

LAPORAN TUGAS BESAR I

IF2211 Strategi Algoritma

Pemanfaatan Algoritma Greedy dalam pembuatan bot permainan Robocode Tank Royale



Disusun oleh:

Muhammad Alfansya	13523005
Muh. Rusmin Nurwadin	13523068
Guntara Hambali	13523114

Program Studi Teknik Informatika

Sekolah Teknik Elektro dan Informatika

Institut Teknologi Bandung

2025

Daftar Isi

Daftar Isi.....	1
BAB I.....	3
DESKRIPSI TUGAS.....	3
1.1 Deskripsi Tugas.....	3
1.2 Spesifikasi.....	7
BAB II.....	9
LANDASAN TEORI.....	9
2.1 Dasar Teori Algoritma Greedy secara Umum.....	9
2.2 Elemen-elemen dalam Algoritma Greedy.....	9
2.2.1 Himpunan Kandidat (C).....	9
2.2.2 Himpunan Solusi (S).....	9
2.2.3 Fungsi Solusi.....	10
2.2.4 Fungsi Seleksi.....	10
2.2.5 Fungsi Kelayakan.....	10
2.2.6 Fungsi Objektif.....	10
2.3 Game Engine.....	10
2.4 Komponen Program Robocode Tank Royale.....	10
2.5 Cara Menjalankan Bot dan Implementasi Greedy pada Bot.....	10
2.6 Mekanisme Teknis Permainan Robocode Tank Royale.....	11
BAB III.....	12
APLIKASI STRATEGI GREEDY.....	12
3.1 Mapping.....	12
3.2 Eksplorasi Alternatif Solusi Greedy.....	12
3.2.1 Greedy by Safe Zone.....	12
3.2.2 Greedy by Fire.....	14
3.2.3 Greedy by Avoiding Enemies.....	15
3.2.4 Greedy by Ram.....	16
3.3 Strategi Greedy yang Dipilih.....	18
BAB IV.....	20
IMPLEMENTASI DAN PENGUJIAN.....	20
4.1 Pseudocode Algoritma Greedy.....	20
4.1.1 Greedy by Safe Zone.....	20
4.1.2 Greedy by Fire.....	24
4.1.3 Greedy by Avoiding Enemies.....	25
4.1.4 Greedy by Ram.....	27
4.2 Implementasi Algoritma Greedy Utama.....	27
4.2.1 Prosedur Utama.....	28
4.2.2 Procedure OnScannedBot.....	28
4.2.3 Prosedur OnHitByBullet.....	28

4.2.4 Prosedur TurnTo.....	29
4.3 Struktur Data Bot Utama (GreedyByFire).....	29
4.3.1 targetX dan targetY.....	29
4.3.2 targetSpeed, MaxSpeed, TurnRate, MaxTurnRate.....	29
4.3.3 Bullet (BulletState).....	30
4.3.4 Direction dan shooterDirection (double).....	30
4.4 Struktur File.....	30
4.5 Analisis dan Pengujian.....	31
4.5.1 Pengujian.....	31
4.5.2 Kondisi Kondisi Setiap Algoritma Bot.....	32
BAB V.....	35
KESIMPULAN DAN SARAN.....	35
5.1 Kesimpulan.....	35
5.2 Saran.....	35
LAMPIRAN.....	36
DAFTAR PUSTAKA.....	37

BAB I

DESKRIPSI TUGAS

1.1 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:

1. Rounds dan Turns

Pertempuran dapat terdiri dari beberapa rounds. Secara default, satu pertempuran berisi 10 rounds, di mana setiap rounds akan memiliki pemenang dan yang kalah. Setiap round dibagi menjadi beberapa turns, yang merupakan unit waktu terkecil. Satu turn adalah satu ketukan waktu dan satu putaran permainan. Jumlah turn dalam satu round tergantung pada berapa lama waktu yang dibutuhkan hingga hanya tersisa bot terakhir yang bertahan.

Pada setiap turn, sebuah bot dapat:

- Menggerakkan bot, memindai musuh, dan menembakkan senjata.
- Bereaksi terhadap peristiwa seperti saat bot terkena peluru atau bertabrakan dengan bot lain atau dinding.
- Perintah untuk bergerak, berputar, memindai, menembak, dan sebagainya dikirim ke server untuk setiap turn.

2. Batas Waktu Giliran

Penting untuk dicatat bahwa setiap bot memiliki batas waktu untuk setiap turn yang disebut turn timeout, biasanya antara 30-50 ms (dapat diatur sebagai aturan pertempuran). Ini berarti bahwa bot tidak bisa mengambil waktu sebanyak yang mereka inginkan untuk bergerak dan menyelesaikan turn saat ini. Setiap kali turn baru dimulai, penghitung waktu

ulang diatur ulang dan mulai berjalan. Jika batas waktu tercapai dan bot tidak mengirimkan pergerakannya untuk turn tersebut, maka tidak ada perintah yang dikirim ke server. Akibatnya, bot akan melewatkan turn tersebut. Jika bot melewatkan turn, ia tidak akan bisa menyesuaikan gerakannya atau menembakkan senjatanya karena server tidak menerima perintah tepat waktu sebelum turn berikutnya dimulai.

3. Energi

Semua bot memulai permainan dengan jumlah energi awal sebanyak 100 poin energi.

- Bot akan kehilangan energi jika ditembak atau ditabrak oleh bot musuh.
- Bot juga akan kehilangan energi jika menembakkan meriamnya.
- Bot akan mendapatkan energi jika peluru dari meriamnya mengenai musuh. Energi yang didapat akan lebih banyak 3 kali lipat dari energi yang digunakan untuk menembakkan peluru.
- Bot dengan energi nol akan dinonaktifkan dan tidak bisa bergerak. Jika bot terkena serangan dalam keadaan ini, bot akan hancur.

4. Peluru

Semakin banyak energi (daya tembak) yang digunakan untuk menembakkan peluru, semakin berat peluru tersebut dan semakin lambat gerakannya. Namun, peluru yang lebih berat juga menghasilkan lebih banyak kerusakan dan memungkinkan bot mendapatkan lebih banyak energi saat mengenai bot musuh.

Seperti disebutkan sebelumnya, peluru yang lebih berat akan bergerak lebih lambat. Ini berarti akan membutuhkan waktu lebih lama untuk mencapai target, meningkatkan risiko peluru tidak mengenai sasaran. Sebaliknya, peluru yang lebih ringan bergerak lebih cepat, sehingga lebih mudah mengenai target, tetapi peluru ringan tidak memberikan banyak poin energi saat mengenai bot musuh.

5. Panas Meriam (Gun Heat)

Saat menembakkan peluru, meriam akan menjadi panas. Peluru yang lebih berat menghasilkan lebih banyak panas dibandingkan peluru yang lebih ringan. Ketika meriam terlalu panas, bot tidak dapat menembak hingga suhu meriam turun ke nol. Selain itu, meriam juga sudah dalam keadaan panas di awal round dan perlu waktu untuk mendingin sebelum bisa digunakan untuk pertama kalinya.

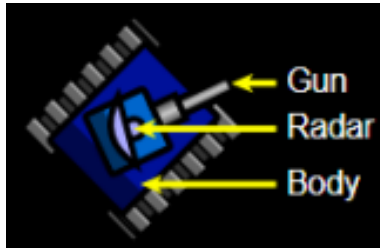
6. Tabrakan

Perlu diperhatikan bahwa bot akan menerima kerusakan jika menabrak dinding (batas arena), yang disebut wall damage. Hal yang sama juga terjadi jika bot bertabrakan dengan bot lain. Jika bot menabrak bot musuh dengan bergerak maju, ini disebut ramming

(menabrak dengan sengaja), yang akan memberikan sedikit skor tambahan bagi bot yang menyerang.

7. Bagian Tubuh Tank

Tubuh tank terdiri dari 3 bagian:



1. Body adalah bagian utama dari tank yang digunakan untuk menggerakkan tank.
2. Gun digunakan untuk menembakkan peluru dan dapat berputar bersama body atau independen dari body.
3. Radar digunakan untuk memindai posisi musuh dan dapat berputar bersama body atau independen dari body.

8. Pergerakan

Bot dapat bergerak maju dan mundur hingga kecepatan maksimum. Dibutuhkan beberapa giliran untuk mencapai kecepatan maksimum. Bot dapat mengalami percepatan maksimum sebesar 1 unit per giliran dan pengereman dengan perlambatan maksimum 2 unit per giliran. Percepatan dan perlambatan maksimum tidak bergantung pada kecepatan bot saat itu.

9. Berbelok

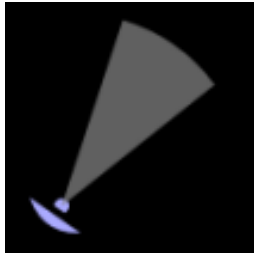
Seperti yang disebutkan sebelumnya, bagian tubuh, turret (meriam), dan radar dapat berputar secara independen satu sama lain. Jika turret atau radar tidak diputar, maka keduanya akan mengarah ke arah yang sama dengan tubuh bot.

