

LAPORAN TUGAS BESAR 1

IF2211 STRATEGI ALGORITMA



Disusun Oleh:

Kelompok 50 - KepesetPolisiTidur

Naufarrel Zhafif Abhista (13523149)

Hasri Fayadh Muqaffa (13523156)

I Made Wiweka Putera (13523160)

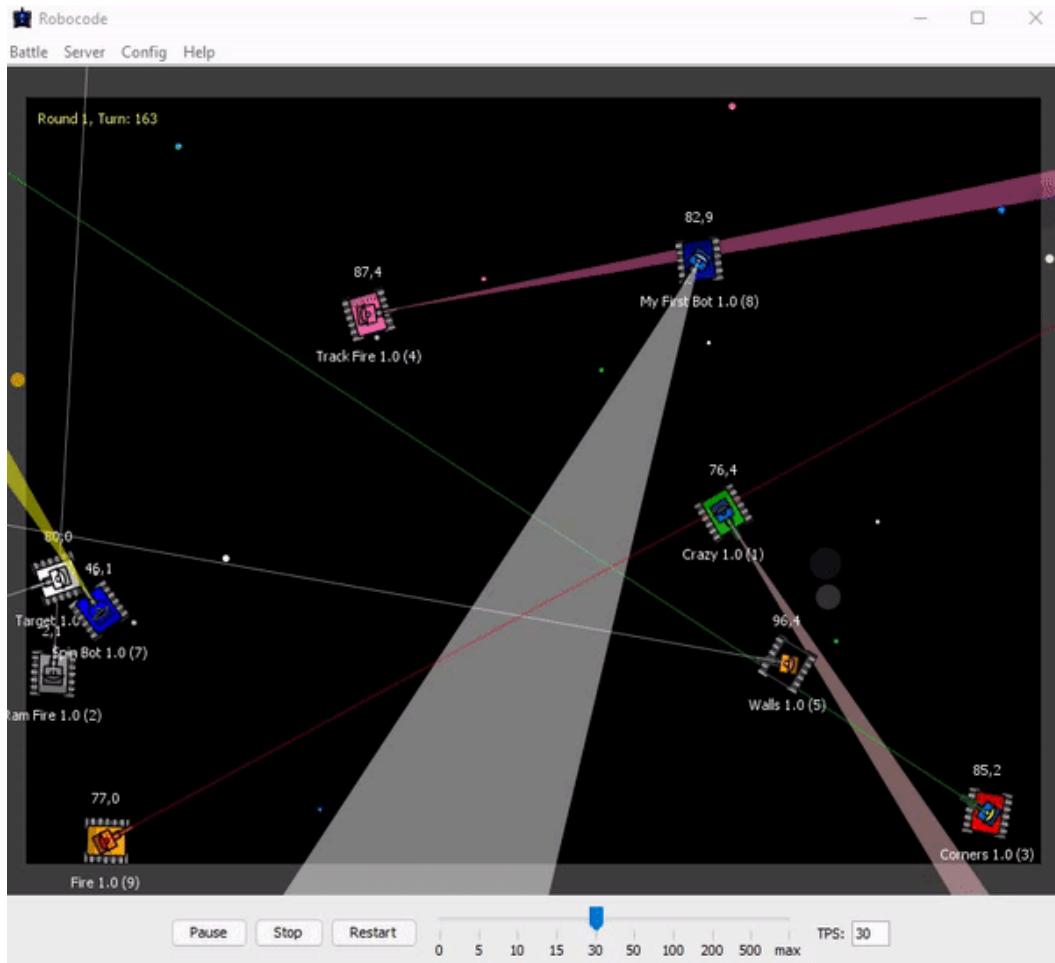
**SEKOLAH TEKNIK ELEKTRO DAN INFORMATIKA
INSTITUT TEKNOLOGI BANDUNG
SEMESTER 2 TAHUN 2024/2025**

