

LAPORAN TUGAS BESAR 1

STRATEGI ALGORITMA

Tugas Besar 1 Strategi Algoritma



Kelompok 9 : TankubanPrau

Stefan Mattew Susanto 13523020

Brian Albar Hadian 13523048

Angelina Efrina Prahastaputri 13523060

PROGRAM STUDI TEKNIK INFORMATIKA
SEKOLAH TEKNIK ELEKTRO DAN INFORMATIKA
INSTITUT TEKNOLOGI BANDUNG
JL. GANESA 10, BANDUNG 40132

2025

DAFTAR ISI

DAFTAR ISI.....	2
BAB I.....	4
BAB II.....	6
2.1. Dasar Teori Algoritma <i>Greedy</i>	6
2.2. Cara Kerja Program Secara Umum.....	7
2.2.1. Cara Bot Melakukan Aksi.....	7
2.2.2. Implementasi Algoritma <i>Greedy</i> dalam Bot	8
2.2.3. Cara Menjalankan Bot	9
BAB III	10
3.1. <i>Mapping</i> Persoalan Robocode TankRoyale.....	10
3.2. Eksplorasi Alternatif Algoritma Greedy	10
1. Algoritma Greedy Bot “BotMeriang”	10
2. Algoritma Greedy Bot “Ormas”	11
3. Algoritma Greedy Bot “ChillPips”	13
4. Algoritma Greedy Bot “KPR”	14
3.3. Strategi Algoritma <i>Greedy</i> yang Dipilih.....	15
BAB IV	17
4.1. <i>Pseudocode</i> Program	17
4.1.1. <i>Pseudocode</i> Algoritma <i>Greedy</i> Bot “BotMeriang”	17
4.1.2. <i>Pseudocode</i> Algoritma <i>Greedy</i> Bot “Ormas”	18
4.1.3. <i>Pseudocode</i> Algoritma <i>Greedy</i> Bot “ChillPips”	20
4.1.4. <i>Pseudocode</i> Algoritma <i>Greedy</i> Bot “KPR”	21
4.2. Penjelasan Solusi <i>Greedy</i> yang Dipilih.....	22
4.2.1. Struktur Data pada Bot ChillPips.....	22
4.2.2. Fungsi dan Prosedur pada Bot ChillPips	22
4.3. Pengujian.....	23
4.4. Hasil Eksperimen	24
4.5. Analisis dan Pembahasan.....	27
BAB V	31
5.1. Kesimpulan	31
5.2. Saran	31
5.3. Komentar.....	31
5.4. Refleksi	31

LAMPIRAN.....	33
DAFTAR PUSTAKA	34

BAB I

DESKRIPSI TUGAS

Robocode adalah permainan pemrograman yang bertujuan untuk membuat kode bot dalam bentuk tank virtual untuk berkompetisi melawan bot lain di arena. Pertempuran Robocode berlangsung hingga bot-bot bertarung hanya tersisa satu seperti permainan Battle Royale, karena itulah permainan ini dinamakan Tank Royale. Nama Robocode adalah singkatan dari "Robot code," yang berasal dari versi asli/pertama permainan ini. Robocode Tank Royale adalah evolusi/versi berikutnya dari permainan ini, di mana bot dapat berpartisipasi melalui Internet/jaringan. Dalam permainan ini, pemain berperan sebagai programmer bot dan tidak memiliki kendali langsung atas permainan. Pemain hanya bertugas untuk membuat program yang menentukan logika atau "otak" bot. Program yang dibuat akan berisi instruksi tentang cara bot bergerak, mendeteksi bot lawan, menembakkan senjatanya, serta bagaimana bot bereaksi terhadap berbagai kejadian selama pertempuran.

Pada Tugas Besar pertama Strategi Algoritma ini, mahasiswa diminta untuk membuat sebuah bot yang nantinya akan dipertandingkan satu sama lain. Tentunya mahasiswa harus menggunakan **strategi greedy** dalam membuat bot ini.

Komponen-komponen dari permainan ini antara lain: *rounds* dan *turns*, batas waktu giliran, energi bot, peluru, panas meriam (*gun heat*), tabrakan, bagian tubuh/anatomii tank, pergerakan bot, berbelok, pemindaian dengan radar, dan skor. Untuk informasi lebih lengkap, silahkan buka dokumentasi Tank Royale pada link berikut. Untuk tugas besar ini, game engine yang akan digunakan sudah dimodifikasi oleh asisten. Source Code untuk game engine dan template bot telah disediakan pada tautan berikut. **Tubes1-if2211-starter-pack**

Spesifikasi Wajib

- Buatlah 4 bot (1 utama dan 3 alternatif) dalam bahasa **C# (.net)** yang mengimplementasikan **algoritma Greedy** pada *bot* permainan Robocode Tank Royale dengan tujuan memenangkan permainan.
- Tugas dikerjakan berkelompok dengan anggota **minimal 2 orang** dan **maksimal 3 orang**, boleh lintas kelas dan lintas kampus.

- Strategi *greedy* yang diimplementasikan setiap kelompok harus dikaitkan dengan fungsi objektif dari permainan ini, yaitu memperoleh skor setinggi mungkin pada akhir pertempuran. Hal ini dapat dilakukan dengan mengoptimalkan komponen skor yang telah dijelaskan diatas.
- Strategi *greedy* yang diimplementasikan **harus berbeda** untuk setiap bot yang diimplementasikan dan setiap strategi greedy harus menggunakan **heuristic** yang berbeda.
- **Bot yang dibuat TIDAK BOLEH sama dengan SAMPEL yang diberikan sebagai CONTOH. Baik dari starter pack maupun dari repository engine asli.**
- Buatlah strategi *greedy* terbaik, karena setiap **bot utama** dari masing-masing kelompok akan diadu dalam kompetisi Tubes 1.
- Strategi *greedy* yang kelompok anda buat harus **dijelaskan dan ditulis secara eksplisit** pada laporan, karena akan diperiksa saat demo apakah strategi yang dituliskan sesuai dengan yang diimplementasikan.
- Setiap kelompok dapat menggunakan kreativitas yang bermacam macam dalam menyusun strategi *greedy* untuk memenangkan permainan. Implementasi pemain **harus dapat dijalankan pada game engine** yang telah disebutkan diatas serta dapat dikompetisikan dengan bot dari kelompok lain.
- Program harus mengandung komentar yang jelas, dan untuk setiap strategi *greedy* yang disebutkan, harus dilengkapi dengan **kode sumber yang dibuat**. Artinya semua strategi harus diimplementasikan
- Mahasiswa disarankan membaca dokumentasi dari *game engine*. Perlu diperhatikan bahwa *game engine* yang digunakan untuk tubes ini **SUDAH DIMODIFIKASI**. Untuk Perubahan dapat dilihat pada bagian starter pack

BAB II

LANDASAN TEORI

2.1. Dasar Teori Algoritma *Greedy*

Algoritma *greedy* merupakan metode yang paling populer dan sederhana untuk memecahkan persoalan estimasi. Persoalan optimasi tersebut merupakan persoalan untuk mencari solusi paling optimal. Terdapat dua macam persoalan optimasi yakni *maximization* dan *minimization*.

Algoritma *greedy* memecahkan persoalan secara langkah per langkah (*step by step*) sedemikian sehingga pada setiap langkahnya, mengambil langkah terbaik yang dapat diperoleh pada saat itu (*locally optimal*) **tanpa** memperhatikan konsekuensi ke depan dan berharap bahwa dengan memilih optimal lokal tersebut pada setiap langkah, akan berakhir dengan solusi yang optimum global (*globally optimal*).

Elemen-elemen pada algoritma *greedy*:

1. Himpunan kandidat, berisi kandidat yang akan dipilih pada setiap langkah
2. Himpunan solusi, berisi kandidat yang sudah dipilih
3. Fungsi solusi, fungsi untuk menentukan apakah himpunan kandidat yang dipilih sudah memberikan solusi
4. Fungsi seleksi (*selection function*), fungsi untuk memilih kandidat berdasarkan strategi *greedy* tertentu (strategi *greedy* bersifat heuristik)
5. Fungsi kelayakan (*feasible function*), fungsi untuk memeriksa apakah kandidat yang dipilih dapat dimasukkan ke dalam himpunan solusi (layak atau tidak)
6. Fungsi obyektif, fungsi yang memaksimumkan atau meminimumkan

Dengan menggunakan elemen-elemen tersebut, dapat dikatakan bahwa algoritma *greedy* melibatkan pencarian sebuah himpunan bagian, S, dari himpunan kandidat, C; yang dalam hal ini, S harus memenuhi beberapa kriteria yang ditentukan, yaitu S menyatakan suatu solusi dan S dioptimisasi oleh fungsi obyektif.