Setiap bagian tubuh memiliki kecepatan putar yang berbeda. Radar adalah bagian tercepat dan dapat berputar hingga 45 derajat per giliran, yang berarti dapat berputar 360 derajat dalam 8 giliran. Turret dan meriam dapat berputar hingga 20 derajat per giliran.

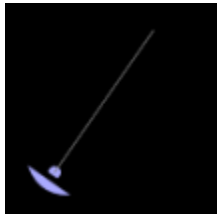
Bagian paling lambat adalah tubuh tank, yang dalam kondisi terbaik dapat berputar hingga 10 derajat per giliran. Namun, ini bergantung pada kecepatan bot saat ini. Semakin cepat bot bergerak, semakin lambat kemampuannya untuk berbelok. Perlu diperhatikan bahwa tidak ada energi yang dikonsumsi saat bot bergerak atau berbelok.

10. Pemindaian

Aspek penting dalam Robocode adalah memindai bot musuh menggunakan radar. Radar dapat mendeteksi bot dalam jangkauan hingga 1200 piksel. Musuh yang berada lebih dari 1200 piksel dari bot tidak dapat terdeteksi atau dipindai oleh radar. Penting untuk diperhatikan bahwa sebuah bot hanya dapat memindai bot musuh yang berada dalam jangkauan sudut pemindaian (scan arc)-nya. Sudut pemindaian ini merupakan "sapuan radar" dari arah radar sebelumnya ke arah radar saat ini dalam satu giliran.



Jika radar tidak bergerak dalam suatu giliran, artinya radar tetap mengarah ke arah yang sama seperti pada giliran sebelumnya, maka sudut pemindaian akan menjadi nol derajat, dan bot tidak akan dapat mendeteksi musuh.



Oleh karena itu, sangat disarankan untuk selalu mengubah arah radar agar tetap dapat memindai musuh.

11. Skor

Pada akhir pertempuran, setiap bot akan di-ranking berdasarkan total skor yang diperoleh masing-masing bot selama keseluruhan pertempuran. Tentunya, tujuan utama pada tugas besar ini adalah membuat bot yang memberikan skor setinggi mungkin. Berikut adalah rincian komponen skor pada pertempuran:

- **Bullet Damage:** Bot mendapatkan poin sebesar damage yang dibuat kepada bot musuh menggunakan peluru.
- **Bullet Damage Bonus:** Apabila peluru berhasil membunuh bot musuh, bot mendapatkan poin sebesar 20% dari damage yang dibuat kepada musuh yang terbunuh.
- **Survival Score:** Setiap ada bot yang mati, bot lainnya yang masih bertahan pada ronde tersebut mendapatkan 50 poin.
- **Last Survival Bonus:** Bot terakhir yang bertahan pada suatu ronde akan mendapatkan 10 poin dikali dengan banyaknya musuh.

- Ram Damage: Bot mendapatkan poin sebesar 2 kalinya damage yang dibuat kepada bot musuh dengan cara menabrak.
- Ram Damage Bonus: Apabila musuh terbunuh dengan cara ditabrak, bot mendapatkan poin sebesar 30% dari damage yang dibuat kepada musuh yang terbunuh.

Skor akhir bot adalah akumulasi dari 6 komponen diatas. Perlu diperhatikan bahwa game akan menampilkan berapa kali suatu bot meraih peringkat 1, 2, atau 3 pada setiap ronde. Namun, hal ini tidak dihitung sebagai komponen skor maupun untuk perangkingan akhir. Bot yang dianggap menang pertempuran adalah bot dengan akumulasi skor tertinggi.

Starter Pack

Untuk tugas besar ini, game engine yang akan digunakan sudah dimodifikasi oleh asisten. Berikut adalah beberapa perubahan yang dibuat oleh asisten dari game engine Tank Royale:

- Theme GUI: diubah menjadi light theme.
- Skor & Energi: masing-masing bot ditampilkan di samping arena permainan agar lebih mudah diamati saat pertarungan.
- Turn Limit: Durasi pertarungan tidak akan bergantung pada banyak ronde. Pada game engine ini, pertarungan akan berakhir apabila banyaknya turn sudah mencapai batas tertentu. Apabila batasan ini tercapai, ronde otomatis langsung berakhir dan pemenang pertarungan akan ditampilkan. Turn Limit dapat diatur pada menu “setup rules”.

1.2 Spesifikasi

Pada tugas ini kami diminta untuk membuat program sederhana dalam bahasa C# yang mengimplementasikan algoritma Greedy pada bot permainan Robocode Tank Royale dengan tujuan memenangkan permainan. Program perlu dibuat dengan spesifikasi sebagai berikut.

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.

Spesifikasi Bonus

- (maks 10) Membuat video tentang aplikasi greedy pada bot serta simulasinya pada game kemudian mengunggahnya di Youtube. Video dibuat harus memiliki audio dan menampilkan wajah dari setiap anggota kelompok. Untuk contoh video tubes stima tahun-tahun sebelumnya dapat dilihat di Youtube dengan kata kunci “Tubes Stima”, “strategi algoritma”, “Tugas besar stima”, dll. Semakin menarik video, maka semakin banyak poin yang diberikan.
- (maks 10) Beberapa kelompok pemenang lomba kompetisi akan mendapatkan nilai tambahan berdasarkan posisi yang diraih.

BAB II

LANDASAN TEORI

2.1 Dasar Teori Algoritma Greedy secara Umum

Secara bahasa, greedy berarti "tamak" atau "serakah". Algoritma greedy merupakan pendekatan penyelesaian masalah optimasi yang mengambil pilihan terbaik pada setiap langkah tanpa mempertimbangkan konsekuensi jangka panjang. Algoritma ini didasarkan pada pengambilan keputusan lokal optimal dengan keyakinan bahwa keputusan tersebut akan mengarah pada solusi global yang optimal. Pendekatan ini terbagi menjadi dua jenis persoalan: maksimasi dan minimasi.

Algoritma greedy memiliki beberapa komponen utama yang mendukung proses penyelesaian masalah. Komponen tersebut meliputi himpunan kandidat sebagai kumpulan pilihan yang tersedia, himpunan solusi yang merupakan pilihan yang telah diambil, fungsi solusi untuk mengevaluasi hasil, fungsi seleksi untuk memilih kandidat sesuai strategi, fungsi kelayakan untuk memeriksa kelayakan kandidat, dan fungsi objektif yang menentukan nilai optimasi.

Dalam implementasinya, algoritma greedy dapat diilustrasikan melalui persoalan penukaran uang kembalian. Pada kasus ini, algoritma akan selalu memilih koin dengan denominasi tertinggi yang masih dapat digunakan pada setiap iterasi. Misalnya, untuk kembalian Rp 7.500 dengan koin Rp 5.000, Rp 1.000, dan Rp 500, algoritma akan memilih satu koin Rp 5.000, dua koin Rp 1.000, dan satu koin Rp 500.

Algoritma greedy memiliki kelebihan berupa komputasi yang relatif cepat dibandingkan metode eksak, sehingga efisien untuk persoalan dengan ruang pencarian yang besar. Namun, algoritma ini juga memiliki keterbatasan yaitu tidak selalu menghasilkan solusi optimal tergantung karakteristik permasalahan, dan pembuktian optimalitasnya secara matematis seringkali cukup sulit. Meski demikian, algoritma greedy tetap menjadi salah satu algoritma berharga dalam mencari solusi aproksimasi dengan cepat untuk berbagai masalah optimasi dalam ilmu komputer.

2.2 Elemen-elemen dalam Algoritma Greedy

Algoritma Greedy terdiri dari beberapa komponen, antara lain:

2.2.1 Himpunan Kandidat (C)

Himpunan ini berisi kandidat yang akan dipilih pada setiap langkah. Sebagai contoh, himpunan kandidat seperti simpul/sisi dalam graf, koin, dan benda lainnya.

2.2.2 Himpunan Solusi (S)

Himpunan ini berisi kandidat yang sudah dipilih sebagai solusi.

2.2.3 Fungsi Solusi

Fungsi ini menentukan apakah himpunan kandidat yang dipilih sudah memberikan solusi yang diinginkan.

2.2.4 Fungsi Seleksi

Fungsi ini bertujuan dalam memilih kandidat berdasarkan strategi *greedy* tertentu. Strategi ini bersifat *heuristic*.

2.2.5 Fungsi Kelayakan

Fungsi ini memeriksa apakah kandidat yang dipilih dapat dimasukkan ke dalam himpunan solusi.

2.2.6 Fungsi Objektif

Fungsi ini bertujuan untuk memaksimumkan atau meminimumkan sesuatu yang menjadi tujuan dalam persoalan.

