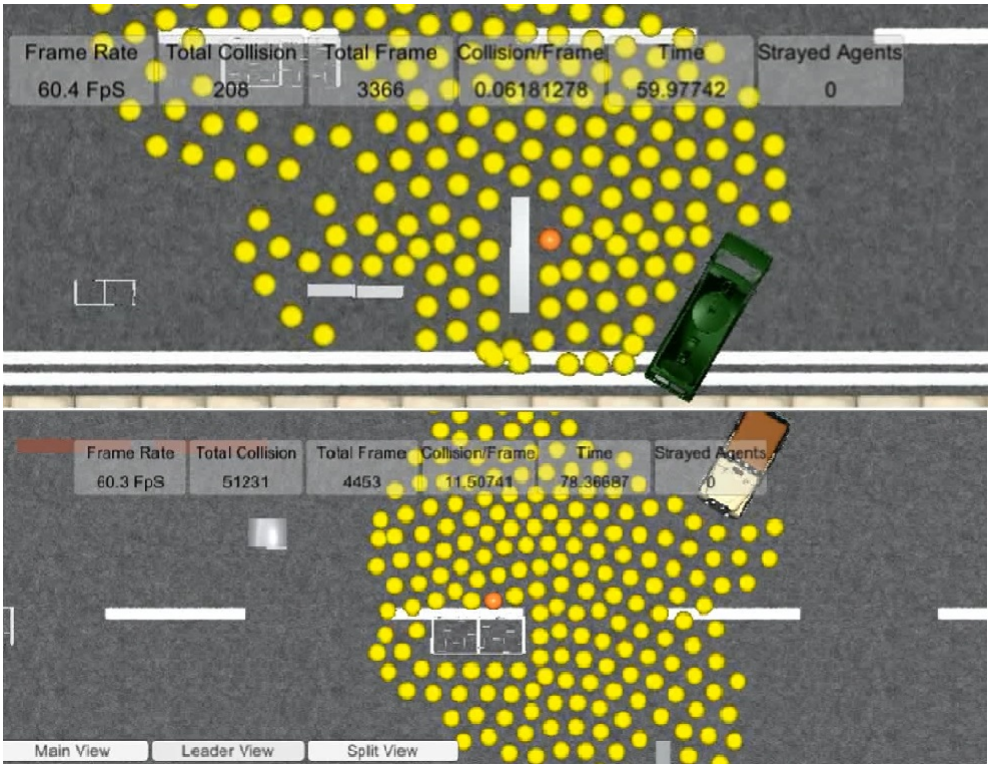
Gambar 4.10: Grafik waktu simulasi skenario pertama dengan prioritas yang sama



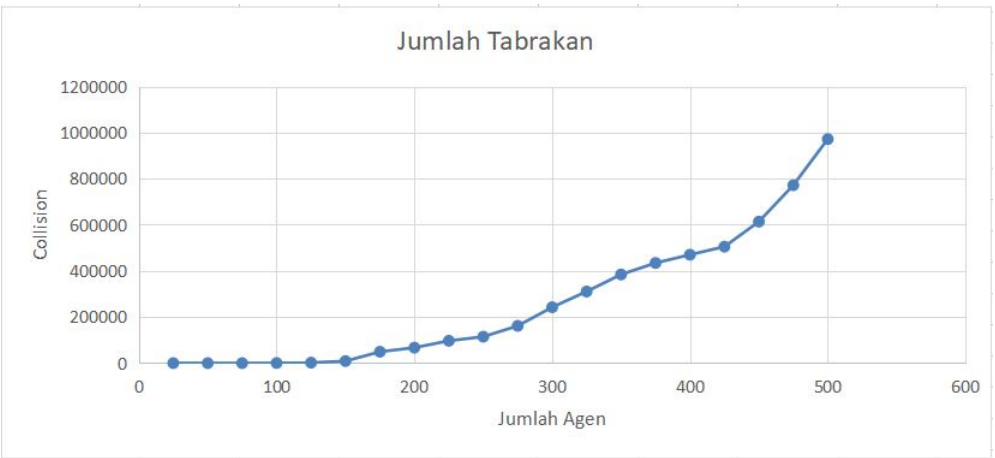
Gambar 4.11: Grafik waktu simulasi skenario pertama dengan prioritas yang sama

Deadlock ini terjadi akibat agen *follower* tidak mampu memberikan jalan ke agen *leader* dengan mengosongkan area rektangular didepan agen *leader*. Hal ini diakibatkan rintangan statis pada simulasi menyebabkan agen *follower* yang mengarahkan diri ke arah vektor keluar dari area rektangular tersebut terhalang rintangan statis dan pada saat yang sama terdapat agen *leader* yang melakukan penghindaran tabrakan terhadap agen-agen *follower* yang belum mengosongkan area rektangular didepan *leader*.

