IF2211 - Strategi Algoritma

Laporan Tugas Besar 1

Pemanfaatan Algoritma Greedy dalam Pembuatan Bot Permainan Robocode Tank Royale

Disusun Oleh:

Adinda Putri	13523071
Heleni Gratia M. Tampubolon	13523107
Naomi Risaka Sitorus	13523122

Program Studi Teknik Informatika

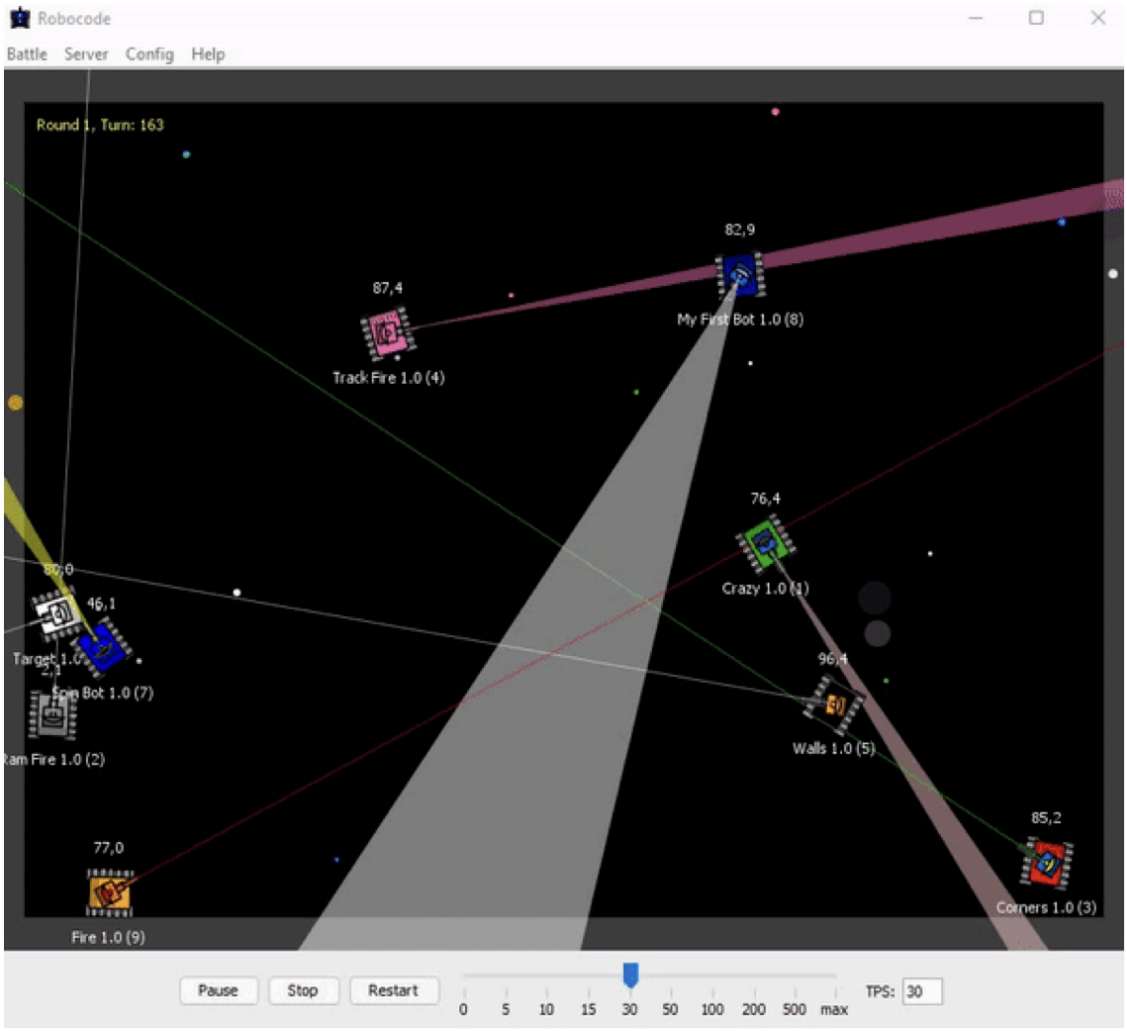
Sekolah Teknik Elektro dan Informatika

Institut Teknologi Bandung

2025

DAFTAR ISI

BAB 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 rounds and turns, batas waktu giliran, energi, peluru, panas meriam (gun heat), tabrakan, bagian tubuh tank, pergerakan, berbelok, pemindaian, dan skor.

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.

   Pada setiap turn, bot akan secara otomatis menerima informasi terbaru tentang posisinya dan orientasinya di medan perang. Bot juga akan mendapatkan informasi tentang bot musuh ketika mereka terdeteksi oleh pemindai.

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.

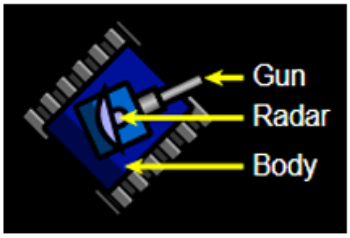
6. Tabrakan

   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:



Gambar 1. Tubuh Tank

- Body adalah bagian utama dari tank yang digunakan untuk menggerakkan tank.
- Gun digunakan untuk menembakkan peluru dan dapat berputar bersama body atau independen dari body.
- 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.

Gambar 2. Ilustrasi Radar Bot

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 diranking 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 lainya 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.

BAB 2
LANDASAN TEORI

2.1 Algoritma Greedy

Algoritma greedy adalah salah satu metode yang populer dan sederhana dalam memecahkan persoalan optimasi. Solusi dalam algoritma tersebut dicapai secara bertahap, langkah demi langkah. Pada setiap langkah, algoritma ini memilih opsi terbaik yang dapat diperoleh pada saat itu tanpa mempertimbangkan dampak atau konsekuensi di masa depan. Prinsip dasar dari algoritma ini mengutamakan keputusan optimal lokal di setiap langkah, dengan harapan bahwa pilihan-pilihan tersebut akan menghasilkan solusi optimal secara keseluruhan. Algoritma greedy digunakan untuk menyelesaikan dua jenis persoalan optimasi utama, yaitu maksimasi (*maximization*) dan minimasi (*minimization*), dengan tujuan mencapai solusi yang terbaik dalam konteks yang diberikan.

2.2. Elemen-Elemen dalam Algoritma Greedy

2.2.1. Himpunan Kandidat

Berisi kandidat yang dapat dipilih pada setiap langkah. Seperti simpul/sisi di dalam graf, job atau task dalam penjadwalan, koin dalam masalah pengembalian uang, atau benda dalam masalah knapsack.

2.2.2. Himpunan Solusi

Berisi kandidat yang telah dipilih dalam proses optimasi atau pemilihan keputusan. Kandidat dalam himpunan solusi dianggap memenuhi kriteria yang sesuai untuk mencapai solusi optimal.

2.2.3. Fungsi Solusi

Fungsi solusi adalah fungsi yang digunakan untuk menentukan apakah himpunan kandidat yang dipilih telah memberikan solusi.

2.2.4. Fungsi Seleksi

Fungsi seleksi adalah fungsi bersifat heuristik yang digunakan untuk memilih kandidat berdasarkan strategi greedy tertentu.

2.2.5. Fungsi Kelayakan

Fungsi kelayakan adalah fungsi yang digunakan untuk memeriksa apakah suatu kandidat yang dipilih dapat dimasukkan ke dalam himpunan solusi.

2.2.6. Fungsi Objektif

Fungsi objektif adalah fungsi yang berisi tujuan dari algoritma tersebut, serta memaksimumkan dan meminimumkan.

2.3. Robocode

Robocode merupakan permainan pemrograman berbasis simulasi yang bertujuan untuk membuat kode bot dalam bentuk tank virtual untuk berkompetisi melawan bot lain di arena. Nama Robocode adalah singkatan dari "Robot Code" yang berasal dari versi asli permainan ini dan hendak merujuk pada esnesi utama permainan yang berfokus pada pemrograman untuk mengontrol perilaku bot. Bot-bot tersebut akan diprogram menggunakan bahasa Java dan berusaha agar bot mampu untuk melakukan berbagai aksi, seperti bergerak, menembak, menghindari serangan, dan berinteraksi dengan robot lain. Pertempuran Robocode akan berlangsung hingga bot-bot bertarung hanya tersisa satu bot yang bertahan sebagai pemenang. Konsep ini menyerupai mode permainan *Battle Royale*.

Pada tugas besar kali ini, dilakukan penyesuaian (pengembangan) terhadap versi Robocode menjadi Robocode Tank Royale, yaitu bot dapat berpartisipasi melalui internet/jaringan. Pemain berperan sebagai programmer yang merancang kode 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 berlangsung dalam bahasa C#.

2.4. Cara Kerja Program

2.2.1. Cara Kerja Game Engine dan Bot pada Robocode Tank Royale

Robocode menggunakan framework berbasis Java (JVM dan dijalankan pada versi Java 11) yang dapat mensimulasikan pertempuran antar robot dalam lingkungan virtual. API robocode terdiri dari beberapa bagian utama:

1. Bot API (package `robocode.TankRoyale.BotApi`), digunakan untuk mengembangkan

robot dan memungkinkan bot untuk berinteraksi dengan game engine melalui websocket. Selain itu, bot juga dapat menerima event dari server game (deteksi musuh, tembakan, dan tabrakan) dan mendukung pengembangan bot dengan bahasa pemrograman lain melalui API berbasis jaringan.

2. Control API, digunakan untuk memungkinkan memulai dan mengontrol pertandingan antar bot dan mengambil data real-time dari pertempuran.
3. Game Server, merupakan server pusat yang menangani komunikasi antar bot dan game engine, serta mengelola aturan permainan, siklus pertempuran, dan pemrosesan event.