2.3 Game Engine

Robocode Tank Royale adalah game engine yang digunakan dalam tugas, merupakan evolusi dari Robocode asli di mana pemain membuat program bot tank virtual untuk berkompetisi satu sama lain. Untuk tugas besar ini, game engine tersebut telah dimodifikasi oleh asisten dengan perubahan berupa tampilan light theme, tampilan skor dan energi bot di samping arena, serta perubahan aturan pertarungan yang tidak lagi bergantung pada jumlah ronde melainkan jumlah turn yang dapat diatur batasnya. Pemain tidak memiliki kendali langsung selama permainan, hanya bertugas membuat program C# (.NET) dengan algoritma Greedy yang menentukan logika bot dalam bergerak, mendeteksi lawan, menembak, dan bereaksi terhadap kejadian selama pertempuran untuk mendapatkan skor tertinggi.

2.4 Komponen Program Robocode Tank Royale

Program ini terdiri atas *game engine* dan *bot starter pack*. Kedua komponen tersebut tersedia pada <https://github.com/Ariel-HS/tubes1-if2211-starter-pack>. Untuk menjalankan game engine digunakan `robocode-tankroyale-gui-0.30.0.jar`. Dalam *repository* yang sama tersedia *starter pack* lain yang dapat digunakan untuk menjalankan program. Sebagai contoh, tersedia file "TemplateBot.zip" yang menyediakan template dasar untuk membuat bot C# dengan struktur file yang diperlukan, termasuk kerangka kode bot, file konfigurasi JSON, dan file build (.csproj, .cmd, .sh).

2.5 Cara Menjalankan Bot dan Implementasi Greedy pada Bot

Untuk menjalankan bot:

1. Jalankan game engine, dengan perintah `java -jar robocode-tankroyale-gui-0.30.0.jar` pada cmd.



2. Setup konfigurasi bot root directory. Dimulai dengan klik tombol “Config”, lalu klik tombol “Bot Root Directories”, kemudian masukkan *directory* yang berisi folder-folder bot terkait.
3. Mulai battle. Dimulai dengan klik tombol “Battle”, lalu klik “Start Battle”. Nantinya akan tersedia bot-bot di dalam *directory* yang telah disetup.
4. Boot bot yang ingin dimainkan. Dimulai dengan memilih bot yang tersedia pada bagian kotak kiri-atas, lalu klik tombol “Boot”. Selanjutnya bot yang telah dipilih akan muncul pada kotak kanan-atas, bot yang berhasil di boot akan muncul pada kotak kiri-bawah. Terakhir tambahkan semua bot dalam permainan dengan menekan tombol “Add”, lalu mulai permainan dengan klik “Start Battle”.

Implementasi bot menggunakan bahasa C# dengan kerangka minimal berisi class yang mewarisi Bot, konstruktor yang memuat file JSON, dan metode Run(). Implementasi algoritma Greedy pada bot dapat dilakukan dengan mengoptimalkan keputusan setiap turn berdasarkan kondisi terkini seperti posisi musuh, energi tersisa, dan status gun heat untuk memaksimalkan skor dengan komponen-komponen seperti Bullet Damage, Survival Score, Ram Damage, dan lain sebagainya.

2.6 Mekanisme Teknis Permainan Robocode Tank Royale

Setiap bot memulai dengan 100 poin energi yang berkurang saat menembak atau terkena serangan dan bertambah saat peluru mengenai musuh (3x lipat energi yang digunakan). Peluru berat bergerak lebih lambat tapi menghasilkan damage lebih besar. Setelah menembak, meriam menjadi panas dan perlu waktu untuk dingin sebelum dapat menembak lagi. Bot menerima damage saat menabrak dinding atau bot lain. Pergerakan bot memiliki percepatan maksimum 1 unit dan perlambatan maksimum 2 unit per giliran. Pemindaian musuh dilakukan dengan radar yang memiliki jangkauan hingga 1200 piksel dan efektif saat bergerak. Setup permainan lebih lanjut akan disesuaikan pada waktu demo.

BAB III

APLIKASI STRATEGI GREEDY

3.1 Mapping

Proses mapping dalam persoalan Robocode Tank Royale menjadi elemen-elemen Algoritma Greedy meliputi himpunan kandidat, himpunan solusi, fungsi solusi, fungsi seleksi, fungsi kelayakan, dan fungsi objektif. Penjelasan lebih lanjut terdapat pada sub bab 3.2.

3.2 Eksplorasi Alternatif Solusi Greedy

Dalam menentukan solusi paling optimal untuk menyelesaikan persoalan, telah dilakukan percobaan dengan berbagai pendekatan berbeda. Algoritma-algoritma tersebut adalah sebagai berikut.

3.2.1 Greedy by Safe Zone

Greedy by Safe Zone adalah pendekatan greedy dengan memprioritaskan keamanan posisi bot berdasarkan beberapa pertimbangan, seperti jarak dari musuh dan dinding. Algoritma ini mencari posisi optimal di arena yang memberikan keuntungan strategis tertinggi berdasarkan beberapa parameter keamanan tertentu. Konsep “Safe Zone” dalam implementasi ini mencari kemungkinan posisi teraman dengan mempertimbangkan jarak dari dinding, jarak dari musuh, jarak dari pusat arena, dan jarak dari posisi saat ini. Selanjutnya diberikan beberapa optimalisasi pada bot, seperti cara menembak yang memperhitungkan jarak dan kecepatan musuh, pola berjalan zig-zag untuk menghindari potensi serangan, dan beberapa optimalisasi lain berkaitan dengan *event* yang mungkin dialami bot. Hal ini ditujukan untuk efektivitas bot yang lebih baik.

a. Mapping Elemen Greedy

1. Himpunan Kandidat : Semua posisi potensial di arena yang berjarak aman dari dinding, dengan jarak antar posisi x dan y (dalam implementasi ini x dan y bernilai 20).
2. Himpunan Solusi : Posisi teraman yang dipilih melalui evaluasi skor keamanan tertinggi.
3. Fungsi Solusi : Memeriksa apakah posisi yang dipilih memiliki skor keamanan tertinggi di antara semua kandidat posisi.
4. Fungsi Seleksi : Memilih posisi dengan skor keamanan tertinggi berdasarkan berbagai pertimbangan tertentu.
5. Fungsi Kelayakan : Memeriksa apakah posisi yang dipilih berada pada jarak aman dari dinding dan musuh. Dalam implementasi jarak aman dari dinding lebih besar dari 10 dan jarak aman dari musuh lebih besar dari 100 (ambang batas bersifat relatif, yang tertera adalah yang digunakan pada implementasi ini).

6. Fungsi objektif : Memaksimalkan skor keamanan posisi untuk keberlangsungan hidup bot dan kemampuan menyerang yang lebih efektif. Bot bertujuan utama mengincar survival skor dan survival bonus yang tinggi.

b. Analisis Efisiensi Solusi

Dalam implementasi Greedy by Safe Zone, untuk mencari posisi optimal diperlukan kompleksitas waktu $O(m \times n)$, dengan m dan n adalah jumlah interval grid pada sumbu x dan y . Untuk arena berukuran tertentu interval grid yang digunakan adalah 20, sehingga kompleksitasnya menjadi $O((width/20) \times (height/20))$. Algoritma ini melakukan iterasi pada semua posisi kandidat untuk mencari skor keamanan tertinggi.

Perhitungan skor keamanan untuk setiap kandidat memiliki kompleksitas $O(1)$ karena hanya melibatkan aritmatika sederhana. Sehingga. Kompleksitas algoritma keseluruhan untuk strategi ini adalah $O(m \times n)$.

c. Analisis Efektivitas Solusi

Greedy by Safe Zone berasal dari adanya kebutuhan bot untuk bertahan hidup sambil tetap efektif dalam menyerang. Namun pendekatan algoritma ini bukanlah solusi yang paling efektif dikarenakan terdapat faktor-faktor seperti pergerakan musuh yang tidak terprediksi, ukuran arena, dan kemungkinan serangan mendadak yang dapat mempengaruhi keamanan posisi bot.

Strategi ini efektif jika:

- Arena memiliki cukup ruang untuk bergerak dan menemukan posisi aman.
- Musuh tidak terlalu agresif dalam mengejar.
- Jumlah musuh tidak terlalu banyak.
- Bot dapat menembak dari jarak jauh dengan akurat.

Strategi ini tidak efektif jika :

- Arena sangat sempit sehingga hampir tidak ada zona aman.
- Musuh menggunakan strategi ram (menabrak) yang meminimalisir keuntungan posisional, atau strategi lain yang pendekatannya adalah memperpendek jarak dengan bot lain.
- Musuh memiliki algoritma prediksi pergerakan yang sangat akurat.
- Terlalu banyak musuh sehingga hampir tidak ada zona aman di arena.

Greedy by Safe Zone menawarkan keseimbangan yang baik antara bertahan dan menyerang, sehingga memungkinkan bot untuk bertahan hidup lebih lama dan mengakumulasi skor lebih tinggi dalam pertempuran jangka panjang. Meskipun tidak

selalu menghasilkan solusi optimal untuk semua skenario, strategi ini cukup adaptif terhadap berbagai kondisi pertempuran.