Kondisi tersebut juga mengakibatkan jumlah tabrakan yang terjadi meningkat sebagaimana pada gambar 4.15. Grafik pada gambar 4.13 menunjukkan data jumlah tabrakan yang terjadi pada skenario pertama tanpa perbedaan prioritas, sedangkan data dengan penggunaan prioritas ditunjukkan pada gambar 4.14. Pola kenaikan tabrakan seiring pertambahan jumlah agen memiliki kemiripan, namun penggunaan tingkat prioritas yang berbeda pada simulasi mengakibatkan penurunan jumlah tabrakan. Rerata Penurunan jumlah tabrakan yang terjadi dengan penggunaan tingkat prioritas sejumlah 23,62%. Tabrakan terjadi mulai dari jumlah agen sebanyak 75 agen untuk simulasi dengan prioritas yang sama, dan 100 agen pada simulasi dengan pembedaan prioritas. Penurunan ini dikarenakan tidak ada *deadlock* pada simulasi dengan tingkat prioritas penghindaran yang berbeda. Sedangkan pada kejadian *deadlock* pa-



Gambar 4.12: Contoh kejadian *deadlock* yang terjadi pada simulasi

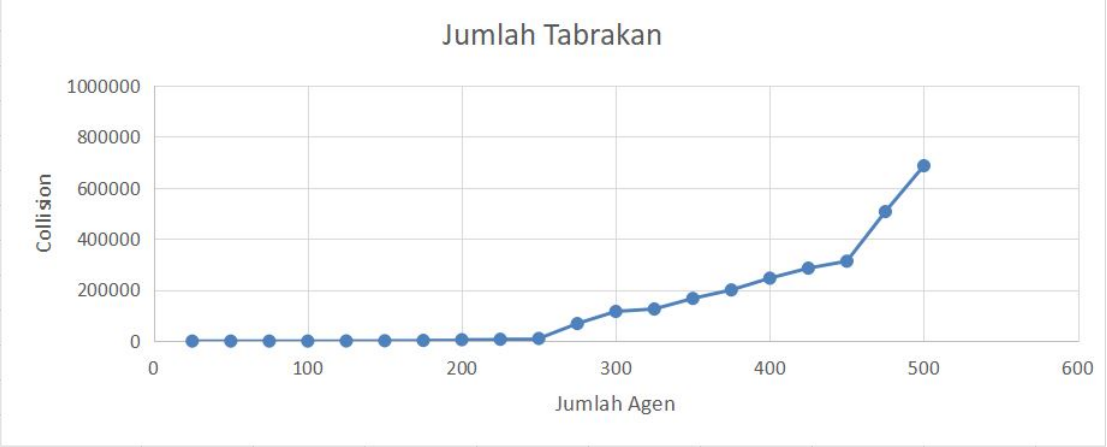


Gambar 4.13: Jumlah tabrakan yang terjadi pada skenario pertama dengan tingkat prioritas yang sama

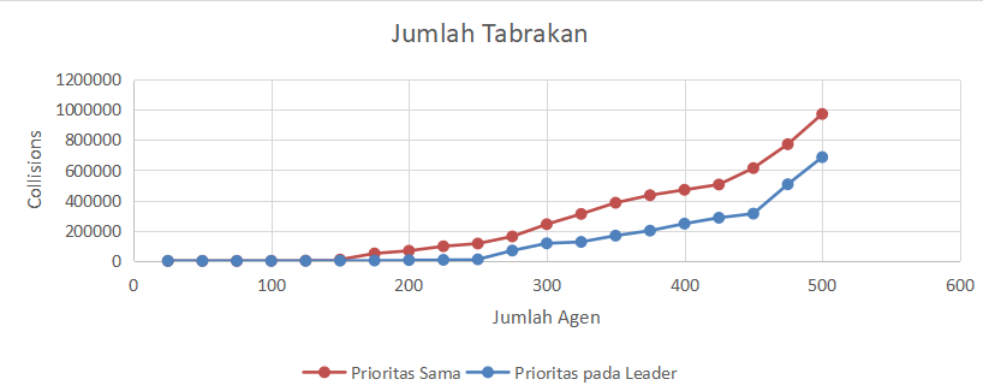
da simulasi dengan tingkat prioritas penghindaran sama, terjadi tabrakan di kejadian *deadlock* tersebut karena terganggunya navigasi agen *leader*.

4.3.2 Skenario Kedua

Skenario simulasi pertama melibatkan simulasi satu kerumunan dengan agen *follower* dan *leader* ditempatkan di titik kemunculan yang berbeda seba-



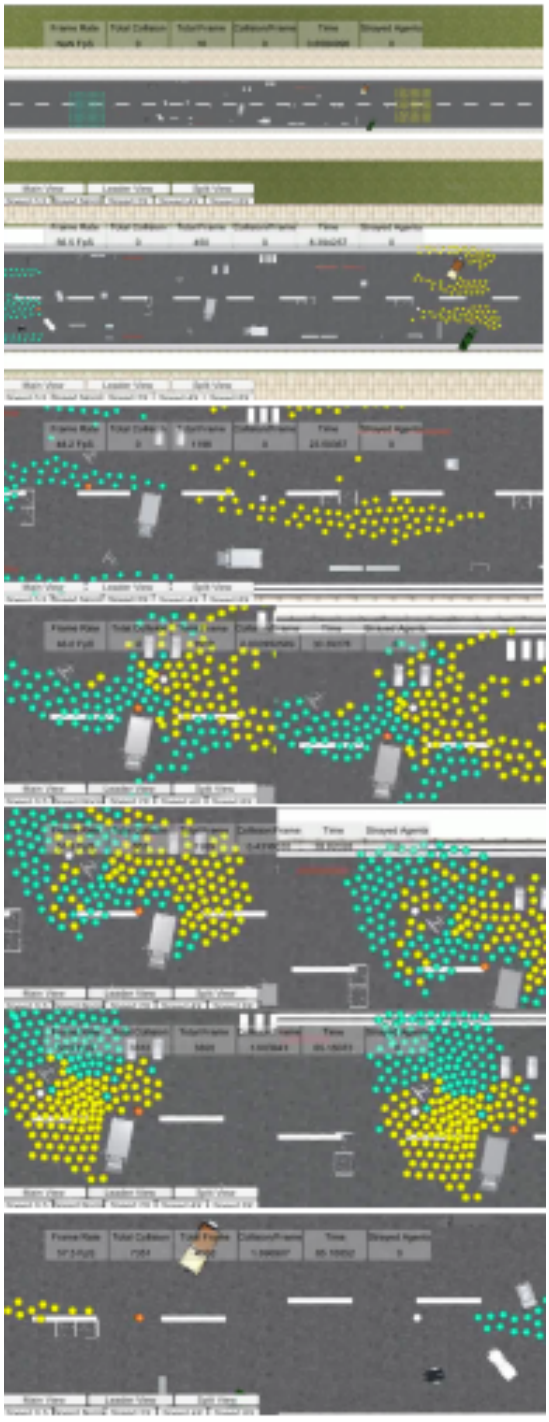
Gambar 4.14: Jumlah tabrakan yang terjadi pada skenario pertama dengan tingkat prioritas yang berbeda