Proses eksekusi API Robocode Tank Royale berupa:

1. Inisialisasi Bot, bot akan dibuat dengan mengimplementasikan Bot class dari Bot API sehingga bot dapat merespons event dari server seperti deteksi lawan dan tembakan.
2. Server dijalankan sehingga bot dapat terhubung melalui WebSocket. Hal ini akan membuat setiap bot dapat menerima data real-time dari server dan mengirimkan perintah kembali.
3. Server akan menangani setiap *turn* permainan dan mengeksekusi perintah bot. Bot kemudian dapat mengontrol pergerakan, menembak, dan lain hal berdasarkan data dari server.

2.2.2. Prosedur Menjalankan Game Engine dam Bot

Agar dapat menjalankan game engine Robocode Tank Royale maupun bot, komputer harus memenuhi seluruh persyaratan (*requirements*) program. Komputer harus memiliki IDE yang sudah terinstal bahasa pemrograman Java dan C# serta terpasang .NET SDK versi 8.0 dan Robocode Tank Royale API. Pemasangan API tersebut dapat dilakukan dengan mengetik `dotnet add package Robocode.TankRoyale.BotApi` pada terminal IDE. Game engine dengan nama file *robocode-tankroyale-gui-0.30.0.jar* dapat diunduh dari Starter Pack Tubes 1 Stima, sedangkan bot dapat diunduh dari *repository* kelompok EREMIKA dengan melakukan

`git clone https://github.com/adndax/Tubes1_EREMIKA.git` pada terminal IDE.

Untuk menjalankan game engine, buka folder lokasi game engine di terminal IDE dan jalankan `java -jar robocode-tankroyale-gui-0.30.0.jar`. Untuk melakukan setup konfigurasi booter di game engine, klik tombol Config lalu Bot Root Directories dan tambahkan directory yang berisi folder bot, yakni …/Tubes1_EREMIKA/src/main-bot dan/atau …/Tubes1_EREMIKA/src/alternative-bots. Lakukan setup server permainan dengan menekan tombol Server lalu Start Local Server. Kemudian, buka terminal baru untuk menjalankan bot yang membuka folder lokasi hasil *clone repository* dan melakukan `cd src`. Terapkan bot secret pada terminal agar bot bisa bergabung ke server dengan perintah `set SERVER_SECRET=<bots_secret>`, bots_secret dapat diperoleh dari file server.properties yang ada di directory yang sama dengan file game engine. Untuk melakukan compile pada bot, jalankan perintah `dotnet build` pada setiap folder bot yang ingin dijalankan. Hal tersebut harus dilakukan setiap kali dilakukan perubahan pada *source code* bot.

Mulai permainan Robocode Tank Royale dengan menekan tombol Battle lalu Start Battle pada game engine. Pilih bot yang ingin dimainkan pada kotak Bot Directories (local only) dan tekan tombol Boot. Bot akan masuk ke kotak Booted Bots lalu ke kotak Joined Bots (local/remote). Tambahkan bot ke dalam permainan dengan memilih bot yang ada di kotak Joined Bots dan menekan tombol Add. Permainan akan dimulai setelah pemain menekan tombol Start Battle.

BAB 3
APLIKASI STRATEGI GREEDY

3.1. Mapping Persoalan

3.1.1. Himpunan Kandidat

Berupa strategi dan algoritma yang dapat digunakan untuk mengendalikan robot dalam pertempuran. Seperti strategi menembak, strategi gerakan, dan strategi menghindar.

3.1.2. Himpunan Solusi

Berupa subset dari himpunan kandidat yang memenuhi syarat sebagai strategi yang dapat diimplementasikan dan bersaing dalam Robocode Tank Royale.

3.1.3. Fungsi Solusi

Fungsi yang menentukan apakah strategi yang dipilih menghasilkan algoritma bot yang efektif.

3.1.4. Fungsi Seleksi

Fungsi yang memilih strategi (kandidat) yang lebih baik untuk dipilih sebagai solusi optimal berdasarkan jumlah kemenangan dalam pertempuran, persentase akurasi tembakan, dan performa robot dalam turnamen.

3.1.5. Fungsi Kelayakan

Fungsi yang menentukan apakah suatu strategi (kandidat) layak digunakan dalam pertempuran berdasarkan efektivitas strategi dan efisiensi penggunaan sumber daya.

3.1.6. Fungsi Objektif

Fungsi yang menentukan apakah suatu hal harus dimaksimumkan atau diminimumkan.

3.2. Alternatif Solusi Greedy

3.2.1. Bot Alternatif 1: Levi

3.2.3.1 Greedy by Shot Accuration

Greedy by shot accuration adalah pendekatan algoritma greedy secara ofensif dengan mempertimbangkan akurasi tembakan dan jarak bot terhadap musuh. Ketika bot berada pada jarak yang cukup jauh atau lebih dari 100 unit dari

musuh yang terdeteksi pada radar, bot akan menembak sambil berputar. Sebaliknya, ketika jarak musuh kurang dari 100 unit, bot akan menabrak dan menembak musuh dengan kecepatan peluru maksimal.

a. Mapping Elemen Greedy

1) Himpunan kandidat

Himpunan kandidat dalam algoritma ini adalah seluruh aksi yang dapat dijalankan oleh bot. Kandidat tersebut berfokus pada akurasi yang didasarkan pada kondisi jarak bot terhadap musuh.

2) Himpunan solusi

Himpunan solusi berisi aksi yang dipilih berdasarkan hubungan akurasi, daya tembakan, dan kondisi jarak bot terhadap musuh, yaitu:

a. Jika jarak musuh jauh atau lebih dari 100 unit, solusi yang dipilih adalah menembak secara berputar untuk menghindari tembakan musuh dan memperbesar kemungkinan tembakan bot mengenai musuh.
b. Jika bot berada dalam jarak dekat, yaitu kurang dari 100 unit, solusi yang dipilih adalah menabrak musuh dan menembak dengan kecepatan peluru maksimal karena akurasi tinggi.

3) Fungsi solusi

Fungsi solusi dalam algoritma ini menentukan apakah strategi greedy yang dipilih optimal berdasarkan akurasi tembakan, yaitu:

a. Jika bot berhasil menembak secara berputar untuk menghindari tembakan musuh.
b. Jika bot berhasil menembak dengan daya besar dari dekat untuk memaksimalkan damage peluru.

4) Fungsi seleksi

Fungsi seleksi digunakan untuk memilih solusi terbaik dari himpunan solusi yang ada berdasarkan kondisi tertentu. Pada algoritma ini, kondisi yang dimaksud terkait dengan hubungan

jarak dan akurasi tembakan. Fungsi seleksi mengevaluasi kondisi jarak bot terhadap musuh dan memilih aksi yang paling optimal dengan mempertimbangkan akurasi tembakan.

5) Fungsi kelayakan

   Fungsi kelayakan mengevaluasi kelayakan dari strategi greedy tembakan yang dilakukan. Strategi bot yang ideal ditentukan dengan beberapa pertimbangan:

   a. Jika jarak jauh atau lebih dari 100 unit, bot dapat menembak secara berputar untuk menghindari bot musuh.
   b. Jika jarak dekat atau kurang dari 100 unit, bot dapat menabrak dan menembak musuh dengan kecepatan peluru maksimal untuk memaksimalkan damage yang diterima.

6) Fungsi objektif

   Fungsi objektif atau tujuan dari strategi tembakan dalam algoritma ini berfokus pada dua hal utama, yaitu:

   a. Mengutamakan peluang untuk menghindari peluru dari musuh dengan pergerakan berputar sambil menembak pada jarak jauh.
   b. Memaksimalkan damage dengan menabrak dan menembak musuh dengan kecepatan dan daya peluru maksimal sehingga menghasilkan damage yang maksimal.

b. Analisis Efisiensi Solusi

Greedy by shot accuration diterapkan dalam fungsi `OnScannedBot()` untuk memprediksi posisi bot musuh dan menentukan tindakan yang harus diambil berdasarkan posisi tersebut. Fungsi ini hanya menjalankan serangkaian operasi sederhana, seperti memeriksa posisi musuh dengan menggunakan percabangan yang memiliki kompleksitas O(1), serta eksekusi perintah tembakan ke bot lawan yang juga dilakukan dalam waktu konstan O(1). Proses ini akan dieksekusi berulang kali setiap kali bot mendeteksi musuh, sehingga

kompleksitas waktu keseluruhan dari strategi ini menjadi O(n), di mana n adalah jumlah iterasi bot mendeteksi musuh.

c. Analisis Efektivitas Solusi

Greedy by shot accuration adalah pendekatan algoritma greedy yang berfokus pada akurasi tembakan sebagai dasar pemilihan tindakan. Dalam algoritma ini, bot membuat keputusan berdasarkan jarak dengan musuh dan mengutamakan akurasi tembakan yang dihasilkan. Algoritma ini efektif digunakan apabila:

1. Musuh memiliki pola pergerakan yang relatif statis atau lambat sehingga bot dapat dengan mudah memprediksi posisi musuh dan menyesuaikan akurasi tembakan untuk memastikan tembakan mengenai target.
2. Bot memiliki energi yang cukup untuk menggunakan daya tembakan tinggi pada jarak dekat.

Algoritma ini kurang efektif digunakan apabila:

1. Pergerakan musuh terlalu aktif dan tidak terduga sehingga menyebabkan tembakan meleset dan kurang akurat.
2. Dua musuh berada pada jarak yang dekat. Bot tidak dapat menyerang keduanya dalam waktu bersamaan, bot harus memilih satu target prioritas untuk ditembak.