Optimum global belum tentu merupakan solusi optimum atau solusi yang terbaik, tetapi bisa jadi merupakan solusi *sub-optimum* atau *pseudo-optimum*. Hal ini dapat terjadi karena algoritma *greedy* tidak beroperasi secara menyeluruh terhadap semua kemungkinan solusi yang ada selayaknya *exhaustive search*. Selain itu, terdapat fungsi seleksi yang berbeda-beda, sehingga harus dipilih fungsi yang tepat jika diinginkan solusi optimal dari algoritma ini. Jika solusi terbaik

mutlak tidak terlalu diperlukan, maka algoritma *greedy* dapat digunakan untuk menghasilkan solusi hampiran. Contoh persoalan yang dapat diselesaikan menggunakan algoritma *greedy* adalah *coin exchange problem*, *activity selection problem*, minimisasi waktu dalam sistem, *knapsack problem*, *job scheduling with deadlines*, *minimum spanning tree*, Huffman *coding-decoding*, dan *Egyptian fraction*.

2.2. Cara Kerja Program Secara Umum

2.2.1. Cara Bot Melakukan Aksi

Bot akan menjalankan fungsi berdasarkan event-driven system, yang berarti bot bereaksi terhadap berbagai peristiwa dalam permainan. Bot akan menggunakan fasilitas dan komponen yang terdapat di dalamnya. Komponen yang terdapat sebagai berikut:

- Body (Badan Bot) – Bagian utama bot yang menentukan posisi di arena dan dapat bergerak maju, mundur, serta berputar.
- Gun (Meriam) – Senjata bot yang bisa diarahkan secara independen dari badan bot untuk menembak musuh.
- Radar – Sensor yang berputar untuk mendekripsi posisi musuh dan memberikan data yang digunakan dalam strategi bot.
- Engine (Mesin) – Komponen yang mengatur kecepatan dan pergerakan bot dengan percepatan maksimum 1 unit per giliran dan perlambatan 2 unit per giliran.
- Bullet (Peluru) – Projektil yang ditembakkan dari meriam dengan kecepatan dan kekuatan yang bergantung pada firepower (0.1 hingga 3).
- Energy (Energi) – Sumber daya bot yang digunakan untuk bertahan dalam pertempuran; energi berkurang jika terkena serangan, menabrak tembok, atau menembak. Energi juga dapat bertambah ketika peluru berhasil menembak musuh ataupun lainnya.
- Turn Rate (Kecepatan Putar) – Mengontrol seberapa cepat bot bisa berbelok, dengan batasan yang bergantung pada kecepatannya
- Gun Heat (Panas Meriam) – Setelah menembak, meriam memiliki cooldown yang menentukan kapan bot bisa menembak lagi.
- Event Handling (Penanganan Peristiwa) – Bot bereaksi terhadap berbagai event seperti OnScannedBot (musuh terdeteksi), OnHitByBullet (ditembak), OnHitWall (menabrak

dinding), dan OnBotDeath (musuh mati) untuk mengambil keputusan dalam strategi pertempuran.

2.2.2. Implementasi Algoritma *Greedy* dalam Bot

Strategi greedy disematkan ke dalam game engine bisa dilakukan dengan cara membuat folder dengan struktur folder sebagai berikut:

```
→src
  →main-bot
    →
    →alternative-bots
      →
      →
      →
  →doc
  →README.md
```

Bot akan memiliki folder khusus yang merupakan isi dari bot tersebut. Pada tiap folder bot akan terdiri dari:

```
→bin
→obj
→bot.cs
→bot.cmd
→bot.csproj
→bot.json
→bot.sh
```

Algoritma greedy akan dimasukkan di dalam bot.cs pada fungsi run. Algoritma ini akan digunakan dan diimplementasikan di dalam fungsi bot itu sendiri. Fungsi yang dapat menerapkan algoritma greedy seperti:

- OnScannedBot()

Tindakan bot yang akan dilakukan saat bot musuh terbaca oleh radar.

- OnHitBot()

Tindakan bot yang akan dilakukan saat menabrak bot musuh.

- OnHitWall()

Tindakan bot saat bot menabrak tembok

- SmartFire()

Bot dapat memilih ukuran bullet yang digunakan berdasarkan jarak ataupun aspek lainnya.

2.2.3. Cara Menjalankan Bot

1. Buka file “ robocode-tankroyale-gui-0.30.0.jar ” atau jalankan dengan terminal “java -jar robocode-tankroyale-gui-0.30.0.jar”
2. Tambahkan folder boot root directories pada menu config di Robocode Tank Royale.
3. Klik start battle pada menu battle.
4. Pilih bot yang ingin digunakan. Lalu boot bot yang akan digunakan .
5. Add bot yang akan dijalankan pada permainan.
6. Jalankan permainan dengan start battle.

BAB III

APLIKASI STRATEGI GREEDY

3.1. *Mapping* Persoalan Robocode TankRoyale

Persoalan Robocode TankRoyale pada Tugas Besar 1 ini kami uraikan menjadi elemen-elemen algoritma *greedy* seperti berikut:

1. Himpunan kandidat
(pergerakan bot, berbelok, pemindaian/memutar radar, menembak dengan *firepower* tertentu)
2. Himpunan solusi
Urutan perintah yang valid yang diterima oleh bot untuk setiap *turn* dan *round* untuk memenangkan permainan
3. Fungsi solusi
Memastikan bot bertahan hidup pada *turn* tersebut
4. Fungsi seleksi (*selection function*)
Memilih perintah yang memberikan damage sebesar mungkin kepada bot lawan sekaligus menerima damage sesedikit mungkin dari bot lawan
5. Fungsi kelayakan (*feasible function*)
Perintah valid dengan mempertimbangkan apakah *gunheat* sedang maksimum, apakah energi masih ada untuk menembak peluru, dan apakah kecepatan gerak valid
6. Fungsi obyektif
Memaksimalkan skor yang didapat untuk setiap *round*

3.2. Eksplorasi Alternatif Algoritma Greedy

1. Algoritma Greedy Bot “BotMeriang”

Dalam permainan, bot memerlukan strategi yang terbaik untuk dapat memperoleh skor terbaik. Untuk memperoleh skor, bot dapat menyerang musuh, menghindari peluru, ataupun menabrak musuh. Strategi yang digunakan pada bot BotMeriang adalah strategi *greedy by survival*, bot akan fokus untuk bertahan hidup dengan cara menggunakan gerakan tertentu. Pada awal pertandingan, bot akan bergerak menuju ke titik pusat permainan. Saat bot sudah mencapai titik pusat, bot akan mengelilingi pusat dengan gerakan melingkar. Pergerakan ini

bertujuan agar bot tidak menjadi target statis yang mudah ditembak, sekaligus mempertahankan posisi strategis di tengah arena permainan. Bot akan melaju dengan kecepatan maksimal supaya terhindar dari peluru yang relatif ringan dan cepat. Radius gerakan bot akan cukup besar supaya bot dapat menabrakkan diri ke musuh yang menempel di walls sehingga memungkinkan algoritma musuh dapat berubah.

Tidak hanya menghindari peluru, bot akan tetap menembak bot yang terbaca. Bot akan menembak musuh yang terbaca dengan *firepower* yang besar jika posisi musuh dekat. Untuk menghemat energi yang terpakai maka musuh dengan posisi yang jauh akan ditembak dengan *firepower* kecil. Namun, jika bot memiliki energi kurang dari 20, bot akan fokus untuk survival dan menghindari semua peluru yang ditargetkan.

Saat bot menabrak musuh ataupun tembok maka bot akan mundur dan berbelok. Lalu bot akan melanjutkan strategi keliling lingkaran kembali. Dengan tetap berada di pusat arena, bergerak dalam pola melingkar, serta menghindari tabrakan, bot ini mampu bertahan lebih lama dibandingkan bot lain dan mengumpulkan survival score sebanyak mungkin. Meskipun tidak selalu menjadi bot dengan jumlah kill terbanyak, strategi ini efektif untuk mendapatkan poin tinggi dalam pertandingan.

Berdasarkan eksperimen, bot BotMeriang memiliki kekurangan seperti arah bullet pada target yang bergerak, lokasi bot yang tidak strategis, dan juga jika ada musuh yang tidak bergerak berada di pusat arena maka program bisa tidak dijalankan dengan baik seperti target awal, karena bot tidak bisa mencapai titik pusat dan tidak bisa menjalankan strategi melingkar. Namun bot BotMeriang memiliki beberapa kelebihan, seperti dapat menghindari bullet secara efektif, menembak bot musuh yang cukup akurat pada kasus pergerakan musuh kurang lincah, dan dapat mengatur kekuatan tembak berdasarkan jarak.