Gambar 4.15: Grafik perbandingan jumlah tabrakan yang terjadi pada simulasi skenario pertama

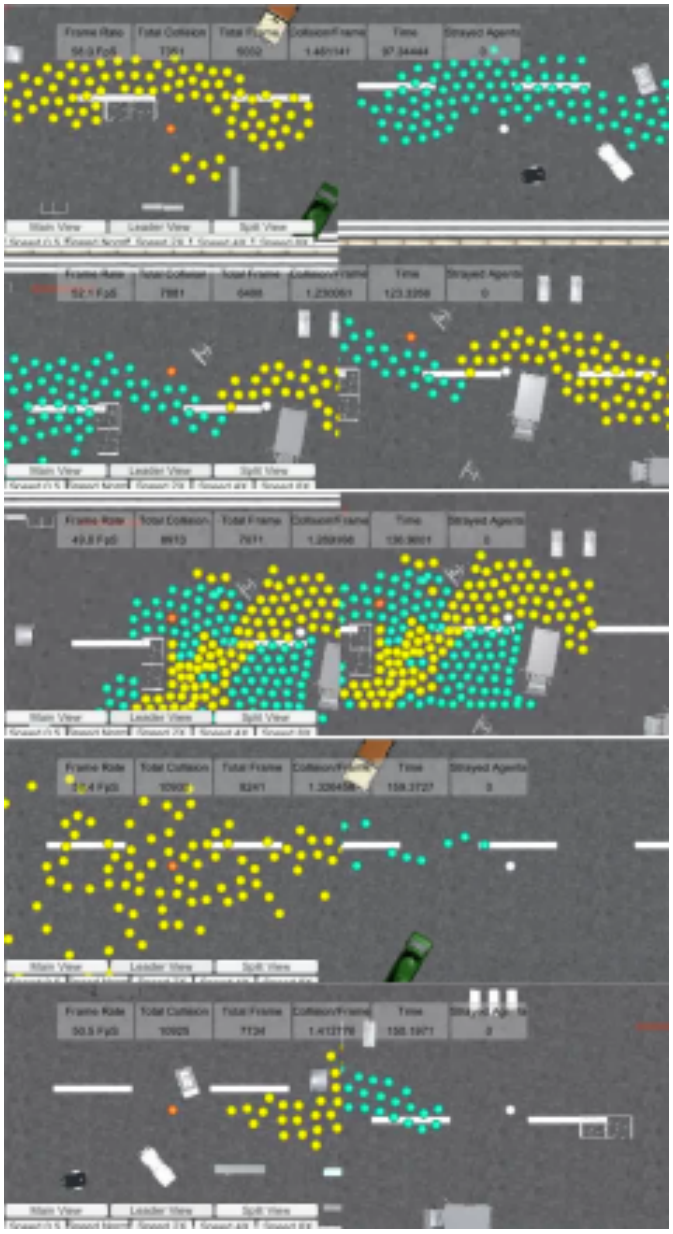
gaimana telah dijelaskan lebih detail pada subbab 3.3. Gambar 4.16 dan 4.17 menunjukkan cuplikan dari simulasi kerumunan skenario kedua. Sebagaimana ditunjukkan pada gamber tersebut, perbedaan yang terjadi pada skenario kedua ini, bila dibandingkan skenario pertama, yaitu adanya kerumunan lain yang saling bertemu ditengah-tengah rintangan statis di lingkungan simulasi. Simulasi skenario kedua ini menghasilkan data yang disajikan pada grafik digambar 4.18 dan 4.24. Dari data yang didapat, terjadi peningkatan tabrakan pada simulasi baik dengan penggunaan tingkat prioritas penghindaran maupun dengan yang tidak menggunakan sebagaimana dengan skenario sebelumnya.