3.2.2 Greedy by Fire

Greedy by Fire adalah pendekatan greedy yang memprioritaskan efektivitas serangan bot berdasarkan kedekatan jarak dengan musuh. Algoritma ini mengutamakan musuh yang dalam jangkauan sebagai target utama. Selain itu, apabila ada bot lain yang menembak, bot ini juga akan langsung menghampiri bot yang menembak tersebut. Konsep tersebut dalam implementasi yang dilakukan akan mencari musuh dengan jarak terjangkau untuk diserang terlebih dahulu, mendekatinya, lalu memberikan tembakan dengan kekuatan maksimum dari jarak dekat.

a. Mapping Elemen Greedy

1. Himpunan kandidat : Semua musuh yang terdeteksi (akibat *scan* atau peluru musuh mengenai bot) di arena pertempuran dan pilihan jarak optimal untuk menyerang.
2. Himpunan solusi : Musuh yang terdeteksi sebagai target dan jarak optimal (*distance* - 50).
3. Fungsi solusi : Memeriksa apakah bot telah mencapai jarak optimal untuk menyerang.
4. Fungsi Seleksi : Memilih musuh yang terdeteksi radar sebagai target atau musuh yang terdeteksi telah menembak kita, lalu mendekat hingga jarak optimal.
5. Fungsi kelayakan: Memeriksa apakah strategi serangan yang dipilih dapat dieksekusi (musuh dalam jangkauan dan bot memiliki energi cukup).
6. Fungsi objektif: Memaksimalkan kerusakan yang diberikan pada musuh sambil memprioritaskan musuh terdekat untuk efisiensi serangan. Mengoptimalkan poin dan bonus *bullet damage*.

b. Analisis Efisiensi Solusi

Dalam pengaplikasiannya, algoritma memiliki kompleksitas waktu $O(1)$ untuk pengambilan keputusan serangan terhadap musuh yang terdeteksi. Tidak perlu melakukan perbandingan antara beberapa musuh karena algoritma hanya merespons musuh yang saat ini terdeteksi dengan radar.

Strategi ini sangat efisien dari segi komputasi, namun memiliki keterbatasan dalam hal strategis karena hanya mempertimbangkan musuh yang sedang terdeteksi tanpa mempertimbangkan posisi relatif musuh lain yang mungkin lebih berbahaya atau lebih strategis untuk diserang.

c. Analisis Efektivitas Solusi

Greedy by Fire memiliki kelebihan dan kekurangan yang perlu dipertimbangkan. Analisis sebagai berikut.

Strategi ini efektif jika :

- Pertempuran cenderung terjadi pada jarak dekat, di mana tembakan kekuatan tinggi sangat efektif.
- Jumlah musuh terbatas sehingga fokus pada satu musuh dalam satu waktu adalah strategi optimal.
- Arena pertempuran relatif kecil, meningkatkan kemungkinan pertempuran jarak dekat.
- Musuh memiliki pergerakan lambat atau dapat diprediksi.
- Bot memiliki pergerakan yang cukup gesit untuk mengatasi kelemahan mendekati musuh.

Strategi ini tidak efektif jika :

- Musuh memiliki pola pergerakan yang sangat acak dan cepat, membuat tembakan sulit mengenai target.
- Terdapat banyak musuh di arena, sehingga fokus pada satu musuh terdekat mungkin bukan strategi optimal.
- Arena pertempuran sangat luas, menyulitkan untuk mempertahankan jarak dekat dengan musuh.
- Musuh menggunakan strategi defensif yang secara khusus menghindari pertempuran jarak dekat.

3.2.3 Greedy by Avoiding Enemies

Greedy ini menggunakan pendekatan pergerakan terus menerus (dinamis) untuk terus menghindari dari lawan. Algoritma ini akan menentukan pergerakan bot dengan mempertimbangkan jarak dan energi lawan. Bot dapat bersikap agresif sambil tetap mempertahankan aspek defensif. Bot akan menyerang lawan berdasar jarak dan kecepatan pergerakannya. Bot akan menghindari dan terus mereposisi saat bertemu dengan lawan dan akan bersikap lebih pasif saat berada cukup jauh dari bot lain.

a. Mapping Elemen Greedy

1. Himpunan kandidat : Semua bot yang tersedia di arena.
2. Himpunan solusi : Posisi ketika bot tidak terlalu dekat dengan lawan.
3. Fungsi solusi : Memeriksa apakah bot berada dekat dengan lawan.

4. Fungsi Seleksi : Memilih arah gerak untuk menghindar berdasar jarak dengan lawan dan energi lawan.
5. Fungsi kelayakan: Memeriksa apakah lawan yang berada di sekitar masuk dalam jangkauan jarak minimum.
6. Fungsi objektif: Memaksimalkan jarak posisi bot terhadap posisi lawan di sekitar sambil tetap memaksimalkan skor bullet damage yang bisa di dapat.

b. Analisis Efisiensi Solusi

Algoritma Greedy by Avoiding Enemies memerlukan perhitungan jarak terus-menerus dan meng-iterasi penyesuaian posisi berdasarkan kondisi arena dan kondisi lawan. Algoritma ini memiliki kompleksitas $O(1)$ karena perlu menilai jarak terhadap lawan yang discan dan memutuskan aksi yang akan dilakukan selanjutnya.

c. Analisis Efektivitas Solusi

Strategi greedy by avoiding enemies memiliki kondisi dimana strategi itu lebih efektif dibanding kondisi lainnya.

Efektif :

- Jumlah bot di arena tidak terlalu ramai sehingga bot tidak bingung mencari posisi.
- Lawan saling bertarung di tempat sempit sehingga bot bisa menunggu di jarak yang cukup jauh.
- Melawan bot dengan akurasi yang buruk

Tidak Efektif :

- Melawan bot pengejar yang agresif
- Jumlah bot di arena banyak sehingga tidak ada tempat untuk pergi dan kemungkinan tertembak menjadi tinggi
- Spawn di titik crowded di awal ronde

3.2.4 Greedy by Ram

Greedy by Ram adalah pendekatan greedy yang memprioritaskan ram sebagai cara melakukan serangan. Algoritma ini mengutamakan musuh yang memiliki energi rendah. Algoritma ini akan mendeteksi musuh dengan cara *scanning* dan ketika di-*hit* oleh peluru musuh. Konsep tersebut dalam implementasi dilakukan dengan mencari musuh dengan energi rendah terlebih dahulu, lalu menabrak dengan kecepatan penuh.

a. Mapping Elemen Greedy

1. Himpunan kandidat : Semua musuh yang terdeteksi (akibat *scan* atau peluru musuh mengenai bot) di arena pertempuran.
2. Himpunan solusi : Musuh yang terdeteksi sebagai target dan memiliki energi di bawah 70.
3. Fungsi solusi : Memeriksa apakah bot musuh telah memiliki energi di bawah 70.
4. Fungsi Seleksi : Memilih bot musuh yang energinya kurang dari batas tertentu (dalam implementasi ini energi musuh kurang dari 70).
5. Fungsi kelayakan: Memeriksa apakah strategi serangan yang dipilih dapat dieksekusi (musuh dalam jangkauan dan bot memiliki energi cukup).
6. Fungsi objektif: Memaksimalkan kerusakan tabrakan terhadap musuh, hal ini bertujuan untuk memperoleh poin ram damage dan ram bonus.

b. Analisis Efisiensi Solusi

Dalam pengaplikasian Greedy Ram Revenge, algoritma memiliki kompleksitas waktu $O(1)$ untuk pengambilan keputusan target dan pergerakan. Algoritma ini membuat keputusan langsung berdasarkan kondisi saat ini (ada/tidak adanya target dan kondisi tabrakan) tanpa memerlukan iterasi kompleks.

Strategi ini sangat efisien dari segi komputasi dan memungkinkan respons cepat terhadap perubahan dalam pertempuran, terutama saat bot tertembak dan perlu membalas dengan cepat.

c. Analisis Efektivitas Solusi

Greedy by Ram memiliki kelebihan dan kekurangan yang perlu dipertimbangkan. Analisis sebagai berikut.

Strategi ini efektif jika :

- Pertempuran cenderung terjadi pada jarak dekat sehingga mudah untuk *lock* musuh
- Musuh tidak melakukan greedy by fire karena ketika bot mendekat, musuh jadi lebih mudah menembak secara akurat.
- Arena memiliki ruang gerak yang cukup untuk melakukan tabrakan.
- Bot memiliki energi yang cukup untuk bertahan dari serangan saat bergerak menuju target.