2. Algoritma *Greedy* Bot “Ormas”

Meninjau bahwa tujuan utama dari permainan Robocode adalah mendapatkan jumlah skor tertinggi dalam setiap ronde, kami mengambil pendekatan untuk meningkatkan skor dengan pendekatan agresif, yakni mendekati dan menyerang seluruh bot yang ada menggunakan radar. Hal ini kami manfaatkan dengan menerapkan strategi *tailing* atau mengekor suatu target dan menyerang bot lain dengan harapan dapat memberikan *bullet*

damage sebanyak mungkin sekaligus mengurangi kemungkinan terjadi penyerangan karena selalu bergerak mengikuti target yang sudah ditentukan sebelumnya.

Strategi yang digunakan oleh bot Ormas berpusat pada penerapan strategi *greedy by target*. Hal ini diimplementasikan dengan bot Ormas akan mencari musuh terdekat (dalam rentang jarak > 100 pixel) dan mengikuti bot tersebut sembari menembakkan peluru ke sembarang tempat. Penembakkan peluru tersebut dilakukan secara acak dengan harapan pada awal permainan, bot Ormas dapat mengenai sebanyak mungkin bot lawan dan mendapatkan poin awal terbanyak. Selanjutnya, bot Ormas tersebut akan melakukan scan kembali dan melakukan proses tersebut hingga selesai. Jika bot yang menjadi target sebelumnya telah mati atau bot Ormas sempat menabrak tembok, bot akan melakukan scan kembali untuk mendapatkan target terbaru. Jika bot tersebut sempat mengalami *ramming*, maka bot tersebut akan mengarahkan meriam ke arah tabrakan dan menembak kembali bot tersebut. Jika bot Ormas mengalami tembakan, maka bot akan berbelok ke kiri dan ke kanan yang dilakukan secara bergantian. Jika bot Ormas memiliki energi di bawah 20, maka bot akan menempel pada dinding dan berjalan di sekitar dinding dengan menembak asal selama energi masih mencukupi. Hal ini dilakukan untuk mendapatkan skor sebanyak mungkin dan menghindari *ramming* atau konfrontasi secara langsung dengan tank musuh lainnya.

Berdasarkan eksperimen yang dilakukan, Bot Ormas memiliki kekurangan diantaranya

1. Bot Ormas membutuhkan banyak waktu untuk merespon sehingga meningkatkan kemungkinan bot untuk mengalami *ramming* atau terkena tembakan
2. Bot Ormas memiliki kelemahan terhadap bot musuh yang memiliki algoritma menyasar satu bot dalam satu waktu.
3. Bot Ormas memiliki kelemahan terhadap bot yang memiliki pola gerak yang tidak dapat diprediksi atau memiliki pola yang sulit untuk direplika karena serangan bot Ormas berfokus pada banyaknya tembakan yang mengenai bot musuh tanpa mempedulikan apakah bot tersebut adalah bot yang sama seperti sebelumnya atau bukan.

Sementara itu, Bot Ormas juga memiliki kelebihan, diantaranya,

1. Bot Ormas selalu bergerak sehingga menurunkan kemungkinan untuk terkena peluru dari bot lain.

2. Bot Ormas memiliki waktu pengisian ulang yang kecil sehingga meningkatkan banyaknya peluru yang dapat mengenai bot lain.

3. Algoritma *Greedy* Bot “ChillPips”

Dalam teknis permainan, setiap ada bot lainnya yang mati, maka bot yang masih bertahan akan mendapatkan *survival score* sebesar 50 poin. Karena poin terbilang cukup besar, maka kami merancang algoritma *greedy by survival* atau bertahan hidup selama mungkin untuk mempertahankan energi sebanyak mungkin untuk mendapatkan poin sebanyak-banyaknya. Dalam permainan, awalnya bot akan bergerak menuju tembok terdekat lalu bergerak sepanjang tembok (tidak menabrak tembok) dengan kecepatan random (antara 5 sampai 8). Aksi ini mengurangi kemungkinan terkena peluru dari musuh dengan memposisikan jarak sejauh mungkin dengan posisi yang lebih sulit diduga. Mekanisme *defense* dari bot ini adalah jika bot mendeteksi musuh, bot akan menembak peluru ke arah musuh dengan *firepower* 1. Jika jarak ke musuh lebih dekat daripada suatu jarak “aman” yang diatur, maka bot akan menembakkan peluru dengan *firepower* tertinggi yakni sebesar 3 ke arah musuh sambil bergerak dengan kecepatan maksimum. Dengan kecepatan maksimum, bot dapat menghindar dari musuh sehingga meminimalkan *damage* dari musuh, sedangkan dengan menembak peluru, harapannya musuh lebih cepat mati daripada bot sehingga bot lebih lama bertahan hidup. Jika bot mengalami *ramming* (baik menabrak maupun ditabrak), maka akan langsung mengganti arah gerak berlawanan dengan tetap bergerak sepanjang tembok untuk menghindari musuh. Apabila bot bertahan lebih lama daripada bot lain (memanfaatkan kondisi bot lain mati terlebih dahulu), maka akan mendapatkan *survival score* dan *survival score* terbanyak.

Bot ini kurang efektif jika terdapat banyak musuh yang mengepung sekaligus di sekitarnya yang mengakibatkan ruang gerak bot sangat kecil (terbatas tembok juga). Ketika terjadi *ramming*, bot tidak memiliki alternatif gerakan *escaping* dan akan menerima *damage* yang besar akibat ruang gerak yang terbatas. Karena musuh dalam jarak dekat, maka juga bot akan menghabiskan *energy* untuk menembak dengan *firepower* terbesar sehingga lebih cepat mati. Namun, bot ini memiliki kelebihan tersendiri dalam menghindari serangan musuh, baik yang melakukan konfrontasi secara langsung maupun dari jarak jauh, dengan pergerakan yang bervariatif sambil mempertahankan jarak aman dari musuh. Untuk musuh

yang melakukan konfrontasi secara langsung dan melewati jarak aman, bot dapat mengatur kekuatan tembakan yang harapannya jika mengenai musuh akan berpengaruh pada langkah musuh berikutnya atau bahkan menyebabkan matinya musuh. *Energy* yang didapat jika tembakan mengenai musuh juga menjadikan bot lebih lama bertahan hidup.

4. Algoritma *Greedy* Bot “KPR”

Melihat bahwa tujuan utama dalam permainan Robocode adalah untuk mendapatkan skor sebanyak mungkin untuk setiap ronde, Kami mengajukan strategi algoritma untuk menyerang bot dengan jumlah energi terkecil terlebih dahulu. Hal ini dilakukan untuk mengurangi jumlah pemain terlebih dahulu sehingga mengurangi kemungkinan untuk tertembak. Hal ini diimplementasikan dengan pendekatan *clustering*, yakni mengelompokkan bot dengan energi terendah terlebih dahulu, lalu memutuskan untuk mendekati *cluster* tersebut dan menyerang pada daerah tersebut dengan harapan dapat membunuh bot musuh secepat mungkin dan mendapatkan skor tambahan dari *survival bonus*.

Strategi yang digunakan oleh Bot KPR adalah *Greedy by Energy* dengan menyerang terlebih dahulu bot dengan energi yang lebih sedikit terlebih dahulu. Bot KPR akan melakukan scan untuk menentukan urutan Bot KPR yang memiliki energi terkecil hingga terbesar. Selanjutnya, Bot KPR kemudian akan melakukan clustering terhadap maksimum tiga dari urutan bot tersebut dari urutan bot yang telah diurutkan dari terkecil hingga terbesar berdasarkan energi. Kemudian, dilakukan perhitungan terhadap rata-rata koordinat dari cluster tersebut dan Bot KPR akan menuju titik tersebut dan melakukan penembakan asal secara memutar. Hal ini dilakukan agar Bot KPR dapat memiliki skor tertinggi dan dapat membunuh tank musuh terlebih dahulu dan mendapatkan bonus untuk membunuh tank tersebut. Apabila terdapat tank mati, baik yang menjadi anggota dari cluster target ataupun bukan, bot KPR akan melakukan scan untuk mendapatkan cluster terbaru dan dilakukan penargetan kembali terhadap cluster tersebut. Jika bot KPR mengalami *ramming* atau tertembak peluru, maka bot KPR akan melakukan putar kiri sebesar 90 derajat dan berjalan untuk menghindari bot musuh tersebut sehingga dapat menghindari pengurangan poin lebih banyak. Apabila bot KPR menabrak tembok atau bot lain, maka bot KPR akan melakukan scan ulang untuk mendapatkan data cluster terbaru.