DAFTAR ISI

DAFTAR ISI.....	1
BAB I	
DESKRIPSI TUGAS.....	2
BAB II	
LANDASAN TEORI.....	9
2.1. Dasar Teori (Algoritma Greedy) Secara Umum.....	9
2.2. Cara Kerja Program.....	9
BAB III	
APLIKASI STRATEGI GREEDY.....	11
3.1. Proses Mapping Persoalan Robocode dengan Algoritma Greedy.....	11
3.2. Algoritma Greedy pada Robocode Pertama.....	16
3.2. Algoritma Greedy pada Robocode Kedua.....	17
3.3. Algoritma Greedy pada Robocode Ketiga.....	18
3.4. Algoritma Greedy pada Robocode Keempat.....	19
3.5. Analisis Efisiensi dan Efektivitas dari Berbagai Robocode.....	20
3.6. Pemilihan Algoritma Greedy.....	23
BAB IV	
IMPLEMENTASI DAN PENGUJIAN.....	25
4.1. Struktur Program.....	25
4.2. Robocode 1.....	26
4.3. Robocode 2.....	29
4.4. Robocode 3.....	32
4.5. Robocode 4.....	34
4.6. Pengujian Semua Robocode.....	36
4.7. Analisis Pengujian.....	37
BAB V	
KESIMPULAN DAN SARAN.....	39
Kesimpulan.....	39
Saran.....	39
LAMPIRAN.....	40
DAFTAR PUSTAKA.....	41

BAB I

DESKRIPSI TUGAS



Gambar 1 Robocode Tank Royale

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.

Perlu diperhatikan bahwa [API \(Application Programming Interface\)](#) bot resmi secara otomatis mengirimkan niat bot ke server di balik layar, sehingga Anda tidak perlu mengkhawatirkannya, kecuali jika Anda membuat API Bot sendiri.

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.

Perlu diketahui bahwa game engine yang akan digunakan pada tugas besar ini tidak mengikuti aturan default mengenai komponen Round & Turns.

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 melewatkkan turn tersebut. Jika bot melewatkkan 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.

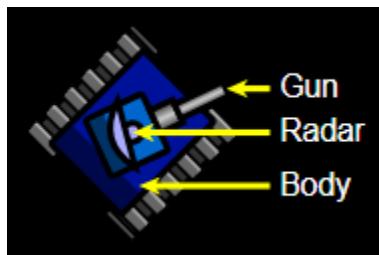
6. Tabrakan

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:



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 penggereman 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.

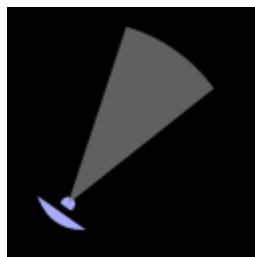
Bagian paling lambat adalah tubuh tank, yang dalam kondisi terbaik dapat berputar hingga 10 derajat per giliran. Namun, ini bergantung pada kecepatan bot saat ini. Semakin cepat bot bergerak, semakin lambat kemampuannya untuk berbelok.

Perlu diperhatikan bahwa tidak ada energi yang dikonsumsi saat bot bergerak atau berbelok.

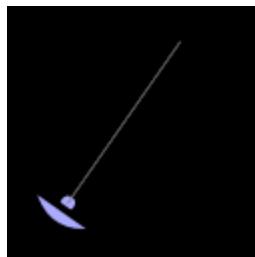
10. Pemindaian

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 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 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”.

Source Code untuk game engine dan template bot telah disediakan pada tautan berikut.

[tubes1-if2211-starter-pack](#)

Adapun panduan mengenai cara menjalankan game engine, membuat bot, dan melihat referensi API dapat dilihat melalui tautan berikut.

 Get Started With Robocode

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

Algoritma greedy adalah metode yang membentuk solusi langkah per langkah dengan memilih opsi yang memberikan keuntungan maksimal pada setiap langkah, tanpa mempertimbangkan konsekuensi jangka panjang. Pendekatan ini berfokus pada pencapaian solusi optimal lokal dengan harapan bahwa serangkaian keputusan optimal lokal akan menghasilkan solusi optimal global.

Prinsip utama algoritma greedy adalah "ambil apa yang bisa kamu dapatkan sekarang" ("take what you can get now!"). Artinya, pada setiap langkah, algoritma memilih opsi terbaik yang tersedia saat itu tanpa memperhatikan dampaknya pada langkah-langkah berikutnya. Meskipun algoritma greedy seringkali tidak menjamin solusi optimal global, algoritma ini dapat memberikan solusi yang mendekati optimal dalam waktu yang relatif cepat.

2.2. Cara Kerja Program

Program yang dikembangkan merupakan bot berbasis algoritma greedy untuk Robocode Tank Royale. Bot akan digunakan untuk bertarung dalam arena dengan memanfaatkan strategi greedy. Berikut adalah tahapan kerja program secara umum:

1. Pengembangan Program

- Bot dikembangkan menggunakan bahasa pemrograman C# dengan file utama berekstensi .cs.
- Proyek ini didukung oleh file konfigurasi .csproj yang mengatur dependensi dan struktur proyek.
- File .json digunakan sebagai informasi tambahan dari bot yang akan dibaca oleh server a.k.a. Robocode Engine (Nama bot, keterangan bot, negara, dan lain-lain).
- File .cmd dan .sh untuk melakukan *build* bot (dilakukan oleh Engine ketika di-boot).