3.2.4.2 Greedy by Enemy's Energy

Greedy by enemy's energy adalah pendekatan algoritma greedy yang berfokus pada pemilihan aksi terbaik berdasarkan energi musuh. Dalam algoritma ini, tindakan yang diambil oleh bot dipilih berdasarkan jumlah energi yang dimiliki musuh. Algoritma ini akan memprioritaskan tindakan yang mengoptimalkan kerusakan yang dihasilkan dengan cara menyesuaikan kekuatan tembakan bot, seperti menyerang dengan kekuatan penuh jika musuh memiliki energi tinggi, atau menggunakan daya tembakan yang lebih rendah untuk menghemat energi jika musuh memiliki energi rendah karena kemungkinan musuh akan segera hancur.

a. Mapping Elemen Greedy

1) Himpunan kandidat

   Himpunan kandidat dalam algoritma Greedy by enemy's energy terdiri dari semua langkah atau aksi yang dapat dipilih oleh bot berdasarkan posisi dan energi musuh. Tindakan ini mencakup kekuatan tembakan peluru yang diberikan pada musuh.

2) Himpunan solusi

   Himpunan solusi adalah aksi-aksi yang dipilih oleh bot pada setiap kondisi berdasarkan himpunan kandidat yang ada. Solusi yang dipilih merupakan aksi terbaik yang didasarkan energi musuh, yaitu:

   a. Jika energi musuh lebih dari 16, solusi yang dipilih adalah menembak dengan daya penuh (Fire(3)) untuk memberikan kerusakan maksimal.
   b. Jika energi musuh antara 10 hingga 16, solusi yang dipilih adalah menembak dengan daya sedang (Fire(2)) untuk memaksimalkan peluang mengalahkan musuh dan menghemat energi.
   c. Jika energi musuh antara 4 hingga 10, solusi yang dipilih adalah menembak dengan daya rendah (Fire(1)) karena kemungkinan musuh akan segera hancur.
   d. Jika energi musuh kurang dari 4, solusi yang dipilih adalah menembak dengan daya sangat rendah untuk menghindari pemborosan daya tembakan.

3) Fungsi solusi

   Fungsi solusi adalah bagian dari algoritma yang menilai apakah aksi yang dipilih dalam himpunan solusi optimal atau tidak. Fungsi ini mengevaluasi apakah aksi yang dipilih dalam himpunan solusi optimal berdasarkan energi musuh dan kerusakan yang dihasilkan, yaitu:

a. Jika energi musuh tinggi, tembakan daya tinggi dapat menghasilkan kerusakan maksimal dengan risiko lebih rendah.
b. Jika energi musuh rendah, tembakan daya rendah dapat menghasilkan kerusakan dengan sedikit daya yang dikeluarkan.

4) Fungsi seleksi

Fungsi seleksi bertugas untuk memilih solusi terbaik dari himpunan solusi yang ada berdasarkan energi musuh. Fungsi ini memilih strategi yang optimal untuk menyerang atau bertahan, seperti:

a. Jika energi musuh tinggi, bot memilih untuk menyerang dengan daya penuh untuk mengalahkan musuh yang masih memiliki banyak energi.
b. Jika energi musuh rendah, bot memilih untuk menembak dengan daya rendah untuk mengalahkan musuh tanpa membuang daya tembakan.

5) Fungsi kelayakan

Fungsi kelayakan mengevaluasi apakah solusi yang dipilih layak diterapkan berdasarkan energi musuh dan potensi serangan yang dapat dilakukan. Kelayakan yang dimaksud adalah bot dapat mengalahkan musuh tanpa kehilangan energi yang berarti.

6) Fungsi objektif

Fungsi objektif berfokus pada tujuan utama dari strategi yang diterapkan, yaitu untuk memaksimalkan kerusakan pada musuh dan menghemat daya tembakan ketika energi musuh sudah rendah.

b. Analisis Efisiensi Solusi

Greedy by enemy's energy diterapkan pada fungsi `OnHitBot()` untuk menabrak musuh dan menembak berdasarkan sisa energi musuh. Fungsi ini hanya melibatkan serangkaian operasi sederhana, seperti memanggil fungsi `TurnToFaceTarget()` untuk menghadap musuh

dan menembak bot lawan, yang keduanya dilakukan dalam waktu konstan O(1). Proses ini akan terus dieksekusi setiap kali bot menabrak musuh, sehingga kompleksitas waktu keseluruhan dari strategi ini adalah O(n), dengan n sebagai jumlah kali bot menabrak musuh.

c. Analisis Efektivitas Solusi

Greedy by enemy's energy adalah pendekatan algoritma greedy yang berfokus pada pemilihan tindakan berdasarkan energi dan posisi musuh. Algoritma ini efektif digunakan apabila:

1. Musuh memiliki pola pergerakan yang relatif statis atau lambat sehingga bot dapat lebih mudah memilih tembakan yang sesuai dengan energi musuh yang terdeteksi.
2. Bot hanya berhadapan dengan satu musuh utama, karena keputusan berdasarkan energi musuh akan lebih mudah dievaluasi.

Algoritma ini kurang efektif digunakan apabila:

1. Musuh memiliki kemampuan untuk menghindari serangan dengan pola gerakan evasive sehingga tembakan bot meleset.
2. Musuh memiliki energi yang berubah-ubah sehingga bot kesulitan untuk memilih aksi yang tepat.

3.2.2. Bot Alternatif 2: Eren

3.2.2.1 Greedy by Shot Velocity

Greedy by shot velocity adalah pendekatan algoritma greedy dengan memprioritaskan kecepatan tembakan terhadap musuh (bot lain). Konsep greedy by shot velocity dalam kode ini ialah menyesuaikan kekuatan tembakan berdasarkan jarak dari musuh yang dipindai. Kekuatan tembakan tersebut akan secara otomatis menyesuaikan kecepatan tembakan yang terjadi karena peluru yang memiliki daya tembak lebih kuat akan lebih berat sehingga menyebabkan pergerakan yang lambat. Sedangkan peluru yang daya tembak lebih kecil akan lebih ringan dan memiliki pergerakan yang cepat.

a. Mapping Elemen Greedy

1) Himpunan kandidat

Himpunan kandidat dalam algoritma ini adalah seluruh perhitungan algoritma kekuatan tembakan yang dapat digunakan oleh bot. Kandidat ini didasarkan pada berbagai skenario jarak antara bot dengan musuh yang mempengaruhi kekuatan dan kecepatan peluru yang ditembakkan.

2) Himpunan solusi

Himpunan solusi dalam algoritma ini adalah seluruh perhitungan yang terpilih.berdasarkan fungsi solusi yang menentukan kekuatan tembakan optimal untuk setiap jarak. Solusi yang dipilih ialah formula:

```
firePower=max(1,min(3,400/jarak_musuh))
```

3) Fungsi Solusi

Fungsi solusi dalam pendekatan algoritma ini menentukan hubungan antara jarak musuh dengan kekuatan tembakan yang digunakan oleh bot. Hal ini berdasarkan bahwa:

a. Jika musuh jauh, maka bot akan memilih tembakan dengan daya lebih kecil agar peluru lebih cepat mencapai target.
b. Jika musuh dekat, maka bot akan memilih tembakan dengan daya yang lebih besar, meskipun kecepatannya lebih lambat untuk memberikan damage yang lebih tinggi.

4) Fungsi Seleksi

Fungsi seleksi dalam algoritma ini memilih strategi tembakan yang paling optimal berdasarkan kondisi pertempuran. Kriteria pemilihan strategi berdasarkan:

a. Efektivitas tembakan, memilih kekuatan tembakan yang memungkinkan mengenai target dengan akurasi tinggi.
b. Jumlah kemenangan dalam pertempuran akibat algoritma kekuatan tembakan yang dipilh.

5) Fungsi Kelayakan

Fungsi kelayakan memastikan bahwa strategi tembakan yang dipilih efektif (tidak menyebabkan bot kehabisan energi dengan cepat dan dapat menembak akurat) dan sesuai aturan permainan Robocode Tank Royale. Beberapa pertimbangan yang dilakukan ialah:

a. Bot tidak boleh menembakkan peluru dengan daya terlalu besar jika energinya sangat rendah.
b. Bot dapat menembak dengan tepat sasaran beberapa kali saat dilakukan pengujian.

6) Fungsi Objektif

Fungsi objektif dalam algoritma ini adalah memaksimalkan damage ke musuh sambil meminimalkan energi yang terbuang, yaitu

a. Menembak dengan daya maksimum ketika peluang mengenai target tinggi.
b. Menggunakan daya rendah ketika musuh berada jauh untuk mempercepat waktu tempuh peluru.

b. Analisis Efisiensi Solusi

Dalam pengaplikasian Greedy by shot velocity, fungsi `OnScannedBot()` bekerja dengan mengevaluasi posisi bot terhadap bot lain yang terdeteksi dan menyesuaikan kekuatan tembakan berdasarkan algoritma greedy yang diterapkan. Fungsi ini hanya melakukan serangkaian operasi sederhana, seperti pengecekan kondisi posisi bot dengan kompleksitas O(1) dan eksekusi perintah pergerakan bot yang juga berjalan dalam waktu konstan O(1). Namun, karena fungsi ini dapat dieksekusi berulang kali selama permainan berlangsung, jumlah eksekusinya bergantung pada seberapa sering bot mendeteksi lawan dan menyesuaikan strategi tembakannya berdasarkan jarak, yang dinyatakan sebagai n. Oleh karena itu, kompleksitas keseluruhan strategi ini menjadi O(n), dengan n adalah banyaknya kali fungsi tersebut dieksekusi sepanjang permainan.

c. Analisis Efektivitas Solusi

Greedy by shot velocity didasarkan pada penyesuaian kekuatan tembakan berdasarkan jarak musuh dengan tujuan untuk meningkatkan peluang mengenai target secara efisien. Namun, strategi ini bukanlah solusi yang paling optimal karena terdapat faktor-faktor yang dapat mempengaruhi efektivitasnya, seperti pola pergerakan lawan dan tingkat akurasi tembakan.