Berdasarkan eksperimen, bot KPR memiliki beberapa kelebihan diantaranya,

1. Bot KPR memiliki kemungkinan tinggi untuk mendapatkan bonus tambahan dari membunuh bot dengan energi yang lebih rendah karena survival bonus pada awal game
2. Bot KPR memiliki kemungkinan tinggi untuk bertahan hidup karena hanya membutuhkan waktu yang sedikit untuk mengarah ke arah baru dan bergerak dengan jarak yang tidak jauh (< 100 pixel) sehingga mengurangi kesempatan untuk ditembak kembali oleh peluru yang sama dan meningkatkan waktu untuk menarget kembali cluster.

Sementara itu, bot KPR juga memiliki beberapa kekurangan diantaranya,

1. Bot KPR memiliki kemungkinan tinggi untuk bergerak ke arah musuh yang banyak karena sistem cluster memungkinkan bot KPR untuk mendapatkan cluster yang mengarah ke koordinat dengan banyak bot lainnya sehingga meningkatkan kemungkinan tertembak peluru.
2. Bot KPR kurang bisa beradaptasi dalam periode yang masih
3. memiliki banyak bot karena bot KPR tidak dapat mengganti target dengan mudah sehingga tidak dapat melakukan disrupt atau balasan terhadap bot yang menyerang, tetapi tidak termasuk dalam cluster.

3.3. Strategi Algoritma *Greedy* yang Dipilih

Strategi algoritma greedy yang akhirnya kami pilih adalah strategi algoritma greedy pada bot ChillPips. Menurut kami, strategi greedy by survival yang diterapkan pada bot ChillPips dapat memenangkan permainan dengan cukup konsisten karena algoritma ini berfokus pada bagaimana bot dapat bertahan hidup selama mungkin. Hal ini dapat dicapai dengan meminimalisasikan damage yang diterima dengan cara melakukan pergerakan yang variatif dan pada jarak sejauh mungkin dari musuh (menyusuri tembok sehingga tidak menerima wall damage). Selain itu, musuh yang dekat pada bot dapat dihindari serangannya dengan pergerakan yang cepat (max speed), dan mekanisme defense dengan memberikan tembakan pada musuh berdasarkan jarak aman bot dengan musuh. Selain meminimalisasikan damage yang diterima, bot perlu energy untuk terus bertahan hidup. Maka dari itu, bot ChillPips akan terus menembak musuh yang terdeteksi meskipun tidak melewati jarak aman dengan harapan peluru yang terkena musuh dapat memberikan energy tambahan untuk bertahan hidup. Kelemahan dari strategi ini adalah ketika bot

dikepung oleh banyak musuh dari berbagai arah dan juga terhalang oleh tembok, yang menyebabkan bot mengalami ramming dengan musuh dan menghabiskan energy pada mekanisme defense sehingga memungkinkan bot sangat cepat untuk mati. Namun, hal ini kami anggap sebagai kondisi yang sangat mungkin untuk terjadi terutama jika dalam permainan bot yang ditandingkan berjumlah cukup banyak. Selain itu, karena bot tidak melakukan serangan secara konfrontasi langsung kepada musuh, maka bot bisa saja exhausted karena serangan jarak jauh bot kurang akurat dan menyebabkan bot bisa saja mati kehabisan energy. Hal ini juga kami anggap sebagai kondisi yang sangat mungkin terjadi terutama saat *late-game*.

Kami tidak memilih strategi algoritma greedy pada bot BotMeriang karena BotMeriang bergerak kurang fleksibel saat menabrak tembok atau pun saat menabrak musuh, BotMeriang akan mundur dan berbelok sedikit yang bisa lebih mudah untuk terkena peluru karena masih berada dalam jangkauan musuh. BotMeriang harus mencapai titik pusat arena untuk menjalankan algoritma melingkar, jika ada satu musuh diam berada di tengah pusat, maka BotMeriang harus menunggu musuh itu mati baru bisa jalan melingkar. Sedangkan algoritma ChillPips jika ada musuh yang menghalangi maka bot tetap jalan dan akan berbalik arah untuk mencapai dinding lainnya.

Kami tidak memilih bot Ormas karena penerapan strategi algoritma greedy kurang efektif dalam menyerang. Skema algoritma bot Ormas berfokus menembakkan ke berbagai arah sehingga mengurangi kemungkinan peluru mengenai lawan. Selain itu, bot Ormas juga hanya berfokus pada pemilihan target secara acak sehingga mengurangi kemungkinan untuk menyerang bot lawan secara terfokus hingga bot musuh tersebut mati. Bot Ormas juga tidak memiliki cara untuk kabur pada saat terjadi tabrakan antara bot ataupun tembok sehingga mengurangi kemungkinan bertahan hidup apabila kondisi masih bersifat dinamis.

Terakhir, Kami tidak memilih bot KPR karena strategi algoritma *greedy* pada bot tersebut mengakibatkan bot untuk bergerak menuju *cluster* dengan jumlah bot dengan energi yang sedikit. Hal ini meningkatkan kemungkinan bot KPR untuk terkena tembakan lebih sering karena memasuki area terbuka terbatas dengan *cluster*. Selanjutnya, KPR juga tidak memiliki mekanisme untuk bergerak apabila sudah mencapai titik yang ditentukan berdasarkan *cluster* sehingga meningkatkan kemungkinan untuk dikepung oleh banyak bot musuh lainnya.

BAB IV

IMPLEMENTASI DAN PENGUJIAN

4.1. *Pseudocode* Program

4.1.1. *Pseudocode* Algoritma Greedy Bot “BotMeriang”

```

procedure Main()
    new BotMeriang().Start()
procedure BotMeriang()
    base <- BotInfo.FromFile("BotMeriang.json")
procedure Run()
    GunTurnRate <- 15
    while (isRunning) do
        if (not sudahDiPusat) then
            MovePoint()
        else
            OrbitCenter()
            Go()
procedure OnScannedBot(input e)
    if (Energy > 20) then
        jarak <- DistanceTo(e.X, e.Y)
        SmartFire(jarak)
procedure SmartFire(input distance)
    firePower <- 1
    if (distance < 100) then
        firePower <- 3
    else if (distance < 300) then
        firePower <- 2 Fire(firePower)
procedure OnHitBot(input e)
    Back(50)
    TurnRight(15)
    TargetSpeed <- 8
    Go()
procedure OnHitWall(input e)
    Back(30)
    TurnRight(90)
    TargetSpeed <- 8
    Go()
procedure MovePoint()
    centerX <- ArenaWidth / 2
    centerY <- ArenaHeight / 2
    jarak <- DistanceTo(centerX, centerY)
    if (jarak > 10) then
        bearingToCenter <- CalcBearing(centerX, centerY)
        TurnRate <- NormalizeBearing(bearingToCenter)
        TargetSpeed <- 8
        Forward(min(jarak, 50))
    else
        sudahDiPusat <- true
procedure OrbitCenter()

```

```

centerX <- ArenaWidth / 2
centerY <- ArenaHeight / 2
bearingToOrbit <- CalcBearing(centerX, centerY) + 90
TurnRate <- NormalizeBearing(bearingToOrbit)
TargetSpeed <- 8
Go()
function CalcBearing(input targetX, targetY) -> double
    angle <- atan2(targetY - Y, targetX - X) * (180 / PI)
    -> NormalizeBearing(angle - Direction)
function NormalizeBearing(input angle) -> double
    while (angle > 180) do
        angle <- angle - 360
    while (angle < -180) do
        angle <- angle + 360
    -> angle

```

4.1.2. Pseudocode Algoritma Greedy Bot “Ormas”

```

procedure Main()
    new Ormas().Start()
procedure Ormas()
    base <- BotInfo.FromFile("Ormas.json")
procedure Run()
    while (isRunning) do
        if (Energy >= 20) then
            TurnRadarLeft(40)
        else
            EvadeMode()
        targetID <- (-1)
procedure OnScannedBot(input e)
    if (targetID = -1) then
        if (DistanceTo(e.X, e.Y) > 100) then
            targetID <- e.ScannedBotId
        ChaseTarget(e)
procedure OnHitByBullet(input e)
    if (isHit) then
        TurnLeft(BearingTo(e.Bullet.X * 2, e.Bullet.Y))
        isHit <- true
    else
        TurnRight(BearingTo(e.Bullet.X * 3, e.Bullet.Y))
        isHit <- false
    Forward(50)
    targetID <- (-1)
procedure OnHitBot(input e)
    if (Energy >= 20) then
        TurnToFaceTarget(e.X, e.Y)
        Fire(2)
    else
        TurnLeft(180)
procedure OnBotDeath(input e)
    if (e.VictimId = targetID) then
        targetID <- (-1)