2. Implementasi API Robocode Tank Royale

Agar dapat berfungsi di dalam game, bot memanfaatkan API dari Robocode Tank Royale, yang memungkinkan interaksi dengan server seperti:

- Pergerakan Bot: TurnRight(angle), TurnLeft(angle), Ahead(distance), Back(distance).
- Pemindaian Musuh: TurnGunRight(360), TurnGunLeft(360), GetScannedRobotEvent().
- Serangan: Fire(power), yang digunakan untuk menembakkan peluru dengan kekuatan tertentu.

- Respons terhadap Keadaan: GetEnergy(), GetVelocity(), dan OnHitByBullet(), untuk mengatur strategi berdasarkan kondisi bot.

3. Mengimplementasikan Algoritma Greedy ke dalam Bot

Dalam implementasinya, algoritma greedy diterapkan dengan cara mendekomposisi strategi bot ke dalam langkah-langkah yang selalu memilih keputusan terbaik di setiap tahapan tanpa mempertimbangkan dampak jangka panjang. Proses ini melibatkan beberapa aspek utama:

- Pemindaian dan Identifikasi Target

Bot melakukan pemindaian arena menggunakan scanner yang disediakan oleh API Robocode Tank Royale untuk mengidentifikasi musuh. Keputusan untuk menyerang atau bertahan dibuat berdasarkan data real-time, misal, berdasarkan jarak dan posisi lawan.

- Pemilihan Keputusan Optimal Secara Lokal

Bot selalu memilih aksi terbaik berdasarkan kondisi saat itu. Misalnya, jika musuh berada dalam jangkauan tembakan, bot akan langsung menyerang dengan kekuatan maksimal yang memungkinkan. Jika bot dalam posisi terancam, maka ia akan bermanuver untuk menghindari serangan.

- Pemanfaatan API yang Tepat

Implementasi strategi dilakukan dengan memanfaatkan fungsi bawaan API, seperti pergerakan (Forward, Back), rotasi turret (TurnGunRight, TurnGunLeft), serta sistem deteksi (DistanceTo, BearingTo). Setiap keputusan dieksekusi melalui pemanggilan API yang relevan sesuai dengan kondisi di arena.

4. Menjalankan Bot

Bot dijalankan menggunakan Robocode Tank Royale, dengan langkah-langkah:

- Menjalankan Custom Robocode Tank Royale Engine dengan ekstensi .jar.
- Mem-boot Parent Folder Bot yang berisi semua bot dalam sub-folder masing-masing. Langkah ini akan otomatis membuild bot sesuai dengan skrip .cmd atau .sh yang telah disediakan
- Mem-boot masing-masing bot dan menjalankan pertempuran.

BAB III

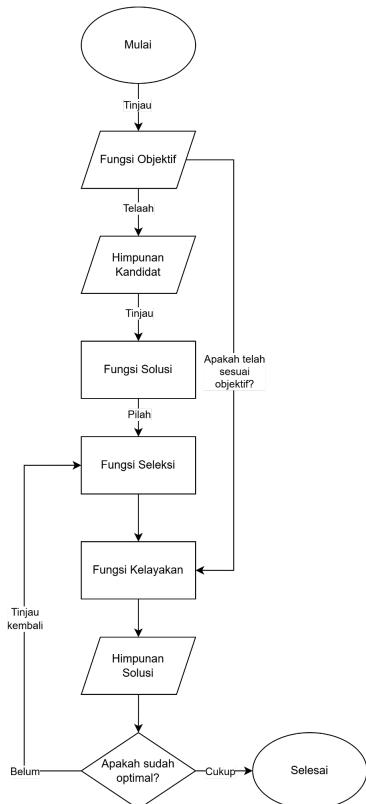
APLIKASI STRATEGI GREEDY

3.1. Proses Mapping Persoalan Robocode dengan Algoritma Greedy

Berdasarkan prinsip dari algoritma greedy, terdapat enam elemen algoritma greedy, yakni:

1. Himpunan kandidat, C : berisi kandidat yang akan dipilih pada setiap Langkah (misal: simpul/sisi di dalam graf, job, task, koin, benda, karakter, dsb)
2. Himpunan solusi, S : berisi kandidat yang sudah dipilih
3. Fungsi solusi: menentukan apakah himpunan kandidat yang dipilih sudah memberikan solusi
4. Fungsi seleksi (selection function): memilih kandidat berdasarkan strategi greedy tertentu. **Strategi greedy ini bersifat heuristik.**
5. Fungsi kelayakan (feasible): memeriksa apakah kandidat yang dipilih dapat dimasukkan ke dalam himpunan solusi (layak atau tidak)
6. Fungsi obyektif : memaksimumkan atau memminimumkan

Pada pengonsepan strategi greedy ini, kami menyusun elemen-elemen tersebut secara runtut, dengan alur yang tercantum pada gambar di bawah.