Strategi ini tidak efektif jika :

- Musuh memiliki energi yang tinggi, membuat resiko tabrakan beresiko tinggi bagi bot.
- Musuh memiliki kemampuan menghindar yang baik.
- Terdapat banyak musuh yang menyerang secara bersamaan, membuat strategi kurang optimal.
- Musuh menggunakan strategi greedy by fire karena ketika bot mendekat, musuh akan lebih mudah menembak bot secara akurat

3.3 Strategi Greedy yang Dipilih

Pada akhirnya, kami memilih greedy by fire sebagai algoritma utama untuk bot kami. Hal ini didasarkan pada argumen bahwa strategi fire menghasilkan poin paling besar dibandingkan dengan komponen fungsi objektif lain. Selain itu, penerapannya mudah serta tidak memerlukan struktur data yang terlalu kompleks. Hal ini didukung dengan pengujian bot sebanyak lima kali dengan kondisi setup default. Dari lima kali pengujian (**Bab 4 bagian 4.5.1**) tersebut, dapat disimpulkan bahwa Greedy by Fire memiliki tingkat kemenangan paling tinggi, yakni lima kali kemenangan dari total lima kali pengujian. Maka dari itu, strategi greedy by fire dipilih menjadi strategi untuk bot utama nantinya.

BAB IV

IMPLEMENTASI DAN PENGUJIAN

4.1 Pseudocode Algoritma Greedy

4.1.1 Greedy by Safe Zone

```
Greedy by Safe Zone
procedure GreedyBySafeZone()

Deklarasi

    lastScannedBot: ScannedBotEvent?
    wallDist: const = 10
    enemyDist: const = 100
    currentSafezone: Position? = null
    cekZone: integer = 0
    needMove: boolean = true
    gunDir: integer = 1
    enemyX: double = -1
    enemyY: double = -1
    scanTime: integer = -1
    rand: Random

    {Position dan ScannedBotEvent adalah nullable type}

Algoritma

    {Inisialisasi warna bot}

    while (IsRunning) do
        if (needMove) then
            FindZone()
            GoToZone()
        endif

        TurnGunRight(10 * gunDir) {ganti arah}

        if sudah 11 turn then
            ganti arah putar
        endif

        if tidak melihat musuh ketika sudah 10 turn then
            putar radar 360 derajat
        endif
    endwhile
```

```
function FindZone() -> Position?
{mencari zona aman terbaik dengan pendekatan greedy}
```

Deklarasi

```
    candidates : List<Position>
```

Algoritma

```
    candidates <- GenPos() {Generate semua posisi kandidat}
    return SelectBest(candidates)
```

```
function GenPos() → List<Position>
{Menghasilkan himpunan kandidat posisi dalam arena}
```

Deklarasi

```
    candidates : List<Position>
```

Algoritma

```
    candidates ← {}
    for tiap posisi x dengan interval 20 do {10,30,dst.}
        for tiap posisi y dengan interval 20 do
            candidates ← Postition(x, y)
        endfor
    endfor

    return candidates
```

```
function IsSafe(pos: Position) -> boolean
{Fungsi kelayakan untuk menentukan apakah posisi aman}
```

Deklarasi

```
    distanceToWall : double
    distanceToEnemy : double
```

Algoritma

```
    distanceToWall ← jarak ke dinding
    if (jarak ke dinding kurang dari 10) then
        return false
    endif
```

```

if (waktu scan > 0 and TurnNumber - waktu scan < 30)
then
    if (jarak ke musuh kurang dari 100) then
        return false
    endif
endif

return true

```

```

function CalcScore(pos: Position) → double
{Fungsi seleksi untuk mengevaluasi posisi}

```

Deklarasi

```

score : double
distanceToWall : double
distanceToEnemy : double
distanceToCenter : double
distanceFromCurrent : double

```

Algoritma

```

score ← 100 {skor dasar}

distanceToWall ← Jarak ke dinding

score ← score bertambah tergantung jarak dari dinding

if (scanTime > 0 and TurnNumber - scanTime < 30) then
    distanceToEnemy ← Jarak ke musuh
    score ← score bertambah tergantung jarak ke musuh
endif

distanceToCenter ← jarak ke pusat arena
score ← score bertambah tergantung jarak ke pusat arena

distanceFromCurrent ← jarak new safe zone ke safe zone
sekarang

if (distanceFromCurrent > 250) then
    score ← score bertambah tergantung jarak dari safe
    zone sekarang
endif

return score

```

```

function SelectBest(candidates: List<Position>) →

```

```
Position?  
{Memilih posisi terbaik berdasarkan skor tertinggi}
```

Deklarasi

```
bestPosition : Position?  
bestScore : double
```

Algoritma

```
bestPosition ← null  
bestScore ← MinValue {asumsikan nilai yang sangat kecil}  
  
for each pos in candidates do  
  if (IsSafe(Pos)) then  
    score ← CalcScore(pos)  
    if (score > bestScore) then  
      bestScore ← score  
      bestPosition ← pos  
    endif  
  endif  
endfor  
  
return bestPosition
```

```
procedure GoToZone()
```

Deklarasi

```
bearing : double  
moveDist : double
```

Algoritma

```
if (tidak ada posisi aman) then  
  return  
  
bearing <- Putar ke arah safe zone  
SetTurnRight(bearing)  
Go()  
  
while (distanceToSafeZone > 0) do  
  bergerak zig-zag ke safe zone  
endwhile
```

```
{fungsi/prosedur lain}
```

```

function Attack(e: ScannedBotEvent)
{strategi tambahan untuk penyerangan, result fire}

procedure OnScannedBot(ScannedBotEvent e)
{menangani ketika radar melakukan scan}

procedure OnHitByBullet(HitByBulletEvent e)
{menangani ketika bot terkena peluru}

procedure OnHitWall (HitWallEvent e)
{menangani ketika bot menabrak dinding}

procedure OnHitBot (HitBotEvent e)
{menangani ketika bot menabrak bot lain}

procedure OnBulletHit (BulletHitBotEvent e)
{menangani ketika peluru mengenai bot lain}

```

4.1.2 Greedy by Fire

```

Greedy by Fire
procedure GreedyByFire()

Deklarasi

Algoritma

    while (IsRunning) do
        if (tidak ada target) then
            Scan berputar
        else
            Tembak, lalu scan ulang
        endif

```

```

procedure OnScannedBot(ScannedBotEvent e)

Deklarasi

Algoritma
    Arahkan body bot ke target
    Maju dengan kecepatan maksimum hingga jarak 50 dari
    musuh

```

```

procedure OnHitByBullet(HitByBulletEvent e)

```


Deklarasi**Algoritma**

Arahkan body bot ke arah peluru ditembakkan
Maju dengan kecepatan maksimum hingga jarak 50 dari musuh

4.1.3 Greedy by Avoiding Enemies

```
procedure GreedyByAvoidingEnemies()
```

Deklarasi

maju : boolean

Algoritma

maju <- true

While (IsRunning) **do**

If (maju) **then**

 Scan sekitar

 Bot gerak maju berpola belok-belok

endif

```
procedure OnScannedBot(enemy)
```

Deklarasi**Algoritma**

If (enemyDistance > 550) **then**

 Set MaxSpeed = 2

If (enemy.Speed < 3) **then**

 Arahkan turret ke lawan

 tembak dengan firepower = 1

endif

else if (enemyDistance < 450) **then**

 Set MaxSpeed = 4

if(kecepatan enemy < 4) **then**

 Arahkan turret ke lawan

 Tembak dengan firepower = 1

endif

else

if(enemyDistance < 300) **then**

 Arahkan turret ke lawan

if(enemyDistance < 200) **then** tembak dengan
firepower 3

else tembak dengan firepower 1

```

        {jika energi musuh kurang dari bot, melambat agar
        menembak lebih akurat}
        if(Energy enemy < Energy sendiri) then
            Set MaxSpeed = 3
            If (enemyDistance < 100) then Belok Kanan
            Set MaxSpeed = 6
        Endif

        {Jika energi musuh lebih banyak}
        else
            Set MaxSpeed = 10
            Belok Kiri
        endif
    endif

```

procedure OnHitWall()

Deklarasi

Algoritma

```

    Set MaxSpeed = 9
    Belok ke kiri sejauh 10 derajat
    ReverseDirection()

```

procedure OnHitBot(enemy)

Set MaxSpeed = 10

If (enemy.IsRammed) **then** ReverseDirection()

procedure ReverseDirection()

{ubah arah gerak}

Deklarasi

maju : boolean

Algoritma

```

    If (maju) then
        Mundur sejauh 40000
    else
        Maju sejauh 40000
    endif

```

Procedure OnHitByBullet(enemy)

Deklarasi

Algoritma

```

    Set MaxSpeed = 8
    Belok kekiri sejauh tegak lurus dari arah datang peluru

```

4.1.4 Greedy by Ram

```
Greedy by Ram  
procedure GreedyByRam
```

Deklarasi

Algoritma

```
    while (IsRunning) do  
        if (tidak ada target) then  
            Scan berputar  
        else  
            Scan ulang  
        endif  
    endwhile
```

```
procedure OnScannedBot(ScannedBotEvent e)
```

Deklarasi

Algoritma

```
    if (Energi musuh < 70) then  
        Ram musuh dengan kecepatan maksimum  
    endif
```

```
procedure OnHitByBullet(HitByBulletEvent e)
```

Deklarasi

Algoritma

```
    if (Energi musuh < 70) then  
        Ram musuh dengan kecepatan maksimum  
    else  
        Maju 100 untuk menghindari peluru lanjutan  
    endif
```

4.2 Implementasi Algoritma Greedy Utama

Berikut implementasi dari greedy by fire yang dipilih menjadi strategi utama yang digunakan.