Strategi ini efektif jika:

1. Musuh memiliki pola gerakan yang relatif konstan sehingga lebih mudah untuk ditembak secara presisi.
2. Bot memiliki cukup energi, sehingga dapat memilih data tembakan yang lebih tinggi tanpa khawatir kehabisan energi.
3. Musuh berada dalam jarak yang sesuai dengan kekuatan tembakan optimal, sehingga tidak terlalu jauh (agar peluru tidak mudah dihindari) dan tidak terlalu dekat (agar bot tetap memiliki waktu untuk menembak).

Strategi ini tidak efektif jika:

1. Musuh bergerak secara acak atau sangat cepat sehingga meskipun peluru yang berdaya kecil lebih cepat, peluang mengenai target tetap rendah.
2. Bot memiliki energi yang rendah sehingga hanya dapat menembak dengan daya kecil, yang mungkin tidak cukup untuk mengalahkan lawan.

3.2.2.2 Greedy by Avoiding Wall Collision

Greedy by avoiding wall collision adalah pendekatan algoritma greedy untuk memastikan bot tidak menabrak dinding dengan mengubah arah geraknya ketika jaraknya terlalu dekat dengan batas arena. Strategi ini didasarkan pada prinsip greedy, yaitu mengambil keputusan terbaik secara lokal untuk menghindari tabrakan setiap saat, tanpa memperhitungkan langkah-langkah di masa depan.

a. Mapping Elemen Greedy

1) Himpunan Kandidat

Himpunan kandidat dalam algoritma ini adalah seluruh kemungkinan arah pergerakan yang dapat diambil oleh bot untuk menghindari tabrakan dengan dinding.

2) Himpunan Solusi

Himpunan solusi adalah pergerakan yang dipilih paling efektif berdasarkan kondisi posisi bot terhadap dinding, yaitu saat ingin melakukan gerakan selanjutnya (berpindah posisi) maka dilakukan pengecekan posisi bot. Jika berada pada jarak minimum dari dinding, yaitu yang dipilih adalah 10 unit jarak, maka bot akan mundur sejauh 50 unit jarak (piksel) dari posisinya kemudian

mengubah sudut sebesar 45 derajat dengan tujuan agar dapat maju dan menjauhi dinding.

3) Fungsi Solusi

Fungsi solusi menentukan bagaimana bot memilih langkah terbaik saat berada dalam kondisi dekat terhadap dinding, yaitu bot dapat mengubah arahnya agar menjauhi dinding.

4) Fungsi Seleksi

Fungsi seleksi memilih pergerakan terbaik dari himpunan kandidat berdasarkan kriteria:

a. Jika posisi bot berada di dekat dinding, pilih langkah yang menjauhkannya dari dinding.
b. Perubahan arah diambil dengan sudut tetap untuk memastikan bot tidak langsung kembali ke arah dinding.
c. Mempertimbangkan kondisi ketika bot berada di dekat dinding dan mengalami tabrakan dengan bot lain atau terkena tembakan, setiap peristiwa tersebut memiliki cara reaksi yang berbeda. Oleh karena itu, perlu dipastikan bahwa aksi yang diambil tidak saling mengganggu atau menyebabkan respons yang tidak diinginkan.

5) Fungsi Kelayakan

Fungsi kelayakan menentukan apakah solusi yang dipilih valid atau tidak. Solusi dianggap layak jika:

a. Pergerakan yang dipilih dapat membawa bot menjauhi dinding.
b. Bot masih memiliki cukup ruang untuk bergerak setelah perubahan arah yang dilakukan.

6) Fungsi Objektif

Fungsi objektif dalam pendekatan ini adalah meminimalkan kemungkinan bot menabrak dinding. Dengan demikian, strategi ini bertujuan untuk memastikan bot tetap berada di area aman sehingga

dapat terus bergerak secara optimal dalam pertempuran dan tidak kehilangan energi saat bertabrakan dengan dinding.

b. Analisis Efisiensi Solusi

Dalam pengaplikasian Greedy by avoiding wall collision, fungsi `CheckWallSmoothing()` bekerja dengan mengevaluasi posisi bot terhadap batas arena dan melakukan penyesuaian arah untuk menghindari tabrakan dengan dinding. Fungsi ini hanya melakukan serangkaian operasi sederhana, seperti pengecekan kondisi posisi bot dengan kompleksitas O(1) dan eksekusi perintah pergerakan bot yang juga berjalan dalam waktu konstan O(1). Namun, karena fungsi ini dapat dieksekusi berulang kali selama permainan berlangsung, jumlah eksekusinya bergantung pada seberapa sering bot berada dalam kondisi yang mengharuskannya menghindari dinding (bergantung jarak), yang dinyatakan sebagai n. Oleh karena itu, kompleksitas keseluruhan strategi ini menjadi O(n), dengan n adalah banyaknya kali fungsi tersebut dieksekusi sepanjang permainan.

c. Analisis Efektivitas Solusi

Greedy by avoiding wall collision didasarkan pada tujuan untuk menjaga pergerakan bot tetap optimal dan tidak kehilangan energi dengan menghindari tabrakan dengan dinding. Dengan menerapkan algoritma ini, bot dapat bergerak secara lebih bebas di arena tanpa kehilangan kecepatan dan energi akibat menabrak dinding, yang dapat mengganggu strategi tempur.

Strategi ini efektif jika:

1. Tidak ada bot lawan yang menghalangi jalur (menembak dan menabrak) setelah bot mengubah arah.
2. Perubahan arah yang dilakukan tidak menyebabkan bot terjebak atau kembali menabrak dinding.

Strategi ini tidak efektif jika:

1. Bot lawan berada dalam posisi yang menghalangi pergerakan sehingga bot tidak bisa keluar dari posisi sempit.

2. Perubahan arah yang dilakukan akibat menghindari dinding justru menempatkan bot ke posisi yang lebih rentan terhadap serangan musuh.
3. Bot terlalu fokus menghindari dinding sehingga kehilangan efisiensi dalam menembak target.

3.2.3. Bot Alternatif 3: Mikasa

3.2.3.1 Greedy by Collision Response

Greedy by collision response merupakan pendekatan algoritma greedy yang memanfaatkan tabrakan dengan bot lawan untuk memaksimalkan serangan. Bot merespons terhadap tabrakan dengan melakukan pergerakan lalu segera menembak demi memaksimalkan kerusakan pada bot lawan.

a. Mapping Elemen Greedy

1) Himpunan kandidat

   Himpunan kandidat dalam algoritma ini adalah seluruh kemungkinan tindakan yang dapat dilakukan setelah terjadi tabrakan, yaitu pergerakan serta tembakan.

2) Himpunan solusi

   Himpunan solusi merupakan aksi bot yang menghitung jarak musuh setelah terjadi tabrakan. Jika jarak musuh dekat, dalam artian kurang dari 150 unit, bot akan mundur dan menembak dengan kekuatan maksimal, yaitu 3. Jika jarak musuh lebih dari itu, bot akan mundur dan menembak dengan kekuatan 2.

3) Fungsi solusi

   Fungsi solusi yang digunakan untuk memilih keputusan terbaik dalam algoritma ini ialah memilih untuk menembak bot setelah tabrakan serta menghindar dengan mundur dengan jarak random.

4) Fungsi seleksi

   Fungsi seleksi memilih tindakan yang harus dilakukan bot dengan prinsip sebagai berikut.

   a. Jika bot lawan posisinya dekat, mundur dan lakukan tembakan dengan kekuatan maksimum.

b. Jika bot lawan posisinya tidak dekat, mundur dan lakukan tembakan dengan kekuatan sedang.
c. Jika bot sendiri memiliki energi rendah, mundur dan lakukan tembakan dengan kekuatan lemah.

5) Fungsi kelayakan

Fungsi kelayakan pada algoritma ini ialah harus terjadi tabrakan dengan bot lawan sebelum aksi dilakukan serta bot harus memiliki energi yang cukup untuk menembak setelah tabrakan.

6) Fungsi objektif

Fungsi objektif pada bot ini bertujuan untuk memaksimalkan kerusakan bot lawan setelah terjadi tabrakan dengan melakukan tembakan dengan kekuatan optimal secepatnya.

b. Analisis Efisiensi Solusi

Greedy by collision response diaplikasikan dalam fungsi `OnHitBot()` untuk mengecek terjadinya tabrakan dengan bot lawan. Fungsi ini hanya melakukan serangkaian operasi sederhana, seperti kondisi bot sendiri dengan kompleksitas O(1) dan eksekusi perintah menembak bot lawan yang juga berjalan dalam waktu konstan O(1). Setiap tabrakan menyebabkan eksekusi fungsi ini sebanyak satu kali sehingga kompleksitas keseluruhan dari strategi ini adalah O(n) dengan n berupa jumlah tabrakan yang terjadi.

c. Analisis Efektivitas Solusi

Solusi ini efektif ketika:

1) Bot musuh bergerak lambat atau dalam posisi "terperangkap" di dekat dinding sehingga posisinya setelah tabrakan tidak jauh dari lokasi tabrakan.
2) Bot memiliki cukup energi setelah tabrakan sehingga dapat menembak dengan kekuatan yang optimal berdasarkan jarak lawan.

Solusi ini kurang efektif ketika:

1) Bot lawan bergerak secara cepat setelah terjadinya tabrakan.

2) Bot lawan memiliki strategi menghindar dari tabrakan yang baik sehingga strategi greedy ini sulit diterapkan selama permainan.
3) Energi bot rendah sehingga tidak dapat memanfaatkan tabrakan untuk menembak dengan kekuatan optimal.