Grafik waktu simulasi pada gambar 4.18 menunjukkan adanya kenaikan waktu yang dibutuhkan untuk agen *leader* menuju titik tujuannya. Data pada



Gambar 4.16: Salah satu cuplikan simulasi skenario kedua

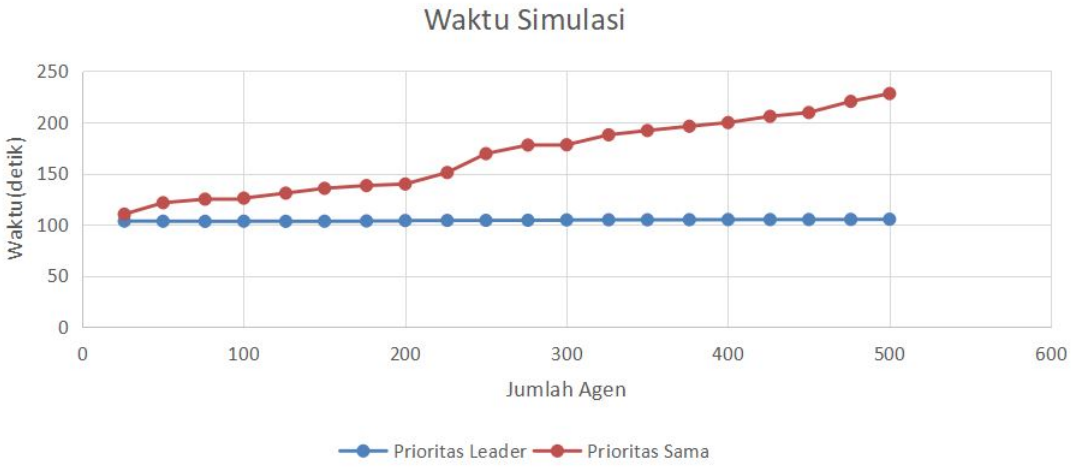
grafik 4.19 menunjukkan simulasi dengan prioritas sama, sedangkan waktu untuk simulasi dengan tingkat prioritas berbeda ditunjukkan pada gambar 4.20. Perbandingan grafik tersebut seperti pada gambar 4.18 menunjukkan penggunaan tingkat prioritas pada skenario ini mengakibatkan penurunan waktu.



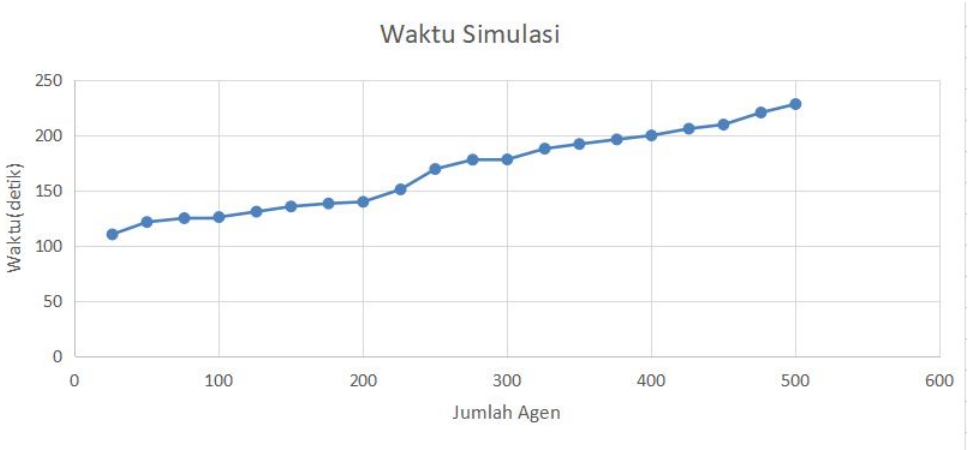
Gambar 4.17: Cuplikan lain simulasi skenario kedua

Penurunan waktu ini diakibatkan karena tidak terjadinya kejadian *deadlock* seperti pada simulasi dengan skenario yang tidak memiliki pembedaan tingkat prioritas penghindaran antar agen.

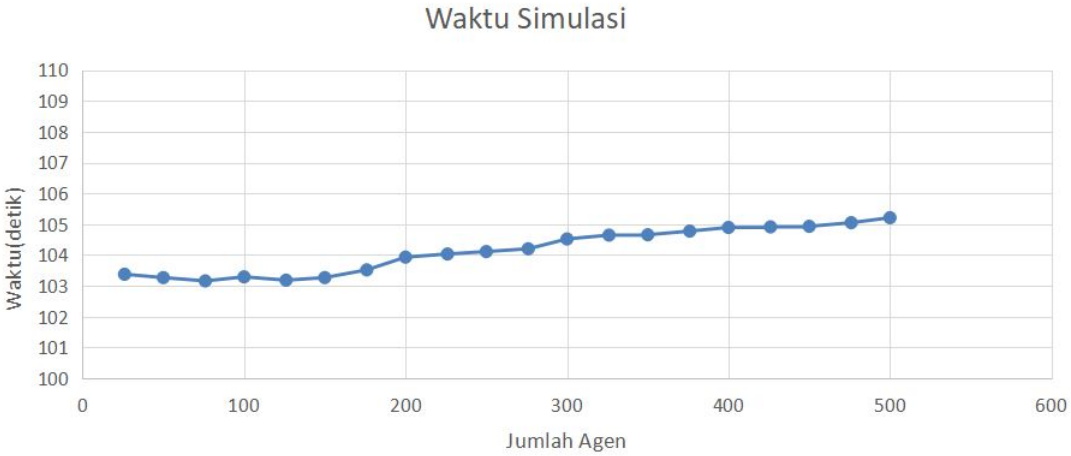
Deadlock ini terjadi akibat agen *follower* tidak mampu memberikan jalan ke agen *leader* dengan mengosongkan area rektangular didepan agen *leader*, sebagaimana pada skenario pertama. Namun perbedaan dengan skenario pertama, penyebab deadlock ini tidak hanya diakibatkan rintangan statis pada