4.2.1 Prosedur Utama

Prosedur ini merupakan bagian utama yang akan menjalankan algoritma greedy by fire

```

Greedy by Fire
procedure GreedyByFire()

Deklarasi
    targetX, targetY: double
Algoritma

    while (IsRunning) do
        if (targetX == -1 || targetY == -1) then
            SetTurnRight(10)
            SetRescan()
            Go()
        else
            Fire(3) {Tembak, lalu scan ulang}
            Rescan()

            targetX = -1 {jika tidak ada yg ter-scan lagi}
            targetY = -1

```

4.2.2 Procedure OnScannedBot

Prosedur ini berfungsi untuk mendeteksi musuh melalui *scan*

```

procedure OnScannedBot(ScannedBotEvent e)

Deklarasi
    distance, TargetSpeed: double
Algoritma
    distance <- DistanceTo(e.X, e.Y)
    TurnTo(BearingTo(e.X, e.Y)) {arahkan ke target}
    Go()

    TargetSpeed <- MaxSpeed
    SetForward(distance - 50) {maju dgn kec max hingga
    jarak 50 unit dari musuh}
    Go()

```

4.2.3 Prosedur OnHitByBullet

Prosedur ini berfungsi untuk mendeteksi musuh melalui peluru yang mengenai bot kita. Sehingga bot bisa membalas serangan yang diterimanya.

```

procedure OnHitByBullet(HitByBulletEvent e)
Deklarasi

```

```

    Bullet: BulletState
    shooterDirection, TurnRate, TargetSpeed: double
Algoritma
    BulletState bullet <- e.Bullet
    shooterDirection <- (bullet.Direction + 180) % 360
    {Ngadep ke arah peluru ditembakin}
    TurnRate <- MaxTurnRate
    TurnTo(shooterDirection)
    Go()

    TargetSpeed <- MaxSpeed    {Maju dgn kec max}
    SetForward(100)
    Go()

```

4.2.4 Prosedur TurnTo

Prosedur ini berfungsi untuk mengarahkan bot menuju target yang diinginkan.

```

procedure TurnTo(double angle)
Deklarasi
    turnAngle, Direction: double

Algoritma
    turnAngle = angle - Direction
    if (turnAngle > 180)
        turnAngle -= 360
    if (turnAngle < -180)
        turnAngle += 360

    SetTurnLeft(turnAngle)

```

4.3 Struktur Data Bot Utama (GreedyByFire)

4.3.1 targetX dan targetY (double)

Struktur data ini berfungsi untuk menyimpan absis dan ordinat posisi musuh. Apabila musuh terdeteksi, struktur data ini akan menyimpan posisi musuh. Sementara itu, jika musuh tak terdeteksi, struktur data ini akan bernilai -1.

4.3.2 targetSpeed, MaxSpeed, TurnRate, MaxTurnRate (double)

Struktur data ini merupakan konstanta umum dalam program. TargetSpeed dan TurnRate berturut-turut merupakan kecepatan yang ingin dicapai bot dan kecepatan putaran bot. Sementara itu, MaxSpeed dan MaxTurnRate berturut-turut merupakan kecepatan maksimal bot dan kecepatan putaran *body* bot yang maksimal.

4.3.3 Bullet (BulletState)

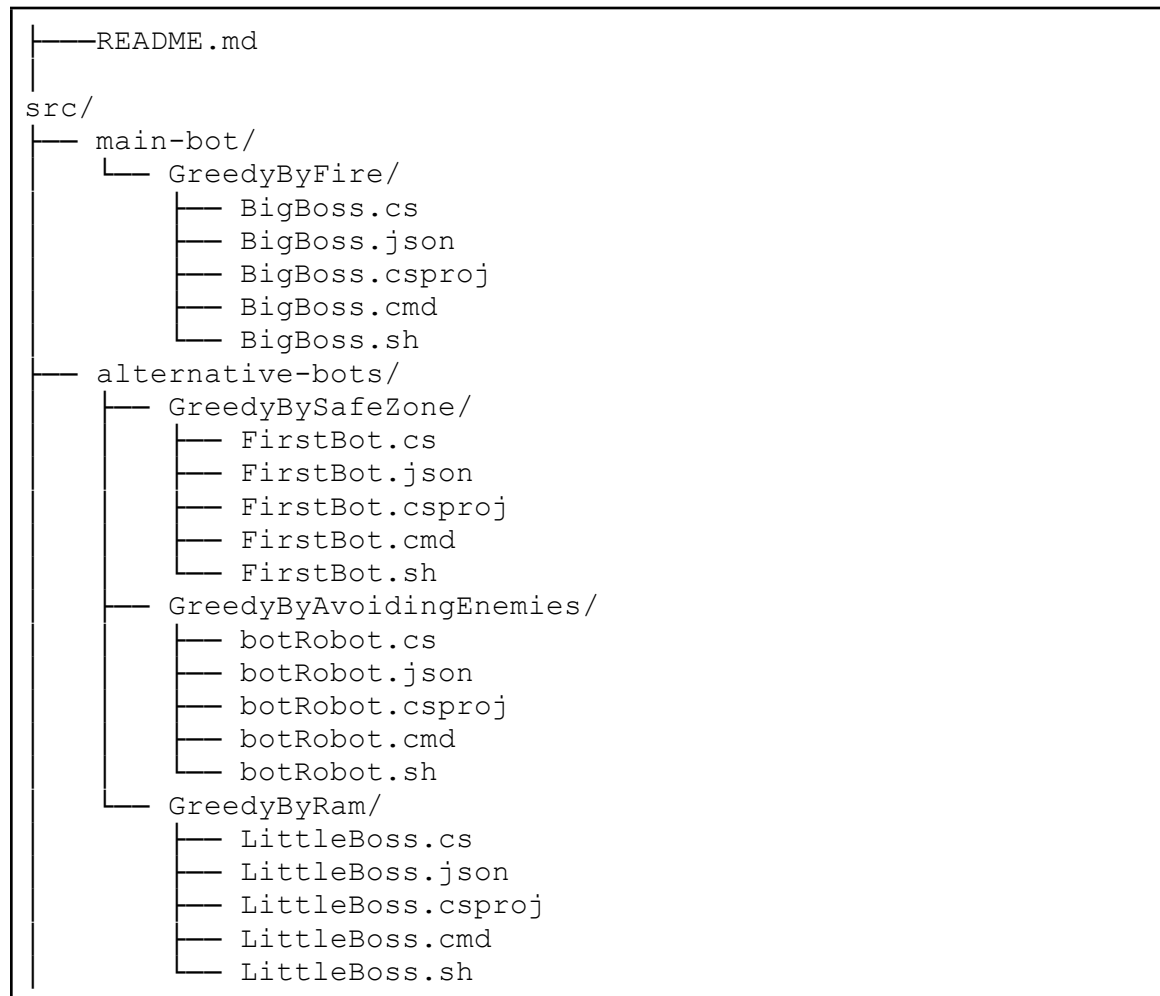
Struktur data ini berfungsi untuk menyimpan data dari peluru yang mengenai bot. Data dari peluru yang dapat diambil mencakup posisi awalnya (absis dan ordinat) serta arah rambat peluru

4.3.4 Direction dan shooterDirection (double)

Struktur data ini berturut-turut berfungsi untuk menyimpan arah *body* bot saat ini dan arah bot musuh.

4.4 Struktur File

Bot yang digunakan dalam permainan Robocode Tank Royale dikembangkan dalam bahasa C#. Program bot memiliki struktur file kurang lebih sebagai berikut.



Folder `src` merupakan folder utama yang berisi semua *source code* dari bot yang ada. Folder `main-bot` berisi bot utama yang digunakan dalam kompetisi. Bot ini dianggap memiliki

implementasi greedy yang paling baik. Folder alternative-bots berisi tiga bot lain dengan implementasi strategi greedy berbeda-beda.

Setiap Folder strategi greedy (GreedBy...) memiliki struktur file yang terdiri atas `.cs`, `.json`, `.csproj`, `.cmd`, dan `.sh`.

[NamaBot].cs adalah file source code C# yang berisi implementasi logika bot. File ini mendefinisikan kelas bot, yang mewarisi dari kelas `Bot` dasar dari Robocode Tank Royale API. File ini berisi seluruh logika untuk memindai, bergerak, menembak, dan merespons peristiwa game.