3.2.3.2 Greedy by Enemy Detection

Greedy by enemy detection adalah pendekatan algoritma greedy yang memprioritaskan untuk menembak ketika mendeteksi bot musuh pada radarnya. Bot akan memilih kekuatan terbesar untuk menembak dengan harapan dapat memaksimalkan kerusakan pada bot musuh jika tembakan tersebut mengenainya.

a. Mapping Elemen Greedy

1) Himpunan kandidat

   Himpunan kandidat dalam algoritma ini adalah seluruh kemungkinan kekuatan tembakan yang dapat dilakukan oleh bot, yaitu dari 1 sampai 3.

2) Himpunan solusi

   Himpunan solusi merupakan aksi yang dipilih oleh bot, yaitu untuk selalu menggunakan kekuatan tembakan maksimal, yaitu 3, ketika energi cukup untuk menembak.

3) Fungsi solusi

   Fungsi solusi digunakan untuk memilih keputusan terbaik yang dalam algoritma ini memprioritaskan kekuatan terbesar agar kerusakan pada bot lawan maksimal.

4) Fungsi seleksi

   Fungsi seleksi pada algoritma ini selalu memilih untuk menggunakan kekuatan tembakan terbesar, yakni 3, kecuali ketika energi tidak mencukupi.

5) Fungsi kelayakan

   Fungsi kelayakan pada algoritma ini memastikan bot hanya melakukan tembakan dengan kekuatan maksimal ketika mendeteksi musuh pada radarnya.

6) Fungsi objektif

Fungsi objektif pada bot ini bertujuan untuk memaksimalkan kekuatan tembakan demi meningkatkan peluang merusak musuh sebanyak mungkin dalam waktu yang singkat.

b. Analisis Efisiensi Solusi

Greedy by enemy detection diaplikasikan dalam fungsi `OnScannedBot()` untuk mendeteksi posisi bot lawan setiap kali sehingga jika ada lawan pada radar, tembakan maksimum dapat dilakukan. Fungsi ini hanya melakukan serangkaian operasi sederhana, seperti pengecekan kondisi posisi bot dengan kompleksitas O(1) dan eksekusi perintah menembak bot lawan dengan kekuatan terbesar yang juga berjalan dalam waktu konstan O(1). Akan tetapi, eksekusi berulang kali pada fungsi ini berdasarkan n kali bot mendeteksi lawan menyebabkan kompleksitas keseluruhan strategi ini menjadi O(n).

c. Analisis Efektivitas Solusi

Solusi ini efektif ketika:

1. Bot musuh memiliki pola pergerakan yang repetitif atau statis sehingga memungkinkan akurasi tembakan berdaya tinggi tersebut menjadi lebih tinggi.
2. Bot memiliki cukup energi sehingga dapat menembak dengan kekuatan maksimal yang cukup menguras energi.

Solusi ini kurang efektif ketika:

1. Bot musuh bergerak secara cepat dan acak sehingga tembakan berkekuatan besar yang lebih lambat sulit mengenai target.
2. Energi bot rendah sehingga menghambat kemampuan bot untuk menembak dengan kekuatan maksimum.

3.2.4. Bot Alternatif 4: Armin

3.2.4.1 Greedy by Shot Amount

Greedy by shot amount adalah pendekatan algoritma greedy dengan prinsip menembak lawan dalam jumlah sebanyak mungkin tanpa mempertimbangkan keakuratan dari tembakan tersebut. Bot akan menembak setiap kali meriam siap sehingga jumlah tembakan maksimum dengan harapan

dapat mengenai lawan sebanyak mungkin meskipun tidak melakukan prediksi posisi atau pola pergerakan lawan.

a. Mapping Elemen Greedy

1) Himpunan kandidat

   Himpunan kandidat dalam algoritma ini adalah seluruh kemungkinan tembakan yang dapat dilakukan setelah bot melalui *cooldown*.

2) Himpunan solusi

   Himpunan solusi merupakan aksi bot yang selalu menembak setelah *cooldown* selesai.

3) Fungsi solusi

   Fungsi solusi yang digunakan untuk memilih keputusan terbaik dalam algoritma ini ialah memilih untuk menembak setiap kali bot dideteksi siap untuk menembak.

4) Fungsi seleksi

   Fungsi seleksi memilih tindakan yang harus dilakukan bot dengan prinsip sebagai berikut.

   a. Jika bot siap menembak, lakukan tembakan tanpa pertimbangan jarak.
   b. Jika bot masih dalam proses *cooldown*, tembakan tidak dilakukan dan cek kembali kondisi kesiapan bot untuk menembak.

5) Fungsi kelayakan

   Fungsi kelayakan pada algoritma ini berupa bot hanya berhenti menembak ketika energi tidak cukup untuk menembak. Ketika dalam posisi *cooldown*, bot menunggu untuk meluncurkan tembakannya.

6) Fungsi objektif

   Fungsi objektif pada bot ini bertujuan untuk memaksimalkan jumlah tembakan dengan kekuatan penuh (kombinasi dengan

greedy by shot power) demi meningkatkan peluang merusak musuh sebanyak mungkin dalam waktu yang singkat.

b. Analisis Efisiensi Solusi

Greedy by shot amount diaplikasikan dalam fungsi `OnScannedBot()` untuk mendeteksi kembali posisi bot lawan setiap kali selesai melakukan tembakan sehingga jumlah tembakan maksimum. Fungsi ini hanya melakukan serangkaian operasi sederhana, seperti pengecekan kondisi posisi bot dengan kompleksitas O(1) dan eksekusi perintah menembak bot lawan yang juga berjalan dalam waktu konstan O(1). Akan tetapi, eksekusi berulang kali pada fungsi ini berdasarkan n kali bot mendeteksi lawan menyebabkan kompleksitas keseluruhan strategi ini menjadi O(n).

c. Analisis Efektivitas Solusi

Solusi ini efektif ketika:

1. Bot musuh memiliki pola pergerakan yang repetitif atau statis sehingga memungkinkan akurasi tembakan menjadi lebih tinggi.
3) Bot memiliki cukup energi sehingga dapat menembak terus-menerus tanpa kehabisan daya.

Solusi ini kurang efektif ketika:

1. Bot musuh bergerak secara cepat dan acak sehingga jumlah tembakan yang banyak tidak menjamin akurasi.
2. Energi bot rendah sehingga harus menunggu selama jangka waktu tertentu hingga energi cukup untuk menembak.

3.2.4.2 Greedy by Shot Power

Greedy by shot power adalah pendekatan algoritma greedy yang memprioritaskan kekuatan tembakan tanpa mempertimbangkan keakuratan tembakan. Bot akan selalu memilih kekuatan terbesar tembakan dengan harapan dapat memaksimalkan kerusakan pada bot musuh jika mengenai bot lawan.

a. Mapping Elemen Greedy

1) Himpunan kandidat

Himpunan kandidat dalam algoritma ini adalah seluruh kemungkinan kekuatan tembakan yang dapat dilakukan oleh bot, yaitu dari 1 sampai 3.

2) Himpunan solusi

Himpunan solusi merupakan aksi yang dipilih oleh bot, yaitu untuk selalu menggunakan kekuatan tembakan maksimal, yaitu 3, ketika energi cukup untuk menembak.

3) Fungsi solusi

Fungsi solusi digunakan untuk memilih keputusan terbaik yang dalam algoritma ini memprioritaskan kekuatan terbesar agar daya rusak tembakan besar.

4) Fungsi seleksi

Fungsi seleksi pada algoritma ini selalu memilih untuk menggunakan kekuatan tembakan terbesar, yakni 3, kecuali ketika energi tidak mencukupi, bot tidak melakukan tembakan.

5) Fungsi kelayakan

Fungsi kelayakan pada algoritma ini memastikan bot menembak ketika mendeteksi ada musuh pada radarnya yang berputar atau menabrak/ditabrak lawan serta memiliki energi yang cukup untuk menembak dengan kekuatan terbesar.

6) Fungsi objektif

Fungsi objektif pada bot ini bertujuan untuk memaksimalkan kekuatan tembakan demi meningkatkan peluang merusak musuh sebanyak mungkin dalam waktu yang singkat.

b. Analisis Efisiensi Solusi

Greedy by shot amount diaplikasikan dalam fungsi `OnScannedBot()` untuk mendeteksi kembali posisi bot lawan setiap kali selesai melakukan tembakan sehingga jumlah tembakan maksimum. Fungsi ini hanya melakukan serangkaian operasi sederhana, seperti pengecekan kondisi posisi bot dengan kompleksitas O(1) dan eksekusi perintah menembak bot lawan dengan kekuatan

terbesar yang juga berjalan dalam waktu konstan O(1). Akan tetapi, eksekusi berulang kali pada fungsi ini berdasarkan n kali bot mendeteksi lawan menyebabkan kompleksitas keseluruhan strategi ini menjadi O(n).

c. Analisis Efektivitas Solusi

Solusi ini efektif ketika:

1. Bot musuh memiliki pola pergerakan yang repetitif atau statis sehingga memungkinkan akurasi tembakan berdaya tinggi tersebut menjadi lebih tinggi.
2. Bot memiliki cukup energi sehingga dapat menembak terus-menerus tanpa kehabisan energi.

Solusi ini kurang efektif ketika:

1. Bot musuh bergerak secara cepat dan acak sehingga tembakan berkekuatan besar yang lebih lambat sulit mengenai target.
2. Energi bot rendah sehingga menghambat kemampuan bot untuk menembak secara terus-menerus.