```

```

    Rescan()
procedure EvadeMode()
    TurnLeft(BearingTo(0, Y))
procedure ChaseTarget(input target)
    TurnToFaceTarget(target.X, target.Y)
    distance <- DistanceTo(target.X, target.Y)
    iter <- 12
    step <- distance / iter
    while (iter > 0) do
        TurnGunLeft(30)
        Fire(0.1)
        if (iter = 1) then
            Forward(step + 5)
        else
            Forward(step)
        iter <- iter - 1
procedure TurnToFaceTarget(input x, y)
    bearing <- BearingTo(x, y)
    bearingGun <- GunBearingTo(x, y)
    bearingRadar <- RadarBearingTo(x, y)
    TurnLeft(bearing)
    TurnGunLeft(bearingGun)
    TurnRadarLeft(bearingRadar)
procedure OnHitWall(input e)
    if (Energy >= 20) then
        targetID <- (-1)
        Rescan()
    else
        distance <- DistanceTo(0, 0)
        iter <- 12
        step <- distance / iter
        degStep <- 180 / iter
        TurnLeft(90)
        if (X = 0 and Y = ArenaHeight) then
            if (GunDirection /= 270) then
                TurnGunLeft(BearingTo(X, 0))
            while (iter > 0) do
                TurnGunLeft(degStep)
                Fire(0.1)
                if (iter = 1) then
                    Forward(step + 5)
                else
                    Forward(step)
                iter <- iter - 1
            Forward(ArenaHeight)
        else if (X = ArenaWidth and Y = 0) then
            if (GunDirection /= 90) then
                TurnGunLeft(BearingTo(X, ArenaHeight))
            while (iter > 0) do
                TurnGunLeft(degStep)
                Fire(0.1)
                if (iter = 1) then
                    Forward(step + 5)
                else

```

```

        Forward(step)
        iter <- iter - 1
        Forward(ArenaHeight)
    else if (X = ArenaWidth and Y = ArenaHeight) then
        if (GunDirection != 180) then
            TurnGunLeft(BearingTo(X, 0))
        while (iter > 0) do
            TurnGunLeft(degStep)
            Fire(0.1)
            if (iter = 1) then
                Forward(step + 5)
            else
                Forward(step)
            iter <- iter - 1
            Forward(ArenaWidth)
    else
        if (GunDirection != 0) then
            TurnGunLeft(BearingTo(ArenaWidth, 0))
        while (iter > 0) do
            TurnGunLeft(degStep)
            Fire(0.1)
            if (iter = 1) then
                Forward(step + 5)
            else
                Forward(step)
            iter <- iter - 1
        Forward(ArenaWidth)

```

4.1.3. Pseudocode Algoritma Greedy Bot “ChillPips”

```

procedure Main()
    new ChillPips().Start()
procedure Ormas()
    base <- BotInfo.FromFile("ChillPips.json")
procedure Run()
    MaxSpeed <- 8
    TurnLeft(DirectionToWall(X, Y))
    Forward(DistanceToWall(X, Y))
    TurnRight(90)
    while (IsRunning) do
        MaxSpeed <- random(5, 9)
        SetTurnGunRight(double PositiveInfinity)
        if (movingForward = true) then
            Forward(SafeMoveAmountForward(Direction))
            TurnRight(90)
        else {movingForward = false}
            Back(SafeMoveAmountBackward(Direction))
            TurnLeft(90)
function SafeMoveAmountForward(input direction) -> double
    if (direction = 0) then -> abs(ArenaWidth - X - jarakAmanWall)
    else if (direction = 90) then -> abs(ArenaHeight - Y - jarakAmanWall)
    else if (direction = 180) then -> (X - jarakAmanWall)

```

```

    else -> (Y - jarakAmanWall) {direction = 270}
function SafeMoveAmountBackward(input direction) -> double
    if (direction = 0) then -> (X - jarakAmanWall)
    else if (direction = 90) then -> (Y - jarakAmanWall)
    else if (direction = 180) then -> abs(ArenaWidth - X - jarakAmanWall)
    else -> abs(ArenaHeight - Y - jarakAmanWall) {direction = 270}
function DistanceToWall(input x, y) -> double
    distanceToLeftWall <- DistanceTo(0, y) - jarakAmanWall
    distanceToRightWall <- DistanceTo(ArenaWidth, y) - jarakAmanWall
    distanceToUpperWall <- DistanceTo(x, ArenaHeight) - jarakAmanWall
    distanceToBottomWall <- DistanceTo(x, 0) - jarakAmanWall
    nearestDistance <- min(min(distanceToLeftWall, distanceToRightWall),
    min(distanceToUpperWall, distanceToBottomWall))
    -> nearestDistance
function DirectionToWall(input x, y) -> double
    nearestDistance <- DistanceToWall(x, y)
    if (DistanceTo(0, y) - jarakAmanWall = nearestDistance) then
        -> CalcBearing(180)
    if (DistanceTo(ArenaWidth, y) - jarakAmanWall = nearestDistance) then
        -> CalcBearing(0)
    if (DistanceTo(x, ArenaHeight) - jarakAmanWall = nearestDistance) then
        -> CalcBearing(90)
    if (DistanceTo(x, 0) - jarakAmanWall = nearestDistance) then
        -> CalcBearing(270)
    -> 0
procedure OnScannedBot(input e)
    if (DistanceTo(e.X, e.Y) <= jarakAmanMusuh) then
        Fire(3)
    else
        Fire(1)
procedure OnHitBot(input e)
    MaxSpeed <- 8
    movingForward <- not movingForward

```

4.1.4. Pseudocode Algoritma Greedy Bot “KPR”

```

MAX_TANK_COUNT <- 10
isHit <- false
clusterLength <- 0
enemyCount <- 0
cluster <- null
enemyArray <- array of MAX_TANK_COUNT initialized with null
procedure Main()
    new KPR().Start()
procedure Ormas()
    base <- BotInfo.FromFile("KPR.json")
procedure Run()
    if not IsEmptyEnemyArray(enemyArray) then
        i iterate [0..(MAX_TANK_COUNT - 1)]
        enemyArray[i] <- null
    while (IsRunning) do
        if IsEmptyEnemyArray(enemyArray) then

```

```

        TurnRadarLeft(360)
else
    TurnGunLeft(45)
    Fire(1)
    enemyArray <- null
    cluster <- null
procedure OnScannedBot(input e)
    if IsEmptyEnemyArray(enemyArray) then
        if enemyCount < 10 then
            enemyArray[enemyCount] <- e
            enemyCount <- enemyCount + 1
        sortedEnemyArray <- EnemyEventsOfSmallestDistance(enemyArray)
        clusterLength <- min(3, enemyCount)
        cluster <- array of clusterLength
        k iterate [0.. (clusterLength - 1)]
            cluster[k] <- sortedEnemyArray[k]
    ChaseCluster(cluster)

```

4.2. Penjelasan Solusi Greedy yang Dipilih

4.2.1. Struktur Data pada Bot ChillPips

Program bot ChillPips menggunakan paradigma berorientasi objek dalam bahasa C# dan dengan struktur data terdiri atas variabel primitif dan konstanta, konstruktor, metode-metode pembantu navigasi bot, dan metode bawaan kelas Bot untuk menyerang atau *event handling*.

4.2.2. Fungsi dan Prosedur pada Bot ChillPips

Fungsi / Prosedur	Tujuan
<code>public override void Run()</code>	Untuk menjalankan program berisi perintah aksi untuk bot
<code>private double SafeMoveAmountForward(double direction)</code>	Mengembalikan jarak aman terjauh yang dapat ditempuh bot untuk maju berdasarkan arah gerak tanpa menyentuh tembok
<code>private double SafeMoveAmountBackward(double direction)</code>	Mengembalikan jarak aman terjauh yang dapat ditempuh bot untuk mundur berdasarkan arah gerak tanpa menyentuh tembok
<code>private double DistanceToWall(double x, double y)</code>	Menentukan tembok terdekat dengan bot dan mengembalikan jaraknya

<code>private double DirectionToWall(double x, double y)</code>	Menentukan tembok terdekat dengan bot dan mengembalikan arah (<i>degree</i>) yang harus bot tuju untuk mencapai tembok
<code>public override void OnScannedBot()</code>	Untuk <i>event handling</i> yakni ketika bot mendeteksi musuh pada radar
<code>public override void OnHitBot()</code>	Untuk <i>event handling</i> yakni ketika bot menabrak atau ditabrak bot lainnya