Gambar 4.18: Grafik perbandingan waktu simulasi skenario pertama

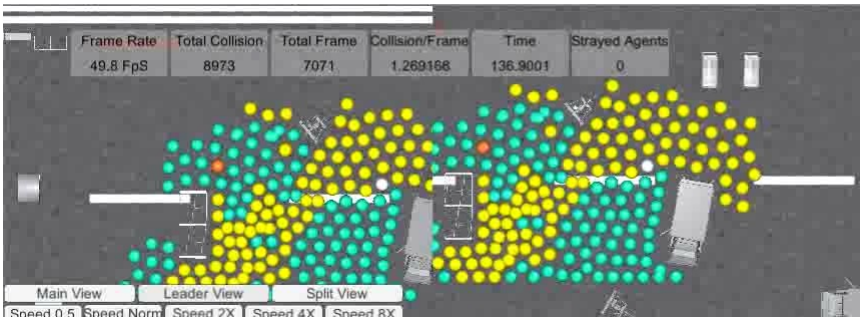


Gambar 4.19: Grafik waktu simulasi skenario pertama dengan prioritas yang sama

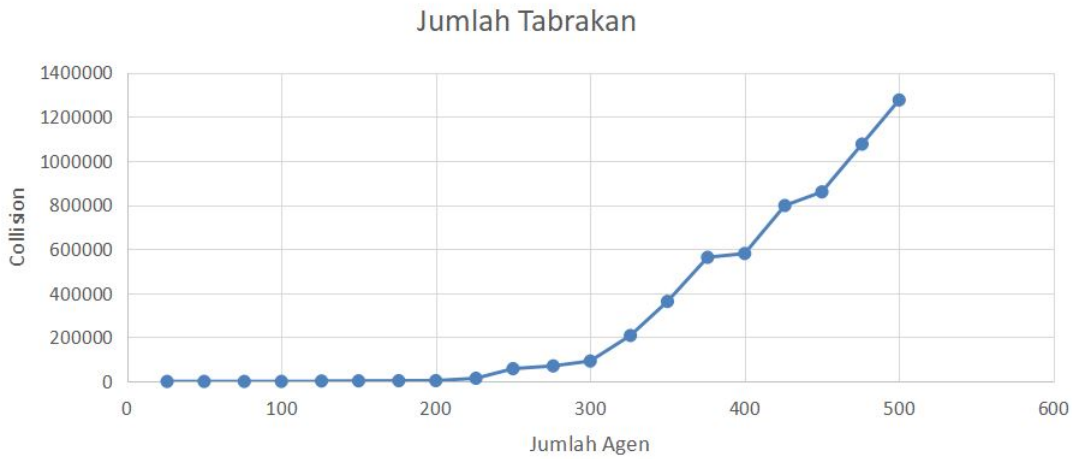


Gambar 4.20: Grafik waktu simulasi skenario pertama dengan prioritas yang sama

simulasi, tetapi juga gerakan kelompok agen lain karena pada skenario ini terdapat dua kelompok kerumunan. Rintangan statis pada lingkungan simulasi menyebabkan agen *follower* yang mengarahkan diri ke arah vektor keluar dari area rektangular tersebut terhalang rintangan statis dan pada saat yang sama terdapat agen *leader* yang melakukan penghindaran tabrakan terhadap agen-agen *follower*, baik itu agen *follower* kelompok sendiri yang belum mengosongkan area rektangular didepan *leader*, dan juga agen *follower* dari kelompok lain

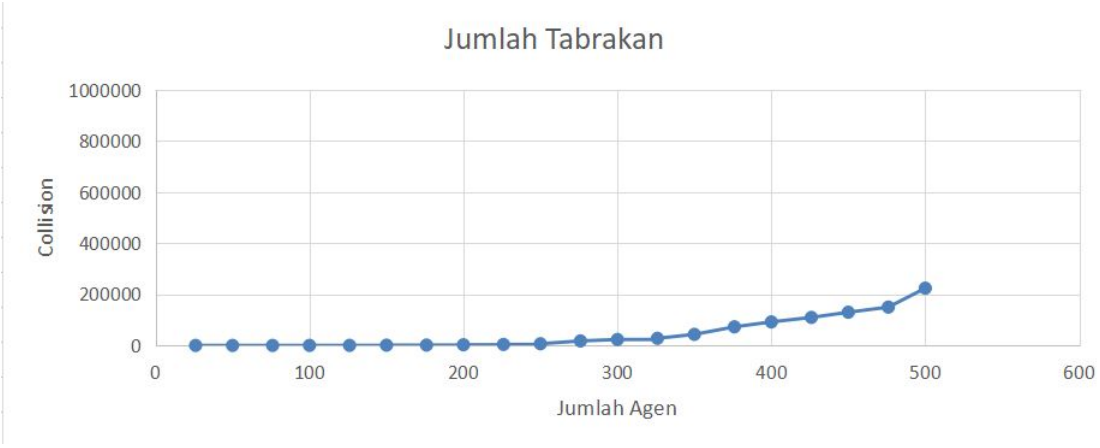


Gambar 4.21: Contoh kejadian *deadlock* yang terjadi pada simulasi skenario kedua