Pada tiap elemen, berikut penjelasannya.

Fungsi Objektif

Fungsi objektif adalah tujuan utama yang ingin dicapai. Dalam algoritma greedy umum, biasanya tujuannya adalah meminimalkan atau memaksimumkan. Dalam Robocode, bot dalam pertempuran **memaksimalkan** skor yang diperoleh selama pertarungan. Fungsi objektif mengarah pada pencapaian skor tinggi dengan mengoptimalkan berbagai aspek pertempuran. **Fungsi objektif** memastikan bahwa setiap aksi yang diambil oleh bot berkontribusi pada tujuan utama, yaitu **mencapai skor tertinggi**. Fungsi inilah yang akan menjadi basis utama penentuan algoritma greedy nantinya.

Himpunan Kandidat

Pada dasarnya, ada banyak strategi yang dapat dilakukan di Bot Robocode. Hal ini disebut dengan himpunan kandidat, yakni semua pilihan tindakan yang bisa diambil oleh bot pada satu titik waktu tertentu. Dalam Robocode, berikut merupakan himpunan kandidat yang telah ditinjau. Beberapa aksi juga telah digabung untuk memperkuat strategi yang dibentuk.

- Menembak musuh yang terlihat
- Menabrak musuh
- Menghindari tembakan musuh
- Konservasi energi

Fungsi Solusi

Fungsi solusi memastikan bahwa tindakan yang dipilih memenuhi tujuan, berdasarkan tiap kandidat yang telah diperiksa. **Fungsi solusi pada Robocode umumnya di-handle oleh API dari Robocode**. Untuk masing-masing kandidat, berikut adalah hal-hal yang harus diperhatikan:

- Menembak musuh yang terlihat: Jika bot memilih untuk menembak musuh, maka akan diperiksa, apakah **tembakan mengenai musuh** dan apakah **damage** yang diberikan sudah dihitung dengan benar (dihitung oleh API Robocode).
- Menabrak musuh: Jika bot memilih untuk menabrak musuh, akan diverifikasi, apakah **tabrakan tersebut berhasil** dan **memberikan Ram Damage** serta apakah musuh terbunuh untuk mendapatkan **Ram Damage Bonus**.
- Menghindari tembakan musuh: Jika bot bertahan hidup setelah beberapa musuh mati, fungsi solusi memeriksa apakah bot berhak mendapatkan **Survival Score** atau **Last Survival Bonus**.
- Konservasi energi: Jika bot memiliki energi yang cukup, bot dapat memilih untuk menyerang atau menabrak musuh. Namun, jika energi bot rendah, bot akan memilih untuk menghindari **tindakan yang menguras energi**, seperti menembak terlalu banyak atau menabrak musuh jika tidak diperlukan. Strategi konservasi energi tidak memiliki API *native* seperti ketiga strategi di atas, sehingga wajib diimplementasi secara mandiri.

Fungsi Seleksi

Fungsi seleksi memilih **aksi terbaik** dari himpunan kandidat berdasarkan strategi greedy yang diimplementasikan. Fungsi ini mengutamakan pilihan yang akan memberikan hasil optimal sesuai dengan **fungsi objektif**. Dalam fungsi seleksi ini juga, akan terlihat pendekatan *greedy* dengan heuristik yang berbeda. Pemilihan aksi bukan hanya berdasarkan satu kriteria, melainkan kombinasi dari beberapa kriteria. Sehingga, sebelum mencapai hasil dengan heuristik tersebut, akan dirumuskan aksi yang memungkinkan. Aksi-aksi tersebut kemudian digabungkan untuk membentuk heuristik yang berbeda.