4.3. Pengujian

Results for 10 rounds											
Rank	Name	Total Score	Survival	Surv. Bonus	Bullet D...	Bullet Bo...	Ram Dmg.	Ram Bonus	1sts	2nds	3rds
1	ChillPips 1.0	981	350	30	528	68	5	0	2	2	0
2	BotMeriang 1.0	907	400	60	422	6	19	0	2	2	0
3	Ormas 1.0	147	100	0	13	0	34	0	0	0	3
4	KPR 1.0	54	50	0	0	0	4	0	0	0	2

Results for 10 rounds											
Rank	Name	Total Score	Survival	Surv. Bonus	Bullet D...	Bullet Bo...	Ram Dmg.	Ram Bonus	1sts	2nds	3rds
1	BotMeriang 1.0	882	350	30	434	44	24	0	1	2	0
2	ChillPips 1.0	781	350	30	352	44	4	0	2	1	0
3	Ormas 1.0	120	100	0	14	0	6	0	0	0	2
4	KPR 1.0	72	50	0	14	0	8	0	0	0	1

Results for 10 rounds											
Rank	Name	Total Score	Survival	Surv. Bonus	Bullet D...	Bullet Bo...	Ram Dmg.	Ram Bonus	1sts	2nds	3rds
1	ChillPips 1.0	1040	400	60	504	60	16	0	3	0	0
2	BotMeriang 1.0	716	300	0	394	5	17	0	0	3	0
3	Ormas 1.0	136	100	0	23	0	13	0	0	0	2
4	KPR 1.0	66	50	0	12	0	4	0	0	0	1

Results for 10 rounds											
Rank	Name	Total Score	Survival	Surv. Bonus	Bullet D...	Bullet Bo...	Ram Dmg.	Ram Bonus	1sts	2nds	3rds
1	ChillPips 1.0	1284	500	90	604	83	7	0	4	0	0
2	BotMeriang 1.0	818	350	0	398	45	24	0	0	4	0
3	Ormas 1.0	165	150	0	2	0	13	0	0	0	3
4	KPR 1.0	62	50	0	8	0	4	0	0	0	1

Results for 10 rounds											
Rank	Name	Total Score	Survival	Surv. Bonus	Bullet D...	Bullet Bo...	Ram Dmg.	Ram Bonus	1sts	2nds	3rds
1	ChillPips 1.0	1284	500	60	644	64	16	0	2	2	0
2	BotMeriang 1.0	1085	450	30	522	58	25	0	2	2	0
3	KPR 1.0	164	150	0	9	0	5	0	0	0	3
4	Ormas 1.0	107	50	0	15	0	42	0	0	0	1

Results for 10 rounds											-	□	×	
Rank	Name	Total Score	Survival	Surv. Bonus	Bullet D...	Bullet Bo...	Ram Dmg.	Ram Bonus	1sts	2nds	3rds			
1	ChillPips 1.0	1136	400	60	608	65	2	0	3	1	0			
2	BotMeriang 1.0	911	350	30	442	45	43	0	1	3	0			
3	KPR 1.0	151	150	0	0	0	1	0	0	0	3			
4	Ormas 1.0	41	0	0	13	0	28	0	0	0	1			

Results for 10 rounds											-	□	×	
Rank	Name	Total Score	Survival	Surv. Bonus	Bullet D...	Bullet Bo...	Ram Dmg.	Ram Bonus	1sts	2nds	3rds			
1	ChillPips 1.0	892	350	60	444	32	6	0	2	1	0			
2	BotMeriang 1.0	809	300	30	432	15	31	0	1	2	0			
3	KPR 1.0	67	50	0	12	0	5	0	0	0	1			
4	Ormas 1.0	56	50	0	2	0	4	1	0	0	2			

Results for 10 rounds											-	□	×	
Rank	Name	Total Score	Survival	Surv. Bonus	Bullet D...	Bullet Bo...	Ram Dmg.	Ram Bonus	1sts	2nds	3rds			
1	BotMeriang 1.0	1289	550	90	542	50	35	21	4	0	0			
2	ChillPips 1.0	827	400	0	404	21	1	0	0	3	1			
3	Ormas 1.0	442	200	0	14	0	205	22	0	1	3			
4	KPR 1.0	12	0	0	5	0	7	0	0	0	0			

Results for 10 rounds											-	□	×	
Rank	Name	Total Score	Survival	Surv. Bonus	Bullet D...	Bullet Bo...	Ram Dmg.	Ram Bonus	1sts	2nds	3rds			
1	ChillPips 1.0	1263	500	60	616	81	6	0	2	2	0			
2	BotMeriang 1.0	1132	450	30	562	49	40	0	2	2	0			
3	Ormas 1.0	122	100	0	3	0	19	0	0	0	2			
4	KPR 1.0	100	100	0	0	0	0	0	0	0	2			

Results for 10 rounds											-	□	×	
Rank	Name	Total Score	Survival	Surv. Bonus	Bullet D...	Bullet Bo...	Ram Dmg.	Ram Bonus	1sts	2nds	3rds			
1	BotMeriang 1.0	1370	550	90	634	64	31	0	4	0	0			
2	ChillPips 1.0	1067	400	0	600	53	13	0	0	4	0			
3	KPR 1.0	154	150	0	0	0	4	0	0	0	3			
4	Ormas 1.0	55	50	0	1	0	4	0	0	0	1			

4.4. Hasil Eksperimen

Skor per Ronde					
Ronde	ChillPips	BotMeriang	Ormas	KPR	
score 1	981	907	147	54	
score 2	781	882	120	72	
score 3	1040	716	136	66	
score 4	1284	818	165	62	
score 5	1284	1085	164	107	
score 6	1136	911	41	151	
score 7	892	809	67	56	
score 8	827	1289	442	12	
score 9	1263	1132	122	100	
score 10	1067	1370	55	154	
Total Score	10555	9919	1459	834	

Tabel 1. Tabel Skor per Ronde

Skor Survival per Ronde				
Ronde	ChillPips	BotMeriang	Ormas	KPR
1	350	400	100	50
2	350	350	100	50
3	400	300	100	50
4	500	350	150	50
5	500	450	50	150
6	400	350	0	150
7	350	300	50	50
8	400	550	200	0
9	500	450	100	100
10	400	550	150	50
Total Score	4150	4050	1000	700

Tabel 2. Tabel Skor berdasarkan Skor Survival per Ronde

Bullet Damage per Ronde				
Ronde	ChillPips	BotMeriang	Ormas	KPR
1	528	422	13	0
2	352	434	14	14
3	504	394	23	12
4	604	398	2	8
5	644	522	15	9
6	608	442	13	0
7	444	432	2	12
8	404	542	14	5
9	616	562	3	0
10	400	550	0	1
Total Score	5104	4698	99	61

Tabel 3. Tabel Skor berdasarkan Bullet Damage per Ronde

Podium 1st per Ronde				
Ronde	ChillPips	BotMeriang	Ormas	KPR
1	2	2	0	0
2	2	1	0	0
3	3	0	0	0
4	4	0	0	0
5	2	2	0	0
6	3	1	0	0
7	2	1	0	0
8	0	4	0	0

9	2	2	0	0
10	0	4	0	0
Total Score	20	17	0	0

Tabel 4. Tabel Jumlah diraihnya Podium Pertama per Ronde

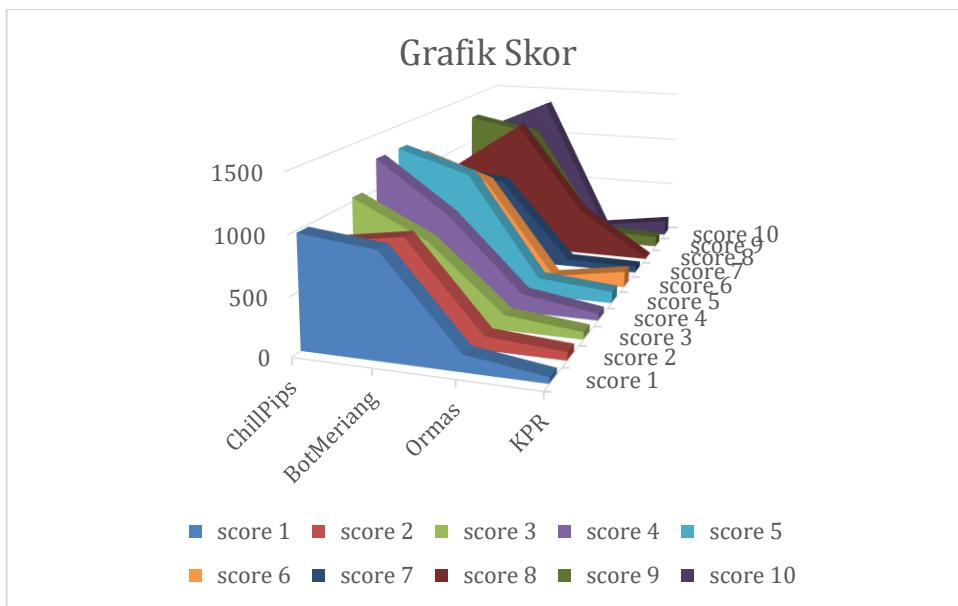
Podium 2nd per Ronde					
Ronde	ChillPips	BotMeriang	Ormas	KPR	
1	2	2	0	0	
2	1	2	0	0	
3	0	3	0	0	
4	0	4	0	0	
5	2	2	0	0	
6	1	3	0	0	
7	1	2	0	0	
8	3	0	1	0	
9	2	2	0	0	
10	4	0	0	0	
Total Score	16	20	1	0	