3.3. Strategi Greedy Pilihan (Bot Terpilih)

Dari keempat bot dengan solusi greedy yang dikembangkan, Levi dipilih sebagai bot utama dari kelompok EREMIKA karena memiliki strategi yang paling efektif dan unggul dibandingkan lainnya. Dengan kombinasi pendekatan yang agresif dan adaptif, Levi mampu menyesuaikan taktiknya berdasarkan situasi pertempuran, baik dalam hal akurasi serangan maupun pengelolaan energi. Kemampuannya dalam menyeimbangkan serangan jarak jauh dan jarak dekat, serta pengambilan keputusan yang optimal terhadap kondisi musuh (energi), menjadikannya pilihan terbaik untuk mewakili kelompok EREMIKA.

BAB 4

IMPLEMENTASI DAN PENGUJIAN

4.1. Implementasi

4.1.1. Bot Alternatif 1: Levi

4.1.3.1. Fungsi Utama (Main)

```
function Main()
   call new Levi().Start() // Memulai bot Levi
end function
```

4.1.3.2. Konstruktor

```
function Levi ()
   call BotInfo.FromFile("Levi.json")
   // Menggunakan konfigurasi dari file JSON
end function
```

4.1.3.3. Prosedur Run()

```
procedure Run()
    // Mengganti warna bot
    warna body bot ← merah
    warna turret bot ← hitam
    warna radar bot ← hitam
    warna scan bot ← hitam
    warna peluru bot ← merah

    while bot masih berjalan do
        putar badan ke kiri sejauh 10000
        MaxSpeed ← 5
        maju sejauh 10000
    endwhile
end procedure
```

4.1.3.4. Prosedur OnScannedBot()

```
procedure OnScannedBot(ScannedBotEvent e)
    hitung jarak ke bot lawan
    if jarak < 100 then
   call TurnToFaceTarget(e.X, e.Y)// Fungsi untuk
mengarahkan bot ke arah lawan
        maju sejauh jarak + 5 untuk menabrak lawan
    else if jarak < 500 then
        tembak dengan kekuatan maksimum(3)
```

```
end procedure
```

4.1.3.5. Prosedur OnHitBot()

```
procedure OnHitBot(HitBotEvent e)
    call TurnToFaceTarget(e.X, e.Y)// Fungsi untuk
mengarahkan bot ke arah lawan
     if energi musuh > 16 then
          tembak dengan kekuatan maksimum(3)
     else if energi musuh > 10 then
          tembak dengan kekuatan 2
     else if energi musuh > 4 then
          tembak dengan kekuatan 1
     else if energi musuh > 2 then
          tembak dengan kekuatan .5
end procedure
```

4.1.3.6. Prosedur TurnToFaceTarget()

```
procedure TurnToFaceTarget(double x, double y)
     hitung bearing ke target
     if bearing >= 0 then
          turnDirection ← 1 (kiri)
     else
          turnDirection ← -1 (kanan)
     endif

       PUTAR ke kiri sebesar bearing // Mengarahkan
arah tembakan ke lawan
end procedure
```

4.1.2. Bot Alternatif 2: Eren

4.1.2.1 Fungsi Utama (Main)

```
function Main()
   call new Eren().Start() // Memulai bot eren
end function
```

4.1.2.2 Konstruktor (Main)

```
function Eren()
           call    BotInfo.FromFile("Eren.json")    //
Menggunakan konfigurasi dari file JSON
end function
```

4.1.2.3 Prosedur Run()

```
procedure Run()
    // Mengganti warna bot
    warna body bot ← hitam
    warna turret bot ← hitam
    warna radar bot ← putih
    warna peluru bot ← hijau
    warna scan bot ← hijau

    ambil ukuran arena (arenaWidth, arenaHeight)

    while bot masih berjalan do
        call MoveInCircleRandom()  // Fungsi untuk
gerak dalam pola melingkar acak
    endwhile
end procedure
```

4.1.2.4 Prosedur MoveInCircleRandom()

```
procedure MoveInCircleRandom()
    turnDirection ← acak {1 (kiri) atau -1 (kanan)}
    spinAngle ← acak antara 5 hingga 45 derajat

    putar bot ke kiri sebesar 10000 * turnDirection
      MaxSpeed ←kecepatan bot secara acak (antara
5-7)

    moveDistance ← 500
    call CheckWallSmoothing() // Untuk memastikan
tidak menabrak dinding
    maju sejauh moveDistance

    moveCounter++
    if moveCounter MOD 5 == 0 then
        call RandomMove()  // Setiap 5 perulangan
gerakan akan bergerak acak
    endif
end procedure
```

4.1.2.5 Prosedur RandomMove()

```
procedure RandomMove()
    randomAngle ← acak antara 0 hingga 360 derajat
// Sudut pergerakan yang acak
    moveDistance ← 200

    putar bot ke kanan sebesar randomAngle
    call CheckWallSmoothing()  // Memanggil fungsi
```

```
untuk menghindari tabrakan dengan dinding
    maju sejauh moveDistance
end procedure
```

4.1.2.6 Prosedur CheckWallSmoothing()

```
procedure CheckWallSmoothing()
    buffer ← 10  // Jarak aman dari dinding
    if posisi X < buffer then
        mundur sejauh 50
        putar ke kanan 45 derajat
    else if posisi X > arenaWidth - buffer then
        mundur sejauh 50
        putar ke kanan 45 derajat
    else if posisi Y < buffer then
        mundur sejauh 50
        putar ke kanan 45 derajat
    else if posisi Y > arenaHeight - buffer then
        mundur sejauh 50
        putar ke kanan 45 derajat
    endif
end procedure
```

4.1.2.7 Prosedur OnScannedBot()

```
procedure OnScannedBot(e)
    hitung jarak ke bot lawan
    firepower ← maximum(1, 3 - (jarak/400))
    // tembak dengan firepower yang dihitung
end procedure
```

4.1.2.8 Prosedur OnHitBot()

```
procedure OnHitBot(e)
    arahkan turret ke bot lawan
    tembak dengan kekuatan maksimum (3)
    mundur sejauh 30 unit(piksel) // Untuk
menghindari tabrakan lebih lanjut
    putar ke kanan 45 derajat
    maju sejauh 30 unit (piksel)
end procedure
```

4.1.2.9 Porsedur OnHitWall()

```
procedure OnHitWall(e)
    mundur sejauh 50 unit (piksel)
```

```
    putar ke kanan 90 derajat // Mengubah arah bot
agar tidak terjebak
    maju sejauh 100 unit (piksel)
end procedure
```

4.1.2.10 Prosedur TurnToFaceTarget()

```
procedure OnHitWall(e)
    hitung bearing ke target
    if bearing >= 0 then
        turnDirection ← 1 (kiri)
    else
        turnDirection ← -1 (kanan)
    endif

     PUTAR ke kiri sebesar bearing // Mengarahkan
arah tembakan ke lawan
end procedure
```

4.1.3. Bot Alternatif 3: Mikasa

4.1.3.1 Fungsi Utama (Main)

```
function Main()
   call new Mikasa().Start() // Memulai bot Mikasa
end function
```

4.1.3.2 Konstruktor (Main)

```
Function Mikasa()
   call BotInfo.FromFile("Mikasa.json")
   // Menggunakan konfigurasi dari file JSON
end function
```

4.1.3.3 Prosedur Run()

```
procedure Run()
    // Mengganti warna bot
    warna body bot ← merah
      warna turret bot ← hitam
    warna radar bot ← hitam
    warna scan bot ← hitam
    warna peluru bot ← merah

    call MoveToWall()

    while bot masih berjalan do
```

```
        if moveRight = true
           then maju sejauh moveDistance + angka
                acak di antara 0-49
        else
           mundur sejauh moveDistance + angka acak
                di antara 0-49
        moveRight ← !moveRight
        moveCount++

        if moveCount >= maxMoveCount
           then maju sejauh max(ArenaWidth,
                   ArenaHeight)
                putar badan ke kanan sejauh 90
                   derajat
                moveCount ← 0
        call Scan360Degrees()

    endwhile
end procedure
```

4.1.3.4 Prosedur MoveToWall()

```
procedure MoveToWall()
    angleToWall ← Direction mod 90
    putar badan sebesar angleToWall
    maju sejauh max(ArenaWidth, ArenaHeight))
    // pergi ke dinding terdekat

    // posisikan meriam agar siap menembak
    putar meriam sejauh 90 derajat
    putar badan sejauh 90 derajat
end procedure
```

4.1.3.5 Prosedur Scan360Degrees()

```
procedure Scan360Degrees()
    putar meriam sejauh 360 derajat
end procedure
```

4.1.3.6 Prosedur OnHitBot()

```
procedure OnHitBot(HitBotEvent e)
    bearing ← BearingTo(e.X, e.Y)
    putar badan sejauh bearing
    distance ← DistanceTo(e.X, e.Y)
    mundur sejauh 100 + angka acak di antara 0-49
    tembak dengan kekuatan sesuai fungsi
```

```
        CalculateFirePower(distance))
end procedure
```

4.1.3.7 Prosedur OnScannedBot()

```
procedure OnScannedBot(ScannedBotEvent e)
    distance ← DistanceTo(e.X, e.Y)
    tembak dengan kekuatan sesuai fungsi
        CalculateFirePower(distance))

    if distance < 150 dan Energy > 50
        then tembak lawan dengan kekuatan maksimum
             tembak lawan dengan kekuatan maksimum
    scan kembali posisi bot lawan
end procedure
```

4.1.3.8 Fungsi CalculateFirePower()

```
function CalculateFirePower(double distance)
    // tembak dengan kekuatan rendah jika energi
       rendah
    if Energy < 20
       → 1
      // tembak dengan kekuatan maks untuk jarak
dekat
    if distance < 200
       → 3
    // tembak dengan kekuatan sedang untuk jarak
       menengah
    if distance < 500
       → 2
    → 1
end function
```

4.1.4. Bot Alternatif 4: Armin

4.1.4.1 Fungsi Utama (Main)

```
function Main()
   call new Armin().Start() // Memulai bot Armin
end function
```

4.1.4.2 Konstruktor (Main)

```
function Armin()
   call BotInfo.FromFile("Armin.json")
```

```
   // Menggunakan konfigurasi dari file JSON
end function
```