Dalam konteks bot Robocode, berikut penjabaran aksi yang kami tinjau:

Menembak Musuh Yang Terlihat

- Bot dapat **menembak musuh dengan energi terendah**.
- Bot dapat **menembak musuh dengan jarak terdekat**.
- Bot dapat **menembak musuh yang pertama kali dilihat dalam radar**

Menabrak musuh

- Bot dapat **menabrak musuh dengan energi terendah**.
- Bot dapat **menabrak musuh dengan jarak terdekat**.
- Bot dapat **menabrak musuh yang pertama kali dilihat**.

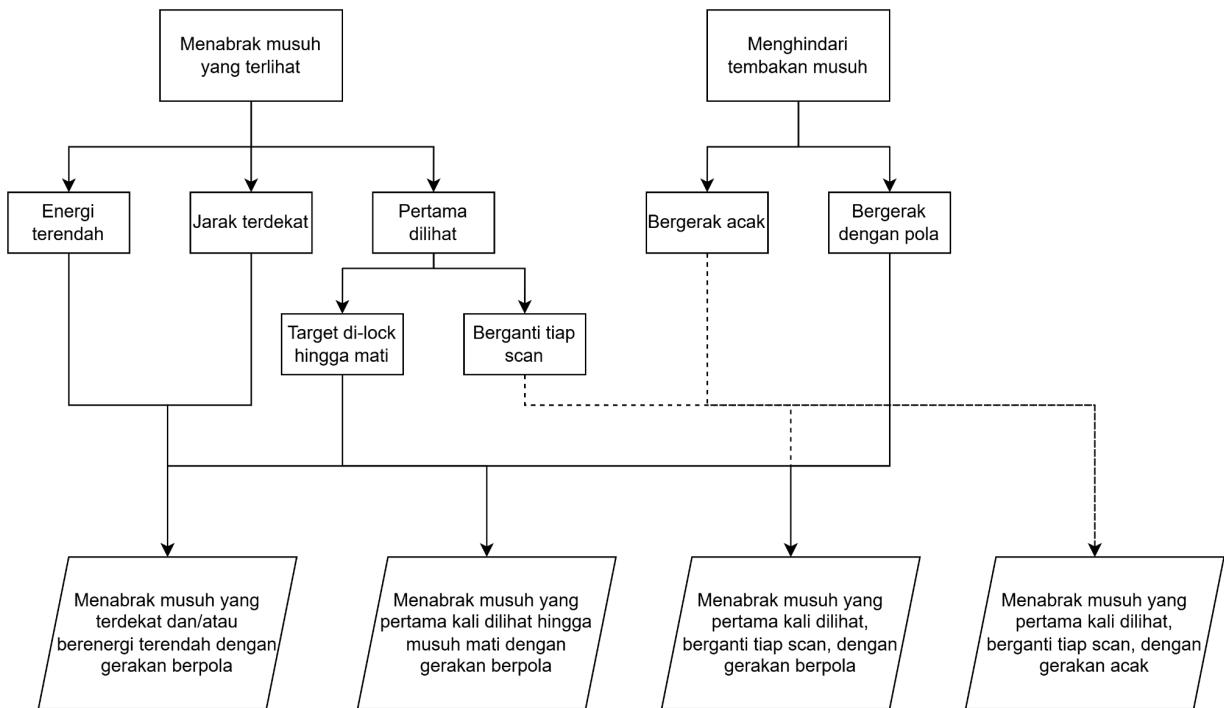
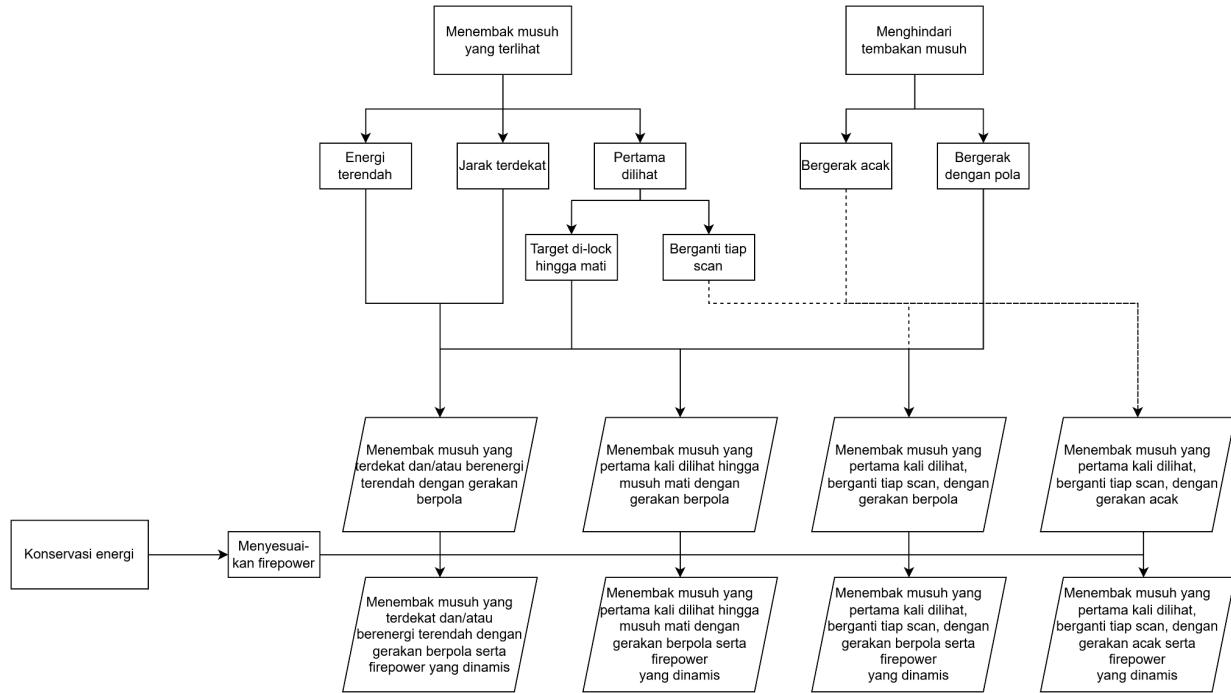
Menghindari tembakan musuh