Gambar 4.22: Jumlah tabrakan yang terjadi pada skenario kedua dengan tingkat prioritas yang sama

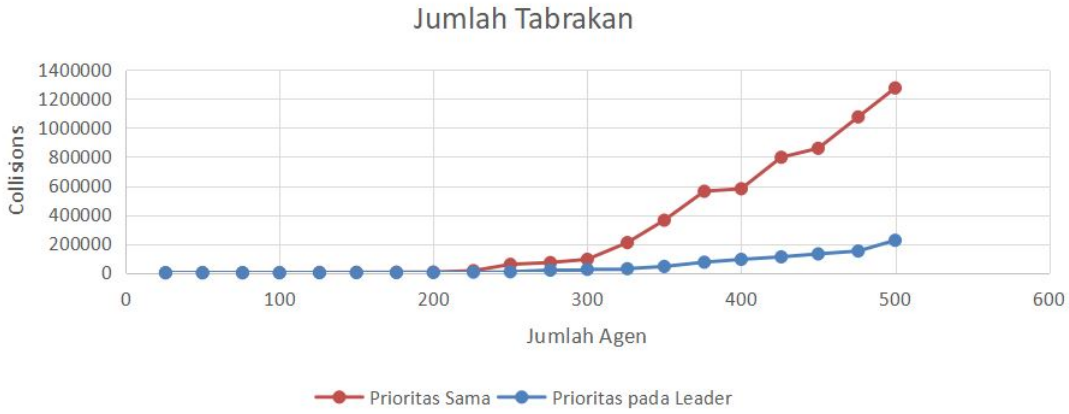
Deadlock yang terjadi tersebut juga mengakibatkan jumlah tabrakan yang terjadi meningkat sebagaimana pada gambar 4.24. Grafik pada gambar 4.22 menunjukkan data jumlah tabrakan yang terjadi pada skenario kedua tanpa



Gambar 4.23: Jumlah tabrakan yang terjadi pada skenario kedua dengan tingkat prioritas yang berbeda

perbedaan prioritas, sedangkan data dengan penggunaan prioritas untuk skenario kedua ditunjukkan pada gambar 4.23. Kenaikan tabrakan yang terjadi mirip dengan skenario pertama dimana jumlah tabrakan bertambah seiring pertambahan jumlah agen. Penggunaan tingkat prioritas yang berbeda pada simulasi mengakibatkan penurunan jumlah tabrakan. Rerata Penurunan jumlah tabrakan yang terjadi dengan penggunaan tingkat prioritas sejumlah 34,65%. Pada skenariokedua ini, tabrakan terjadi mulai dari jumlah agen sebanyak 50 agen untuk simulasi dengan prioritas yang sama, dan 75 agen pada simulasi dengan pembedaan prioritas. Penyebab penurunan ini sebagaimana pada skenario pertama, dikarenakan tidak ada *deadlock* pada simulasi dengan tingkat prioritas penghindaran yang berbeda. Sedangkan pada kejadian *deadlock* pada simulasi dengan tingkat prioritas penghindaran sama, terjadi tabrakan di kejadian *deadlock* tersebut karena terganggunya navigasi agen *leader*.

Pada kedua skenario, penambahan jumlah agen mengakibatkan komputasi yang lebih berat diakibatkan banyaknya gerakan navigasi agen yang perlu dikalkulasi. Grafik pada gambar 4.25 dan 4.26 memperlihatkan penurunan nilai *frame rate* yang memiliki perbedaan pada kecendrungan penurunan. Pada skenario pertama di gambar 4.25, nilai *frame rate* pada simulasi dengan metode pembedaan tingkat prioritas memiliki nilai lebih rendah dibandingkan



Gambar 4.24: Grafik perbandingan jumlah tabrakan yang terjadi pada simulasi skenario kedua

dengan metode sebelumnya. Hal ini mengisyaratkan beban komputasi yang lebih berat pada simulasi dengan pembedaan prioritas. Namun seiring banyaknya jumlah tabrakan yang terjadi akibat bertambahnya jumlah agen pada simulasi, komputasi pendeteksian tabrakan juga mengakibatkan efek yang membuat simulasi dengan tidak memakai pembedaan tingkat prioritas mengalami penurunan *frame rate*. Penurunan ini membuat nilai selisih *frame rate* antara kedua metode berkurang pada jumlah 400 hingga 475 agen. Ketika Simulasi skenario pertama mencapai 500 agen, nilai *frame rate* simulasi dengan tidak menggunakan prioritas pembedaan berada dibawah nilai simulasi dengan adanya pembedaan tingkat prioritas antara agen *leader* dan *follower*

Gambar 4.25: Grafik penurunan *frame rate* simulasi skenario pertama

Pada skenario Kedua di gambar 4.26, nilai *frame rate* pada simulasi de-

ngan metode pembedaan tingkat prioritas memiliki selisih nilai yang tidak terlalu terpaut jauh hingga jumlah agen sebanyak 275 bila dengan metode sebelumnya. Namun pada jumlah agen sebanyak 300 hingga 500 agen, terlihat bahwa nilai *frame rate* simulasi dengan tingkat prioritas yang sama lebih rendah dibandingkan simulasi dengan tingkat prioritas yang dibedakan. Hal ini terjadi karena jumlah tabrakan yang terjadi pada skenario ini secara garis besar lebih banyak dibandingkan skenario pertama akibat adanya dua kelompok pada simulasi ini. Hal ini mengisyaratkan sekalipun beban komputasi yang lebih berat pada simulasi dengan pembedaan prioritas. Dengan berkurangannya jumlah tabrakan yang terjadi, beban komputasi keseluruhan menjadi berkurang. Sehingga seiring banyaknya jumlah tabrakan yang terjadi akibat bertambahnya jumlah agen pada simulasi, komputasi pendeteksian tabrakan menjadi lebih rendah karena terjadinya tabrakan lebih sedikit pada pengunaan tingkat prioritas penghindaran.