Tabel 5. Tabel Jumlah diraihnya Podium Kedua per Ronde

Podium 3rd per Ronde					
Ronde	ChillPips	BotMeriang	Ormas	KPR	
1	0	0	3	2	
2	0	0	2	1	
3	0	0	2	1	
4	0	0	3	1	
5	0	0	3	1	
6	0	0	3	1	
7	0	0	1	2	
8	1	0	3	0	
9	0	0	2	2	
10	0	0	1	3	
Total Score	1	0	23	14	

Tabel 6. Tabel Jumlah diraihnya Podium Ketiga per Ronde

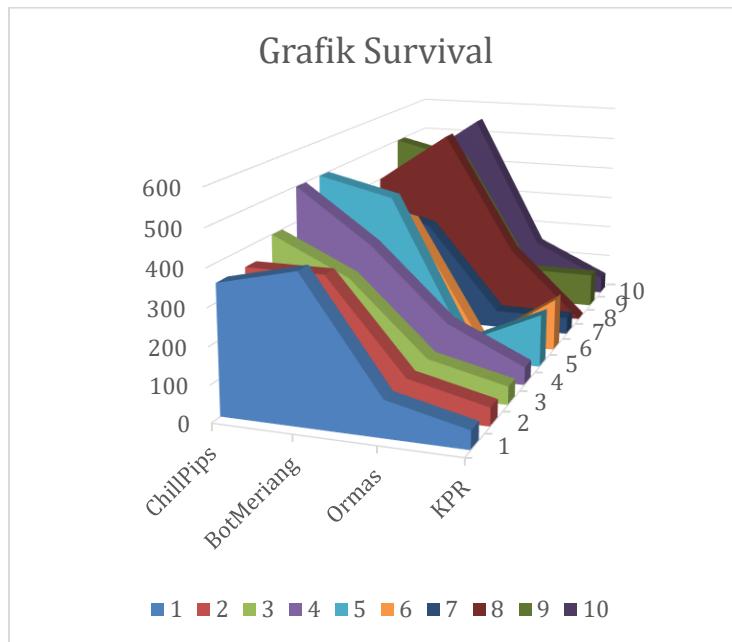
4.5. Analisis dan Pembahasan



Dengan meninjau tabel skor per ronde, dapat dilihat bahwa skor tertinggi diperoleh oleh Bot ChillPips dengan frekuensi tertinggi mendapatkan skor tertinggi tiap ronde. Selanjutnya, disusul dengan Bot BotMeriang, lalu Bot Ormas dan Bot KPR.

Kemenangan berturut-turut oleh Bot ChillPips disebabkan oleh skema algoritma Bot ChillPips yang mengutamakan survival (Greedy by Survival) dengan berada sejauh mungkin dari setiap bot setiap saat, yakni dengan bergerak sepanjang tembok, dan menembakkan peluru secara konstan. Pendekatan ini memungkinkan bot ChillPips untuk menghindari peluru dengan bergerak secara terus-menerus, sekaligus menyerang bot mushu yang berjarak cukup dekat. Dengan cara ini, bot ChillPips meningkatkan skornya secara konstan tanpa harus mengorbankan energi dengan mendekat dengan bot-bot musuh. Apabila dibandingkan dengan bot BotMeriang, bot ChillPips dapat memiliki kemungkinan lebih aman karena posisi bot ChillPips tidak berada pada area terbuka, sementara BotMeriang berada di area terbuka. Hal ini menyebabkan BotMeriang memiliki kemungkinan lebih besar untuk terkena peluru ataupun tertabrak dengan bot lain. Selanjutnya, bot Ormas memiliki pendekatan untuk menembak asal dengan mengikuti bot lain, tetapi cara ini memiliki kelemahan, yakni peluru yang ditembakkan tidak memiliki kemungkinan besar mengenai bot lain dan posisi bot tersebut berada di area terbuka sehingga memiliki kemungkinan perolehan skor dan kemungkinan bertahan hidup yang lebih kecil. Terakhir, karena bot KPR bergerak mendekati cluster yang berisi bot-bot lain yang memiliki energi yang sedikit dan menembak secara asal setelah sampai di titik yang diinginkan, maka

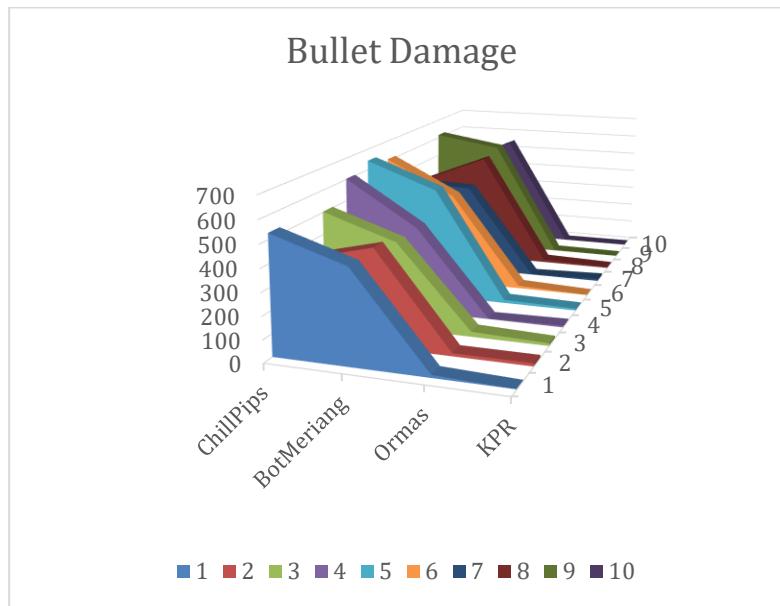
peluru bot KPR memiliki kemungkinan kecil untuk mengenai bot lain dan bot KPR berkemungkinan besar menjadi sasaran dari banyak bot lainnya sehingga perolehan skor dan kemungkinan bertahan hidup yang lebih kecil.



Dengan meninjau tabel skor survival per ronde, dapat dilihat bahwa skor survival tertinggi diperoleh oleh Bot ChillPips dengan frekuensi tertinggi mendapatkan skor tertinggi tiap ronde. Selanjutnya, disusul dengan Bot BotMeriang, lalu Bot Ormas dan Bot KPR.