- Bot dapat **bergerak secara acak**.
- Bot dapat **bergerak dengan pola tertentu**.

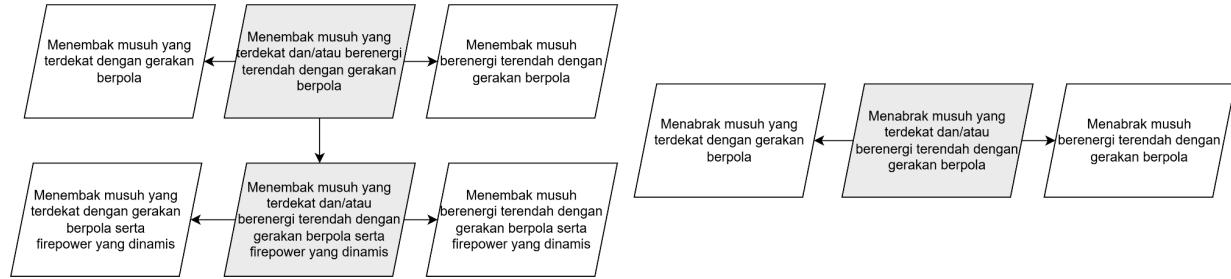
Konservasi energi

- Bot dapat menyesuaikan **firepower** terhadap **jarak dari target yang dipilih**.

Setelah semua aksi tersebut dijabarkan, dilakukan penggabungan antar aksi untuk menciptakan heuristik yang berbeda. Diagram di bawah memvisualisasikan penggabungan ini.



Secara keseluruhan, kami meninjau 12 kemungkinan heuristik dari penggabungan kriteria yang telah disebutkan (ditandai dengan kotak jajar genjang di tiap diagram). Dengan catatan, untuk bot hasil kombinasi energi terendah dan/atau jarak terdekat dapat dipisah lagi.



*Abu-abu = hasil dari diagram sebelumnya

Jika dipisah, akan terdapat enam tambahan kemungkinan, sehingga total akan terdapat 18 kemungkinan heuristik. Masing-masing strategi memilih aksi terbaik berdasarkan kombinasi faktor-faktor seperti energi musuh, jarak terdekat, urutan pertama kali dilihat, dan konservasi energi. Secara keseluruhan, penggabungan kriteria ini memberikan fleksibilitas dalam strategi, memungkinkan bot untuk menyesuaikan perilakunya tergantung pada kondisi yang ada. Fungsi seleksi diimplementasikan dengan mengutamakan kriteria yang paling relevan dalam setiap situasi, baik itu memaksimalkan *damage*, menghindari kerusakan, atau menghemat energi.