Gambar 4.26: Grafik penurunan *frame rate* simulasi skenario kedua

Halaman ini sengaja dikosongkan

BAB 5

Kesimpulan

Dari hasil pengujian RVO pada simulasi kerumunan dengan relasi *leader & follower* dengan tingkat prioritas penghindaran yang sama dan yang dibedakan,dapat ditarik beberapa kesimpulan sebagai berikut :

1. Hasil simulasi RVO berelasi *leader & follower* dengan mengunakan tingkat prioritas penghindaran menghasilkan tabrakan lebih sedikit dibandingkan dengan simulasi tanpa menggunakan tingkat prioritas dengan rerata penurunan 55,53% untuk skenario pertama, dan 55,66% untuk skenario kedua.
2. Pengunaan tingkat prioritas penghindaran pada RVO berelasi *leader & follower* mengakibatkan waktu yang dibutuhkan agen *leader* untuk sampai pada tujuan kerumunan lebih cepat dibandingkan dengan simulasi tanpa menggunakan tingkat prioritas dengan rerata penurunan waktu sebesar 23,62% untuk skenario pertama dan 34,65% untuk skenario kedua.
3. Pengunaan tingkat prioritas penghindaran pada RVO berelasi *leader & follower* mampu mengatasi rintangan statis pada simulasi dan tidak mengakibatkan masalah *deadlock* pada simulasi.

Halaman ini sengaja dikosongkan

DAFTAR PUSTAKA

[1] C. W. Reynolds, "Steering behaviors for autonomous characters," Sony Computer Entertainment America, 1999. (Dikutip pada halaman xvii, 2, 12, 16, 17, 19, 22).

[2] S. Juniastuti, M. Fachri, S. M. S. Nugroho, and M. Hariadi, "Simulasi kerumunan menggunakan navigasi multiagen berbasis reciprocal velocity obstacles dengan relasi leader-follower," in International Symposium on Electronics and Smart Devices, 2016, Nov 2016. (Dikutip pada halaman xvii, 1, 13, 19, 20, 25, 28, 31).

[3] J. van den Berg, M. Lin, and D. Manocha, "Reciprocal velocity obstacles for real-time multi-agent navigation," University of North California, 2008. (Dikutip pada halaman 2, 5, 7, 8, 9, 10, 11, 20).

[4] P. Fiorini and Z. Shiller, "Motion planning in dynamic environments using velocity obstacles," International Journal of Robotics Research, vol. 17, no. 7, p. 760772, 1998. (Dikutip pada halaman 5).

[5] R. Legget, "Real-time crowd simulation: a review," R. Leggett, 2004. (Dikutip pada halaman 10).

[6] A. Kurniawati, S. M. Nugroho, and M. Hariadi, "Simulasi pergerakan pengunjung mall menggunakan potential field," Jurnal ilmiah Kursor, vol. 5, no. 3, January 2010. (Dikutip pada halaman 10).

[7] J. van den Berg, S. Pati, J. Sewall, D. Manocha, and M. Lin, "Interactive navigation of multiple agents in crowded environment," University of North California, 2008. No citations.

Halaman ini sengaja dikosongkan

LAMPIRAN

Tabel A1: Data hasil simulasi skenario satu kelompok kerumunan dengan tingkat prioritas penghindaran sama

No.	Jumlah Agen	Jumlah *collision*	Waktu simulasi	*Frame rate* rata-rata
1	25	0	103.167 detik	59.9222 fps
2	50	0	103.9846 detik	59.412 fps
3	75	8	104.0071 detik	58.6786 fps
4	100	583	122.8374 detik	57.7022 fps
5	125	2109	130.7406 detik	58.3139 fps
6	150	8875	131.7722 detik	57.9788 fps
7	175	49332	132.6729 detik	53.5904 fps
8	200	67067	135.7613 detik	51.1633 fps
9	225	96854	135.9926 detik	50.6424 fps
10	250	114730	136.5997 detik	49.9854 fps
11	275	161607	137.5621 detik	49.2068 fps
12	300	242454	139.891 detik	47.9659 fps
13	325	310654	142.4011 detik	43.2580 fps
14	350	384516	144.2285 detik	38.8966 fps
15	375	434378	146.0559 detik	34.6442 fps
16	400	470240	147.8833 detik	30.4970 fps
17	425	505102	149.7107 detik	26.451 fps
18	450	613824	151.5381 detik	22.5025 fps
19	475	771688	153.3655 detik	18.6482 fps
20	500	971688	155.1929 detik	10.7414 fps

Tabel A2: Data hasil simulasi skenario satu kelompok kerumunan dengan tingkat prioritas penghindaran berbeda