4.1.2.3 Prosedur Run()

```
procedure Run()
    // Mengganti warna bot
    warna body bot ← biru muda (#799497)
    warna meriam/gun bot ← hijau tua (#323A39)
    warna turret bot ← krem (#EDD083)
    warna radar bot ← hijau tua (#323A39)
    warna scan bot ← hijau
    warna peluru bot ← kuning

    while bot masih berjalan do
        putar badan ke kiri sejauh 10000
        MaxSpeed ← 8
        maju sejauh 5000
    endwhile
end procedure
```

4.1.2.4 Prosedur OnScannedBot()

```
procedure OnScannedBot(ScannedBotEvent e)
    tembak lawan dengan kekuatan maksimum, yakni 3
    secara random putar meriam ke kiri atau kanan
       sejauh 30 derajat
    scan kembali posisi bot lawan
end procedure
```

4.1.2.5 Prosedur OnHitByBullet()

```
procedure OnHitByBullet(HitByBulletEvent e)
    secara random putar badan ke kiri atau kanan
       sejauh 60 atau 120 derajat yang dipilih
       secara random
    maju sejauh 200 atau 400 yang dipilih secara
       random
     // pindah posisi untuk menghindari tertembak
lagi
end procedure
```

4.1.2.6 Prosedur OnHitBot()

```
procedure OnHitBot(HitBotEvent e)
    tembak lawan dengan kekuatan maksimum, yakni 3
```

```
    if armin bot menabrak bot lain
        then maju sejauh 150
        // berusaha mengenai bot musuh lagi agar
           energi mereka berkurang

    else // armin bot yang ditabrak oleh bot lain
        putar badan ke kiri sejauh 30 derajat
        maju sejauh 200
        // kabur agar tidak ditabrak lagi
end procedure
```

4.2. Struktur Data

4.2.1 Struktur Data Repository

```
├── doc
│   └── Laporan_EREMIKA.pdf
├── src
│   ├── alternative-bots
│   │   ├── Armin
│   │   │   ├── Armin.cmd
│   │   │   ├── Armin.cs
│   │   │   ├── Armin.csproj
│   │   │   ├── Armin.json
│   │   │   └── Armin.sh
│   │   ├── Eren
│   │   │   ├── Eren.cmd
│   │   │   ├── Eren.cs
│   │   │   ├── Eren.csproj
│   │   │   ├── Eren.json
│   │   │   └── Eren.sh
│   │   └── Mikasa
│   │       ├── Mikasa.cmd
│   │       ├── Mikasa.cs
│   │       ├── Mikasa.csproj
│   │       ├── Mikasa.json
│   │       └── Mikasa.sh
│   └── main-bot
│       ├── Levi.cmd
│       ├── Levi.cs
│       ├── Levi.csproj
│       ├── Levi.json
│       └── Levi.sh
└── README.md
```

1. alternative-bots

   Folder berisi bot-bot alternatif selain bot utama, yakni Armin, Eren, dan Mikasa, dengan setiap bot memiliki file konfigurasi dan *source code*-nya sendiri.
2. main-bot

   Folder berisi bot utama, yaitu Levi, yang memiliki struktur seperti bot pada folder alternative-bots.
3. <nama-bot>.cmd

   File berisi skrip batch yang digunakan untuk menjalankan bot di sistem operasi Windows.
   <nama-bot>.cs

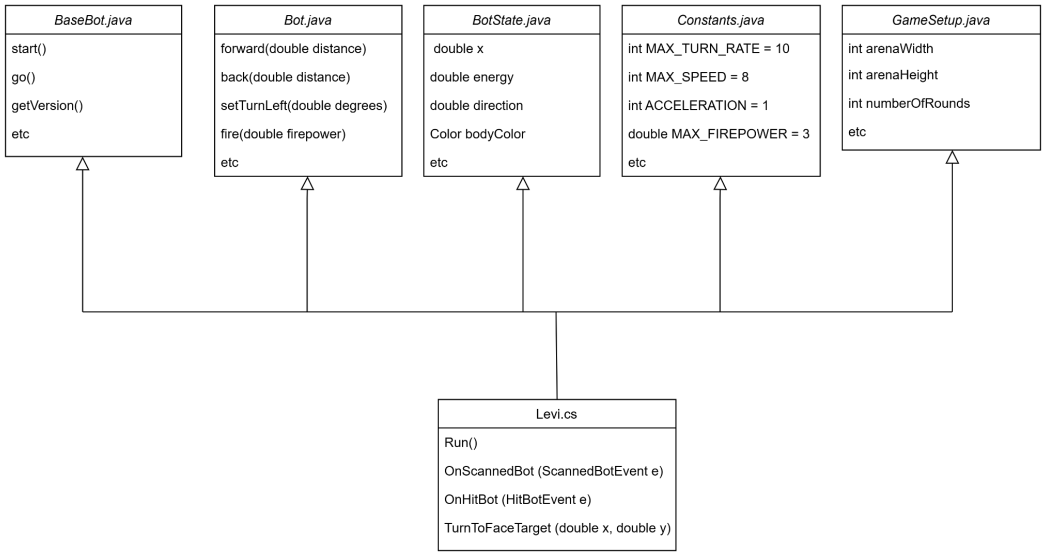
   File berisi logika utama bot yang memanfaatkan algoritma greedy, termasuk strategi menembak, menghindar, dan bergerak, dalam bahasa C#.
4. <nama-bot>.csproj

   File berupa file proyek C# yang digunakan oleh .NET. File ini berisi konfigurasi proyek bot seperti dependensi, framework yang digunakan, dan pengaturan build.
5. <nama-bot>.json

   File berisi informasi metadata tentang bot, seperti nama, deskripsi, warna bot, dan informasi lain yang digunakan oleh game engine Robocode Tank Royale.
6. <nama-bot>.sh

   File berisi skrip shell untuk menjalankan bot di sistem operasi berbasis Unix/Linux.

4.2.2 Struktur Data LeviBot



Struktur kode bot Levi menggunakan konsep inheritance untuk modularitas dan kemudahan pengembangan. BaseBot berperan sebagai kelas dasar yang menyediakan fungsi umum seperti mulai. Bot mengimplementasikan strategi permainan menyediakan metode untuk pergerakan dan aksi pertempuran. BotState digunakan untuk menyimpan informasi kondisi bot, sementara Constants berisi nilai tetap yang digunakan dalam perhitungan strategi. GameSetup bertanggung jawab untuk menangani pengaturan arena. Akhirnya, LeviBot adalah implementasi spesifik dari bot Levi, yang mewarisi semua fungsionalitas dari Bot dan dijalankan sebagai bot utama dalam pertandingan.

4.2.2.1 Variabel Global

Variabel global yang dideklarasikan di dalam class Levi adalah `int turnDirection`, variabel ini digunakan untuk menentukan arah putaran bot. Nilai default adalah 1, bahwa bot akan berputar searah jarum jam.

4.2.2.2 Konstruktor

```
public Levi(): base (BotInfo.FromFile("eremika.json"))
```

Konstruktor ini menginisialisasi bot menggunakan konfigurasi yang terdapat dalam eremika.json yang berisi informasi dasar bot seperti nama, *authors*, deskripsi, dan spesifikasi lainnya.

4.2.2.3 Main Method

```
Static void Main (string[] args)
```

Metode utama yang digunakan untuk memulai bot dengan menjalankan new Levi().Start();

4.2.2.4 Prosedur Run

```
public override void Run()
```

Prosedur (method) berisi loop utama yang terus berjalan selama bot aktif. Bot diberi warna di tiap komponen bot (BodyColor, TurretColor, dan lainnya). Bot juga diinisialisasi akan berputar (`SetTurnLeft`) dan bergerak maju (`Forward`) dengan kecepatan maksimum (`MaxSpeed`).

4.2.2.5 Prosedur OnScannedBot

```
public override void OnScannedBot(ScannedBotEvent e)
```

Prosedur (method) akan dijalankan saat radar bot mendeteksi musuh. Jika jarak musuh <100 unit, bot akan bergerak maju mendekati musuh dan menghadap ke arahnya (`TurnToFaceTarget(e.X, e.Y)`). Namun, jika jarak musuh antara 100-500 unit, bot akan langsung menembak dengan kekuatan penuh (`Fire(3)`).

4.2.2.6 Prosedur OnHitBot

```
public override void OnHitBot(HitBotEvent e)
```

Prosedur (method) akan dijalankan saat bot bertabrakan dengan bot lain. Bot akan menghadap langsung ke lawan (`TurnToFaceTarget(e.X, e.Y)`) kemudian serangan dilakukan berdasarkan energi musuh.