[NamaBot].json adalah file konfigurasi JSON yang berisi metadata bot, seperti nama, versi, pembuat, dan deskripsi. File ini diperlukan oleh game engine untuk mendaftarkan dan mengidentifikasi bot.

[NamaBot].csproj adalah file proyek .NET yang mendefinisikan bagaimana bot dikompilasi. File ini menentukan framework .NET yang digunakan, versi bahasa C#, namespace, dan referensi paket yang diperlukan (terutama `Robocode.TankRoyale.BotApi`).

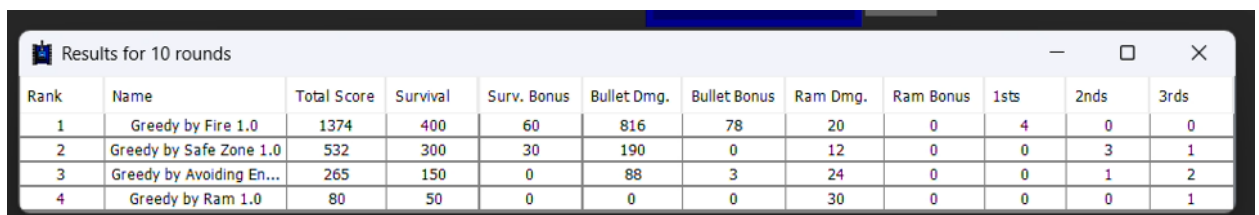
[NamaBot].cmd adalah script batch Windows yang digunakan untuk membangun dan menjalankan bot pada sistem operasi Windows. Script ini biasanya melakukan pembersihan, pembangunan, dan menjalankan bot.

[NamaBot].sh adalah script shell untuk sistem Unix-like (Linux/Mac) yang melakukan fungsi yang sama dengan file `.cmd`, tetapi untuk lingkungan non-Windows.

4.5 Analisis dan Pengujian

4.5.1 Pengujian

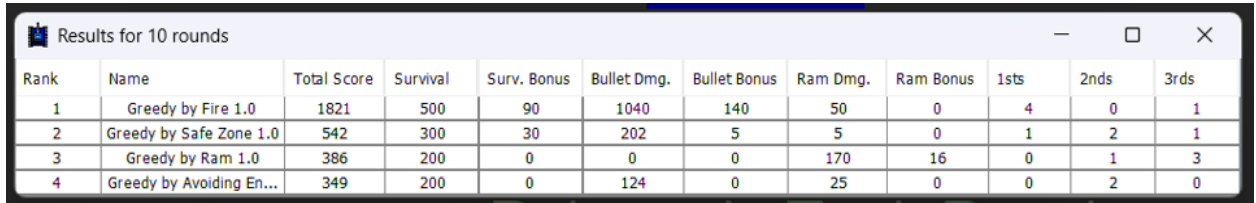
1. Pengujian pertama



Rank	Name	Total Score	Survival	Surv. Bonus	Bullet Dmg.	Bullet Bonus	Ram Dmg.	Ram Bonus	1sts	2nds	3rds
1	Greedy by Fire 1.0	1374	400	60	816	78	20	0	4	0	0
2	Greedy by Safe Zone 1.0	532	300	30	190	0	12	0	0	3	1
3	Greedy by Avoiding En...	265	150	0	88	3	24	0	0	1	2
4	Greedy by Ram 1.0	80	50	0	0	0	30	0	0	0	1

Gambar 4.5.1.1 Pengujian pertama

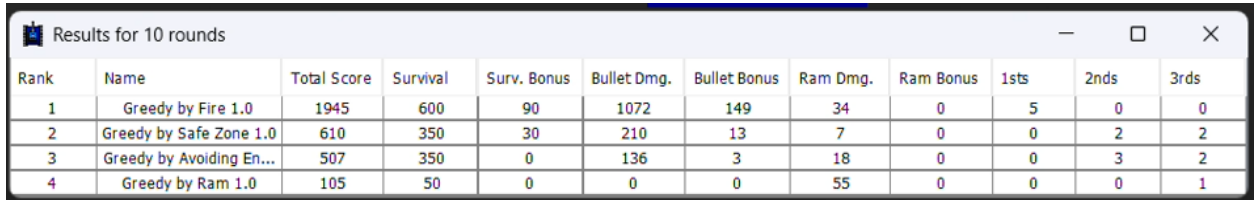
2. Pengujian kedua



Rank	Name	Total Score	Survival	Surv. Bonus	Bullet Dmg.	Bullet Bonus	Ram Dmg.	Ram Bonus	1sts	2nds	3rds
1	Greedy by Fire 1.0	1821	500	90	1040	140	50	0	4	0	1
2	Greedy by Safe Zone 1.0	542	300	30	202	5	5	0	1	2	1
3	Greedy by Ram 1.0	386	200	0	0	0	170	16	0	1	3
4	Greedy by Avoiding Enemies 1.0	349	200	0	124	0	25	0	0	2	0

Gambar 4.5.1.2 Pengujian kedua

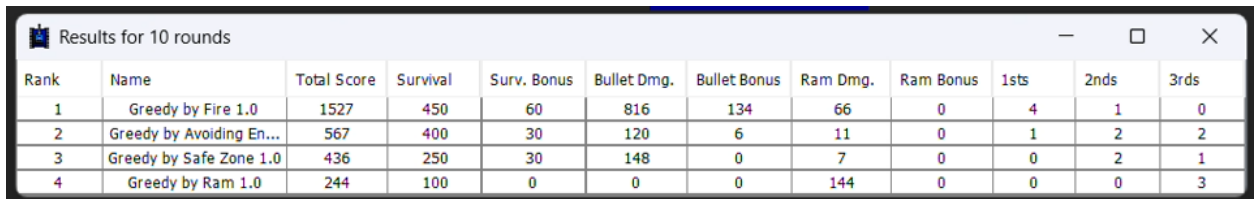
3. Pengujian ketiga



Rank	Name	Total Score	Survival	Surv. Bonus	Bullet Dmg.	Bullet Bonus	Ram Dmg.	Ram Bonus	1sts	2nds	3rds
1	Greedy by Fire 1.0	1945	600	90	1072	149	34	0	5	0	0
2	Greedy by Safe Zone 1.0	610	350	30	210	13	7	0	0	2	2
3	Greedy by Avoiding Enemies 1.0	507	350	0	136	3	18	0	0	3	2
4	Greedy by Ram 1.0	105	50	0	0	0	55	0	0	0	1

Gambar 4.5.1.3 Pengujian ketiga

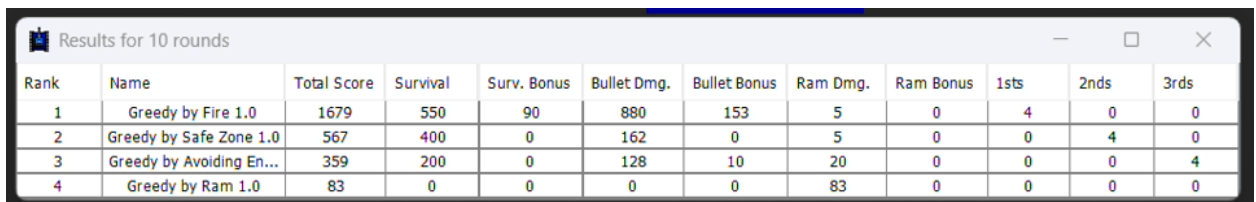
4. Pengujian Keempat



Rank	Name	Total Score	Survival	Surv. Bonus	Bullet Dmg.	Bullet Bonus	Ram Dmg.	Ram Bonus	1sts	2nds	3rds
1	Greedy by Fire 1.0	1527	450	60	816	134	66	0	4	1	0
2	Greedy by Avoiding Enemies 1.0	567	400	30	120	6	11	0	1	2	2
3	Greedy by Safe Zone 1.0	436	250	30	148	0	7	0	0	2	1
4	Greedy by Ram 1.0	244	100	0	0	0	144	0	0	0	3

Gambar 4.5.1.4 Pengujian keempat

5. Pengujian Kelima



Rank	Name	Total Score	Survival	Surv. Bonus	Bullet Dmg.	Bullet Bonus	Ram Dmg.	Ram Bonus	1sts	2nds	3rds
1	Greedy by Fire 1.0	1679	550	90	880	153	5	0	4	0	0
2	Greedy by Safe Zone 1.0	567	400	0	162	0	5	0	0	4	0
3	Greedy by Avoiding Enemies 1.0	359	200	0	128	10	20	0	0	0	4
4	Greedy by Ram 1.0	83	0	0	0	0	83	0	0	0	0

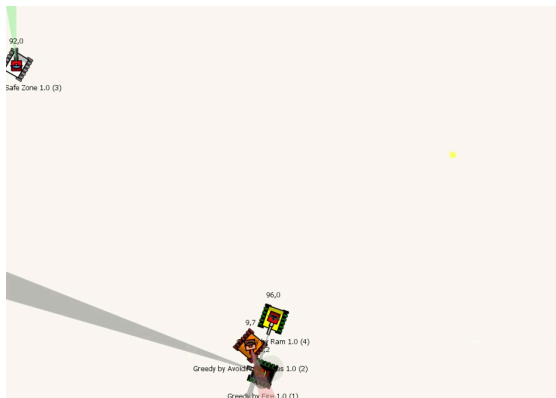
Gambar 4.5.1.5 Pengujian kelima

Dari 5 kali uji coba keempat bot yang kelompok kami sudah buat. Bot utama kami, yang dibuat berdasar greedy by fire, mendapat hasil paling optimal, yakni meraih skor paling tinggi sebanyak lima kali dari lima percobaan.