Fungsi Kelayakan

Walaupun banyak kemungkinan yang telah diperoleh, tidak semuanya *feasible* untuk dimasukkan ke dalam himpunan solusi. Maka dari itu, ditinjau kembali berbagai aksi yang telah digabungkan. Berdasarkan hasil diskusi kelompok, berikut adalah kriteria tambahan yang akan kami khususkan:

1. Penggunaan tabrakan/*ram* sebagai alat serang utama akan sangat menghabiskan energi, baik secara lokal (*konsep greedy*) maupun global. Sehingga, kami memutuskan untuk tidak memasukkan elemen penabrak ke dalam himpunan solusi.
2. Bot yang disusun sebaiknya tidak terlalu kompleks. Secara teori, meskipun bot yang sangat adaptif terhadap medan pertempuran mungkin ideal, kami merasa bahwa kondisi pertempuran yang acak sudah cukup *menantang*, dalam artian, nantinya akan terlalu banyak hasil dari bot lain yang memengaruhi jalannya permainan. Oleh karena itu, kami memutuskan bahwa bot yang kompleks hanya akan dibuat satu, yaitu bot dengan gerakan acak dan adaptasi firepower.

Dengan dua kriteria ini, kami memilih **himpunan solusi** yang tetap mengutamakan **heuristik** yang sederhana namun efektif dalam berbagai skenario.

Himpunan Solusi

Dari kriteria tambahan yang diberikan oleh fungsi kelayakan, empat strategi yang akan kami pilih:

1. Bot yang menembak musuh terdekat dengan gerakan berpola (*fixed movement*).
2. Bot yang menembak musuh terdekat dan berenergi terendah dengan gerakan berpola (*fixed movement*).

3. Bot yang menembak musuh yang pertama kali dilihat, kemudian *di-lock* hingga mati dengan gerakan berpola (*fixed movement*).
4. Bot yang menembak musuh yang pertama kali dilihat, kemudian dilakukan *scan* setelah penembakan untuk penggantian target, dengan *firepower* adaptif dan gerakan acak (*random movement*).

Secara keseluruhan, algoritma greedy yang diterapkan pada bot Robocode mengikuti urutan langkah yang dimulai dari menentukan tujuan (fungsi objektif), lalu memilih aksi terbaik berdasarkan berbagai kandidat tindakan. Fungsi seleksi memastikan bot memilih strategi yang memberikan hasil terbaik dalam jangka pendek dan panjang, sementara fungsi kelayakan memverifikasi bahwa setiap aksi yang dipilih dapat dilaksanakan dengan baik dalam kondisi permainan saat itu. Akhirnya, himpunan solusi menyimpan aksi-aksi yang telah dipilih dan diimplementasi lebih lanjut oleh kami.

Dengan demikian, algoritma greedy ini membantu bot dalam membuat keputusan yang optimal dalam waktu yang singkat, memaksimalkan skor berdasarkan berbagai parameter yang relevan selama pertempuran.

3.2. Algoritma Greedy pada Robocode Pertama

Bot utama, KepelesetPolisiTidur, mengimplementasikan strategi greedy berdasarkan jarak dalam menentukan targetnya. Tujuan utama dari bot ini adalah memilih target yang paling rentan berdasarkan faktor jarak terdekat, yaitu target yang dapat ditembak dengan kesempatan akurasi yang lebih tinggi.

Untuk menjalankan strategi ini, bot akan menghitung jarak dari dirinya ke setiap musuh yang terdeteksi. Bot akan selalu memilih dan menembak target dengan jarak terdekat di antara semua musuh yang terdeteksi. Setelah target terpilih, bot akan menyesuaikan pergerakannya untuk mendekati target secara optimal, yaitu mencapai jarak yang memungkinkan tembakan menjadi lebih akurat tanpa membahayakan bot ini sendiri.

Tahapan algoritma dari bot pertama adalah sebagai berikut:

1. Pemilihan Target:

- a. Bot melakukan pemindaian terhadap semua musuh di arena.
- b. Untuk setiap musuh yang terdeteksi, bot menghitung jarak ke musuh tersebut.
- c. Bot selalu mengupdate dan menyimpan informasi musuh dengan jarak terdekat sebagai target penembakan saat ini.
- d. Jika target yang dipilih mati, bot akan mengulang proses pemilihan target.

2. Pergerakan Menuju Target:

- a. Bot akan menyesuaikan arah dan kecepatan berdasarkan jarak ke target:
 - i. Jarak > 150 : Bot bergerak maju dengan kecepatan penuh untuk memperpendek jarak.

- ii. Jarak antara 50 – 150: Bot bergerak maju dengan kecepatan sedang untuk menjaga keseimbangan antara akurasi tembakan dan keamanan.
- iii. Jarak < 50: Bot mundur untuk menghindari pertempuran jarak dekat yang berisiko tinggi.
- b. Bot selalu mengarahkan badannya ke target agar dapat segera menyerang setelah berada dalam posisi optimal.