4.2.2.7 Prosedur TurnToFaceTarget

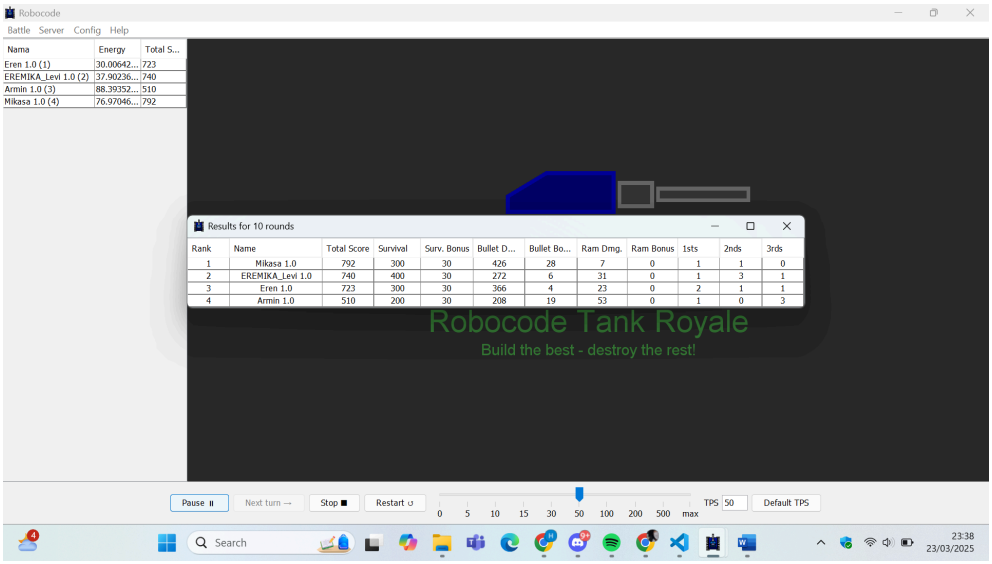
```
private void TurnToFaceTarget(double x, double y)
```

Prosedur (method) yang digunakan agar bot dapat berputar dan menghadap langsung ke posisi musuh berdasarkan koordinat (x,y).

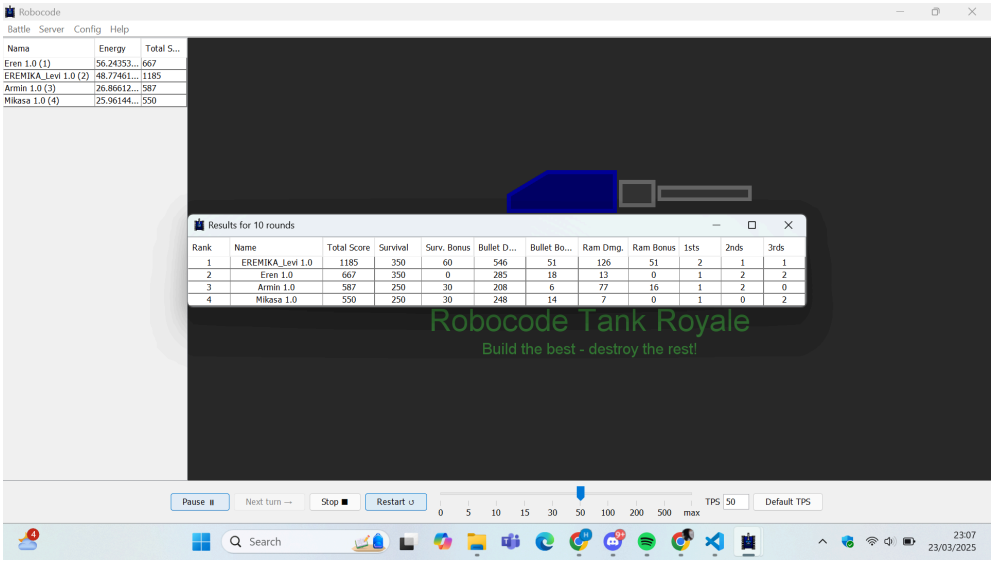
1. `var bearing = BearingTo(x,y);` digunakan untuk menghitung sudut bot terhadap musuh.
2. `turnDirection = bearing >= 0? 1 : -1;` digunakan untuk menentukan arah putaran berdasarkan sudut.
3. `TurnLeft (bearing);` digunakan untuk memutar bot ke arah musuh.

4.3. Pengujian

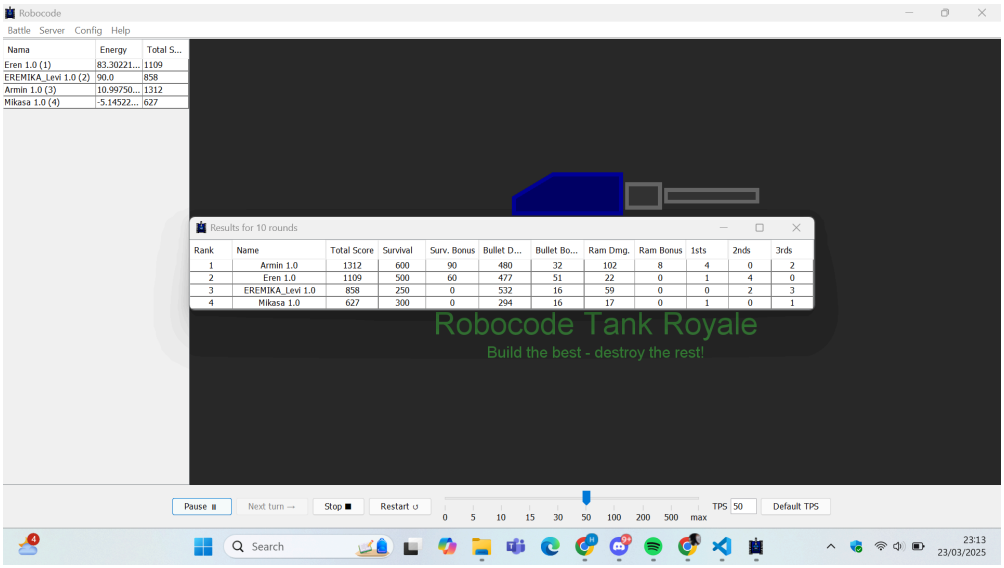
4.3.1. Pengujian 1



4.3.2. Pengujian 2



4.3.3. Pengujian 3



4.3. Analisis Hasil Pengujian

Berdasarkan hasil pengujian, strategi greedy pada LeviBot sering kali menghasilkan performa optimal. Namun, terdapat beberapa kondisi di mana strategi ini tidak mencapai nilai optimal. Salah satu penyebab utamanya adalah ketika ErenBot, ArminBot, dan LeviBot bergerak dalam pola melingkar. Dalam kondisi ini, jika hanya tersisa dua dari mereka, bot-bot tersebut akan kesulitan untuk saling mengalahkan karena terus bergerak tanpa ada yang dapat memberikan serangan efektif. Selain itu, ketika LeviBot ditabrak oleh bot lain, kemenangan dalam duel 1v1 sangat bergantung pada sisa energi yang dimilikinya, karena pertempuran akan berakhir dengan saling menembak hingga salah satu kehabisan energi terlebih dahulu. Kekurangan lain dari LeviBot adalah ketika posisinya awalnya berada di dekat dinding, ia cenderung menabrak dinding secara berulang, yang menyebabkan kehilangan energi secara signifikan dan mengurangi peluangnya untuk bertahan lebih lama dalam pertandingan.

BAB 5
KESIMPULAN DAN SARAN

5.1. Kesimpulan

Dalam tubes ini, kelompok EREMIKA berhasil menciptakan empat bot, yang terdiri atas satu bot utama dan tiga bot alternatif, dengan heuristik algoritma greedy yang berbeda-beda. Bot utama dari tubes ini bernama Levi yang menggunakan pendekatan greedy by shot accuration dan greedy by enemy's energy. Eren bot memanfaatkan pendekatan greedy by shot velocity dan greedy by avoiding wall collision. Mikasa bot memanfaatkan pendekatan greedy by collision response dan greedy by enemy detection. Armin bot memanfaatkan pendekatan greedy by shot amount dan greedy by shot power.

Secara keseluruhan, dapat disimpulkan bahwa penerapan algoritma greedy pada bot Robocode Tank Royale tidak mampu memberikan solusi yang terjamin mangkus dan sangkil. Setiap pendekatan greedy memiliki kelebihan dan kekurangannya tersendiri. Akibatnya, masing-masing bot efektif pada kondisi permainan tertentu.

5.2. Saran

Dari Tugas Besar 1 IF2211 Strategi Algoritma ini, kami menyarankan akan perlunya eksplorasi variasi pendekatan algoritma greedy yang lebih luas lagi. Hal tersebut dapat mengarah pada terciptanya algoritma bot Robocode Tank Royale yang lebih terjamin mangkus. Selain itu, pengaplikasian algoritma greedy pada keempat bot yang telah diciptakan kelompok EREMIKA dapat dioptimalkan demi menghasilkan bot yang lebih baik.

LAMPIRAN

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.	✓	

Tautan Repository GitHub

https://github.com/adndax/Tubes1_EREMIKA

Tautan Video

https://youtu.be/un59QAZ0I1U?si=KvdQy5rxRkA1dxUc

DAFTAR PUSTAKA

Microsoft. Tanpa Tahun. ".NET Platform". [*Online*]. Tersedia: https://dotnet.microsoft.com/en-us/. [18 Maret 2025].

Munir, R. 2025. "Algoritma Greedy (Bagian 1)". [*Online*]. Tersedia: https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2024-2025/04-Algoritma-Greedy-(2025)-Bag1.pdf. [20 Maret 2025].

Munir, R. 2025. "Algoritma Greedy (Bagian 2)". [*Online*]. Tersedia: https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2024-2025/05-Algoritma-Greedy-(2025)-Bag2.pdf. [20 Maret 2025].

Munir, R. 2025. "Algoritma Greedy (Bagian 3)". [*Online*]. Tersedia: https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2024-2025/06-Algoritma-Greedy-(2025)-Bag3.pdf. [20 Maret 2025].

Munir, R. 2025. "Tugas Besar 1: Pemanfaatan Algoritma Greedy dalam Pembuatan Bot Permainan Robocode Tank Royale". [*Online*]. Tersedia: https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2024-2025/Tubes1-Stima-2025.pdf. [17 Maret 2025].

Robocode Developers. Tanpa Tahun. "Robocode Tank Royale". [*Online*]. Tersedia: https://robocode-dev.github.io/tank-royale/. [17 Maret 2025].

RoboWiki. Tanpa Tahun. "RoboWiki". [*Online*]. Tersedia: https://robowiki.net/wiki/Main_Page. [20 Maret 2025].