No.	Jumlah Agen	Jumlah *collision*	Waktu simulasi	*Frame rate* rata-rata
1	25	0	101.6197 detik	60.3328 fps
2	50	0	101.6263 detik	59.6204 fps
3	75	0	101.6329 detik	58.9081 fps
4	100	25	101.6395 detik	58.1959 fps
5	125	505	101.6529 detik	57.5291 fps
6	150	1369	101.6361 detik	51.3695 fps
7	175	2563	101.671 detik	48.6668 fps
8	200	5373	101.6658 detik	42.6102 fps
9	225	7043	101.6711 detik	40.8081 fps
10	250	10196	101.6748 detik	39.1346 fps
11	275	68466	101.7158 detik	37.4180 fps
12	300	116116	101.7281 detik	34.7102 fps
13	325	125544	101.734 detik	33.0175 fps
14	350	166983	101.7352 detik	30.1960 fps
15	375	200224	101.7813 detik	27.6868 fps
16	400	246314	101.8006 detik	26.6403 fps
17	425	285447	101.9159 detik	24.3927 fps
18	450	312705	102.0587 detik	20.7822 fps
19	475	506811	102.1938 detik	18.6508 fps

Dilanjutkan di halaman berikutnya

Tabel A2 – Lanjutan dari halaman sebelumnya

No.	Jumlah Agen	Jumlah *collision*	Waktu simulasi	*Frame rate* rata-rata
20	500	686326	102.4712 detik	17.4000 fps

Tabel A3: Data hasil simulasi skenario dua kelompok kerumunan dengan tingkat prioritas penghindaran sama

No.	Jumlah Agen	Jumlah *collision*	Waktu simulasi	*Frame rate* rata-rata
1	26	0	110,19	60,2686 fps
2	50	3	121,3315	60,0504 fps
3	76	24	124,8471	60,3298 fps
4	100	41	125,8374	60,2762 fps
5	126	2017	130,7406	60,2108 fps
6	150	2673	135,4402	59,4949 fps
7	176	3329	138,1398	59,0634 fps
8	200	3580	139,6307	56,9789 fps
9	226	14236	150,8955	57,8679 fps
10	250	57817	169,3352	57,4718 fps
11	276	70338	177,7749	50,1421 fps
12	300	92479	178,021	40,0458 fps
13	326	208207	187,7749	40,7829 fps
14	350	362670	191,9483	39,8753 fps
15	376	562670	196,1217	34,0503 fps
16	400	579903	199,7311	30,0204 fps
17	426	797487	205,8472	22,5701 fps
18	450	859332	209,5552	19,8086 fps
19	476	1076070	220,4587	14,8100 fps
20	500	1276070	228,0982	10,9558 fps

Tabel A4: Data hasil simulasi skenario dua kelompok kerumunan dengan tingkat prioritas penghindaran berbeda

No.	Jumlah Agen	Jumlah *collision*	Waktu simulasi	*Frame rate* rata-rata
1	26	0	103,3769	60,3230 fps
2	50	0	103,2688	60,3280 fps
3	76	18	103,1607	60,4978 fps
4	100	88	103,2933	60,3234 fps
5	126	208	103,1852	60,2024 fps
6	150	1087	103,2687	60,8897 fps
7	176	1447	103,5193	60,4235 fps
8	200	2642	103,9327	59,6829 fps
9	226	3955	104,0346	58,2018 fps
10	250	6617	104,1126	56,2564 fps
11	276	17632	104,2034	52,7718 fps
12	300	23261	104,5217	51,4152 fps
13	326	27875	104,6465	49,6242 fps
14	350	43288	104,6584	46,3126 fps
15	376	72797	104,7767	43,7597 fps
16	400	92100	104,8951	40,7550 fps

Dilanjutkan di halaman berikutnya

Tabel A4 – Lanjutan dari halaman sebelumnya

No.	**Jumlah Agen**	**Jumlah *collision***	**Waktu simulasi**	***Frame rate* rata-rata**
17	426	109605	104,9134	38,0695 fps
18	450	129892	104,9317	36,1187 fps
19	476	149898	105,0501	33,9647 fps
20	500	223947	105,2111	31,9263 fps

Halaman ini sengaja dikosongkan

BIOGRAFI PENULIS

Moch Fachri, lahir pada 18 Januari 1992 di Surabaya, Jawa Timur. Penulis menempuh pendidikan strata satu di Jurusan Teknik Elektro ITS Surabaya bidang studi Teknik Komputer dan Telematika sejak tahun 2010 hingga lulus pada tahun 2015. Saat di kuliah penulis aktif menjadi kepala departemen syiar kajian Islam Himatektro ITS 2012/2013. Penulis juga aktif menjadi Asisten laboratorium B201 (Telematika) hingga saat ini dan pernah menjabat sebagai koordinator *game development* Lab B201 dan asisten praktikum rangkaian digital periode 2013/2014. Selama masa kuliah penulis aktif dalam mengikuti ajang perlombaan seperti PKM (Program Kreatifitas Mahasiswa) serta kegiatan kerohanian kampus. Penulis menempuh pendidikan S2 di Jurusan Teknik Elektro ITS Surabaya bidang studi Jaringan Cerdas Multimedia bidang peminatan Teknologi Permainan sejak tahun 2016 dan diprediksikan lulus di tahun 2017

Halaman ini sengaja dikosongkan