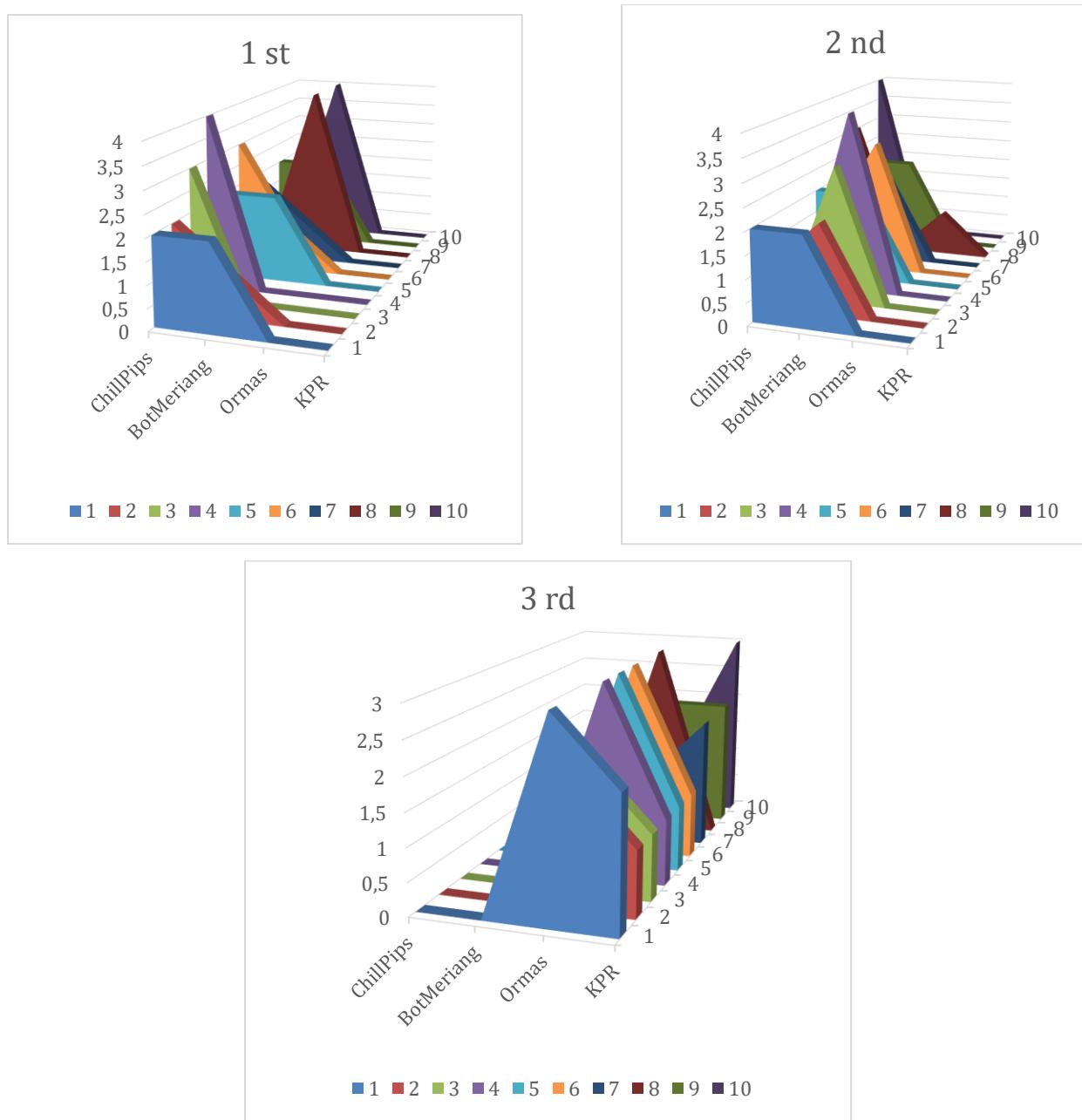
Kemenangan oleh bot ChillPips dipengaruhi oleh pemilihan skema algoritma pada bot ChillPips. Skema algoritma yang berpengaruh utama terhadap skor survival adalah pemilihan skema algoritma dengan mengikuti Greedy by Survival, yakni bergerak disamping tembok dan menembakkan musuh secara konstan. Hal ini meningkatkan kemungkinan bertahan hidup karena bot ChillPips menghindari konfrontasi langsung dengan bot lain, sekaligus memperoleh skor tambahan apabila peluru mengenai bot musuh sehingga memperbesar kemungkinan bertahan hidup. Selanjutnya, posisi skor dipegang oleh BotMeriang yang dipengaruhi utamanya oleh skema algoritma *Greedy by Survival*, yakni bergerak secara melingkar dengan menembak bot lain berdasarkan posisi yang diperoleh melalui radar. Hal ini memperbesar kemungkinan untuk bertahan hidup karena mengurangi kemungkinan terkena peluru meskipun berada pada area terbuka. Lalu, bot Ormas dan bot KPR tidak memiliki kemungkinan *survival* yang tinggi karena algoritma yang dipilih memaksa bot untuk bergerak mengikuti musuh tanpa memedulikan apakah akan memasuki area dengan bot yang lebih banyak atau tidak. Dengan begitu, keduanya

tidak memiliki survival rate yang baik, meskipun bot Ormas memiliki performa lebih baik karena selalu bergerak, sementara bot KPR berhenti ketika sudah mencapai titik yang ditentukan.



Selanjutnya, dengan meninjau besar kerusakan akibat peluru per ronde, dapat dilihat bahwa skor survival tertinggi diperoleh oleh Bot ChillPips dengan frekuensi tertinggi mendapatkan skor tertinggi tiap ronde. Selanjutnya, disusul dengan Bot BotMeriang, lalu Bot Ormas dan Bot KPR.

Keberhasilan dari Bot ChillPips dipengaruhi utamanya oleh penerapan skema algoritma, yakni ChillPips akan menembak bot dengan peluru dengan kerusakan yang lebih besar apabila terdapat musuh berada dalam jarak yang cukup dekat dengan bot ChillPips. Hal ini meningkatkan kemungkinan peluru untuk mengenai bot musuh dengan kerusakan yang tinggi. Hal yang sama juga terjadi untuk bot BotMeriang karena algoritma yang mirip, yakni menembak peluru dengan kerusakan yang lebih besar jika jarak sudah cukup dekat. Di sisi lain, bot Ormas dan bot KPR memiliki skema algoritma utama menembak secara acak sehingga tidak menjamin peluru mengenai setiap bot yang ditembak sehingga kerusakan akibat peluru semakin kecil. Dengan demikian, skema algoritma yang disarankan untuk meningkatkan kerusakan akibat peluru adalah menembak peluru dengan kerusakan yang lebih besar jika bot ChillPips dengan bot musuh berada dalam jarak tertentu.



Dengan demikian, berdasarkan keseluruhan pengujian tersebut, diperoleh bahwa bot ChillPips dan BotMeriang memiliki kemungkinan terbaik untuk menang karena menerapkan skema *greedy* bersama dengan kombinasi lain sedemikian sehingga memberikan hasil terbaik terhadap setiap parameter analisis.

BAB V

KESIMPULAN DAN SARAN

5.1. Kesimpulan

Berdasarkan hasil penggerjaan dan proses pengembangan bot ini, kami semakin memahami bagaimana cara kerja bot dalam . Kami juga mempelajari berbagai strategi yang dapat diterapkan, khususnya dalam menerapkan algoritma greedy untuk pengambilan keputusan secara optimal dalam kondisi permainan yang dinamis. Selain itu, kami memperoleh wawasan lebih dalam mengenai bagaimana bot dapat bergerak, menghindar, dan menyerang secara efisien dengan mempertimbangkan faktor seperti posisi lawan, energi, dan keadaan arena. Proses ini juga memberikan pemahaman lebih lanjut tentang implementasi Strategi algoritma untuk mengembangkan bot yang kompetitif dalam permainan strategi otomatis.

5.2. Saran

Program yang kami buat masih jauh dari sempurna. Untuk kedepannya, bisa lebih memperhatikan struktur program, flow program yang lebih jelas, dan juga menyisihkan waktu lebih banyak untuk bug fixing secara total.

5.3. Komentar

Kami mengucapkan terima kasih kepada asisten-asisten yang bertanggung jawab atas pelaksanaan tugas besar ini dan karena telah memberikan kesan yang terbaik untuk tugas pertama di Teknik Informatika ITB juga memberi gambaran akan bagaimana tugas-tugas besar lain yang akan dihadapi kedepannya. Kami juga mengucapkan terima kasih kepada mahasiswa Teknik Informatika ITB lain yang telah memberikan semangat dan dukungan sehingga pada akhirnya tugas besar ini dapat selesai dengan baik.

5.4. Refleksi

Tugas besar ini telah memberikan kami pengalaman yang sangat mengesankan dari senang hingga tekanan. Kami menghadapi berbagai macam kendala seperti alokasi waktu yang kurang efektif akibat kesibukannya masing-masing, belum terbiasa dengan bahasa java sendiri, dan kinerja dari kami sendiri yang kurang baik. Untuk kedepannya, diharapkan tugas besar ini bisa

menjadi motivasi untuk kami agar kami lebih berusaha kedepannya baik dalam pengerjaan tugas maupun menghadapi hidup

LAMPIRAN

Link Repository GitHub:

https://github.com/BrianHadianSTEI23/Tubes1_TankubanPrau.git

No	Poin	Ya	Tidak
1	Bot dapat dijalankan pada Engine yang sudah dimodifikasi asisten.	<input checked="" type="checkbox"/>	
2	Membuat 4 solusi greedy dengan heuristic yang berbeda.	<input checked="" type="checkbox"/>	
3	Membuat laporan sesuai dengan spesifikasi.	<input checked="" type="checkbox"/>	
4	Membuat video bonus dan diunggah pada Youtube.	<input checked="" type="checkbox"/>	

DAFTAR PUSTAKA

<https://robocode-dev.github.io/tank-royale/>

<https://robocode-dev.github.io/tank-royale/api/dotnet/api/Robocode.TankRoyale.BotApi.html>

[https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2024-2025/04-Algoritma-Greedy-\(2025\)-Bag1.pdf](https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2024-2025/04-Algoritma-Greedy-(2025)-Bag1.pdf)