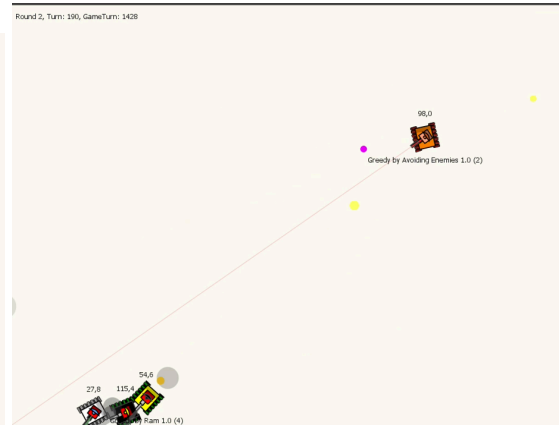
4.5.2 Kondisi Kondisi Setiap Algoritma Bot

a.) Bot Greedy by Safe Zone

Pada saat uji coba bot greedy by safe zone ini sempat mengalami kondisi optimal dan juga tidak optimalnya. Bot mendapat nilai optimal ketika berhasil mencapai safe zone yang berada jauh dari musuh lainnya, bot mendapat skor yang cukup tinggi ketika kondisi optimal ini terpenuhi. Bot ini juga mengalami kondisi tidak optimal, yakni gagal mencapai safe zone nya sehingga tidak mendapat skor optimal dan tereliminasi dengan cepat



Kondisi Optimal



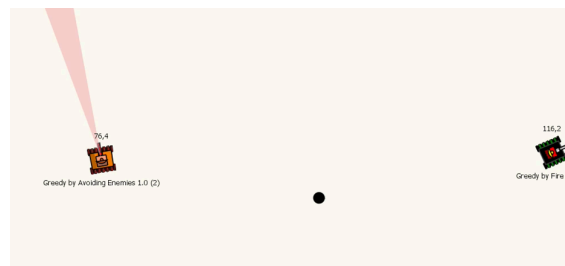
Kondisi Tidak Optimal

b.) Bot Greedy by Fire

Kondisi Optimal bot ini yaitu ketika musuh tersisa sedikit dan jarak musuh dekat, bot akan berhasil mengunci pergerakan musuh dan terus mengikutinya sambil terus menembak. Kondisi kurang optimal dari bot ini yaitu ketika bot lawan berada cukup jauh sehingga scan tidak dapat menjangkau dan bot akan mengalami kebingungan sehingga bergerak tanpa arah sampai berhasil mengunci pergerakan lawan.



Kondisi Optimal

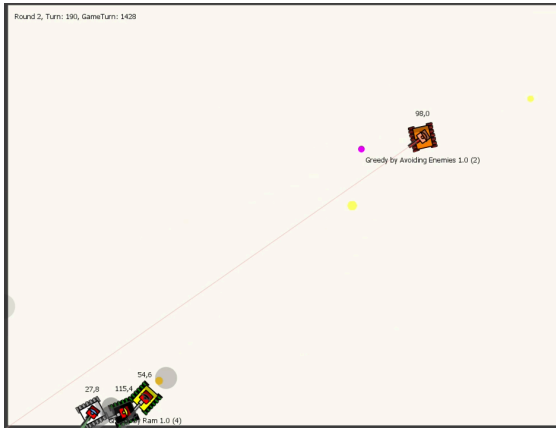


Kondisi Tidak Optimal

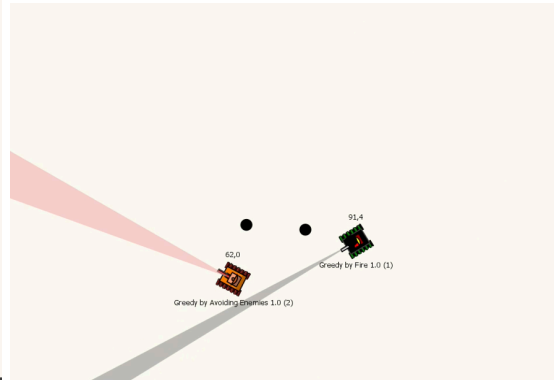
c.) Bot Greedy by Avoiding Enemy

Bot ini mengalami kondisi optimal ketika berhasil menghindari dari lawan dan berada cukup jauh dari lawan lainnya, atau ketika musuh sedang berkumpul dan bot berhasil menghindari kumpulan

tersebut. Bot akan menjadi kurang optimal ketika melawan musuh yang agresif dan memiliki algoritma untuk mengejar dan mengunci pergerakan yang kuat.



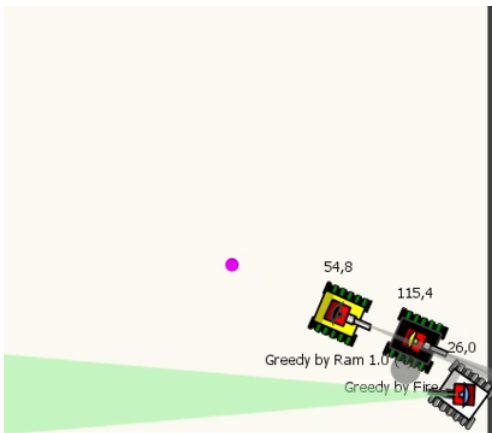
Kondisi Optimal



Kondisi Tidak Optimal

d.) Bot Greedy by Ram

Bot ini berada di kondisi optimal ketika berhasil mengunci lawan dan terus menabrak lawan tersebut sehingga bisa mendapat poin maksimal dan berada di kondisi yang tidak optimal ketika



Kondisi Optimal



Kondisi Tidak Optimal

BAB V

KESIMPULAN DAN SARAN

5.1 Kesimpulan

Melalui program ini, kami telah berhasil menerapkan algoritma greedy untuk membuat bot yang andal dalam memenangkan pertandingan. Algoritma greedy dengan berbagai heuristik menawarkan berbagai alternatif yang menuntut kami untuk bisa melihat sesuatu dari berbagai sudut pandang. Selain itu, proses pengerjaan tugas ini menuntut kami untuk menguji berbagai rancangan yang telah dibuat serta mempertanggungjawabkannya. Dari beragam rancangan dan pengujian yang telah dilakukan, kami mendapati bahwa algoritma yang sederhana namun cerdas lebih menguntungkan. Oleh karena itu, kami dapat menyimpulkan bahwa algoritma yang paling baik adalah *greedy by fire*.

5.2 Saran

Untuk pengembangan selanjutnya, saran yang dapat dilakukan pada bot utama adalah peningkatan akurasi dalam penembakan. Akurasi penembakan pada bot utama masih belum dapat memprediksi kemana musuh akan bergerak setelah di-*scan*, sehingga pengembangan mengenai hal tersebut dapat dilanjutkan di kesempatan yang lain.

LAMPIRAN

Tabel :

No	Poin	Ya	Tidak
1.	Bot dapat dijalankan pada Engine yang sudah dimodifikasi asisten.	✓	
2.	Membuat 4 solusi greedy dengan heuristic yang berbeda.	✓	
3.	Membuat laporan sesuai dengan spesifikasi.	✓	
4.	Membuat video bonus dan diunggah pada Youtube.	✓	

Link Github : https://github.com/Rusmn/Tubes1_bebas-aja

Link Youtube : <https://youtu.be/NjCDjsYdjYo>

DAFTAR PUSTAKA

Rinaldi, M. (n.d.). Algoritma Greedy (2021) Bag1. Diakses 24 Maret 2025 melalui [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)

Rinaldi, M. (n.d.). Algoritma Greedy (2021) Bag2. Diakses 24 Maret 2025 melalui [https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2024-2025/05-Algoritma-Greedy-\(2025\)-Bag2.pdf](https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2024-2025/05-Algoritma-Greedy-(2025)-Bag2.pdf)

Rinaldi, M. (n.d.). Algoritma Greedy (2021) Bag3. Diakses 24 Maret 2025 melalui [https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2024-2025/06-Algoritma-Greedy-\(2025\)-Bag3.pdf](https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2024-2025/06-Algoritma-Greedy-(2025)-Bag3.pdf)