3. Penargetan dan Penembakan

- a. Rotasi Turret: Bot terus memutar senjata sebesar 360° untuk melakukan pemindaian musuh.
- b. Jika bot mendeteksi target yang telah dipilih, bot akan langsung menembak.

4. Penanganan Kolisi

- a. Tabrakan dengan Dinding: Jika bot menabrak dinding, bot akan bergerak mundur sejauh 50 unit. Setelah itu, bot akan berbelok 90° ke kanan.
- b. Tabrakan dengan Bot Lain: Jika bot bertabrakan dengan lawan, bot akan mundur sejauh 100 unit. Setelah itu, bot akan berbelok 45° ke kanan.

3.2. Algoritma Greedy pada Robocode Kedua

Bot kedua, KepesetPolisiTidurAlt1, mengimplementasikan strategi greedy berdasarkan kombinasi HP (Energy) dan jarak dalam menentukan targetnya. Tujuan utama dari bot ini adalah memilih target yang paling rentan berdasarkan dua faktor:

1. HP terendah : Target yang dari segi HP paling cepat dikalahkan.
2. Jarak terdekat : Target yang dapat ditembak dengan kesempatan akurasi yang lebih tinggi.

Untuk menjalankan strategi ini, bot akan menghitung skor untuk setiap musuh yang terdeteksi dengan formula:

$$\text{score} = \text{enemyHP} + \text{distance}$$

Bot akan selalu memilih dan menembak target dengan skor terendah di antara semua musuh yang terdeteksi. Setelah target terpilih, bot akan menyesuaikan pergerakannya untuk mendekati target secara optimal, yaitu mencapai jarak yang memungkinkan tembakan menjadi lebih akurat tanpa membahayakan bot ini sendiri.

Tahapan algoritma dari bot pertama adalah sebagai berikut:

1. Pemilihan Target:

- e. Bot melakukan pemindaian terhadap semua musuh di arena.
- f. Untuk setiap musuh yang terdeteksi, bot menghitung skor ($\text{enemyHP} + \text{distance}$) dari musuh tersebut.
- g. Bot selalu mengupdate dan menyimpan informasi musuh dengan skor terendah sebagai target penembakan saat ini.
- h. Jika target yang dipilih mati, bot akan mengulang proses pemilihan target.

2. Pergerakan Menuju Target:

- c. Bot akan menyesuaikan arah dan kecepatan berdasarkan jarak ke target:

- i. Jarak > 150 : Bot bergerak maju dengan kecepatan penuh untuk memperpendek jarak.
- ii. Jarak antara $50 - 150$: Bot bergerak maju dengan kecepatan sedang untuk menjaga keseimbangan antara akurasi tembakan dan keamanan.
- iii. Jarak < 50 : Bot mundur untuk menghindari pertempuran jarak dekat yang berisiko tinggi.
- d. Bot selalu mengarahkan badannya ke target agar dapat segera menyerang setelah berada dalam posisi optimal.

3. Penargetan dan Penembakan:

- a. Rotasi Turret: Bot terus memutar senjata sebesar 360° untuk melakukan pemindaian musuh.
- b. Jika bot mendeteksi target yang telah dipilih, bot akan langsung menembak.

4. Penanganan Kolisi:

- a. Tabrakan dengan Dinding: Jika bot menabrak dinding, bot akan bergerak mundur sejauh 50 unit. Setelah itu, bot akan berbelok 90° ke kanan.
- b. Tabrakan dengan Bot Lain: Jika bot bertabrakan dengan lawan, bot akan mundur sejauh 100 unit. Setelah itu, bot akan berbelok 45° ke kanan.

3.3. Algoritma Greedy pada Robocode Ketiga

Bot ketiga, KepesetPolisiTidurAlt2, mengimplementasikan strategi greedy dengan pendekatan *first-scanned* target, yaitu memilih dan mengunci musuh pertama yang terdeteksi, lalu mempertahankan target tersebut sampai musuh itu mati.

Tujuan utama dari bot ini adalah menyederhanakan logika penargetan dengan tetap mempertahankan efektivitas tembakan. Dengan fokus pada satu target saja, bot dapat mengurangi waktu pengambilan keputusan dan memaksimalkan konsistensi tembakan ke arah target yang sudah dikenali.

Tahapan algoritma dari bot kedua adalah sebagai berikut:

1. Pemilihan Target:

- a. Bot melakukan pemindaian (scan) ke segala arah dengan memutar turret sebesar 360° .
- b. Ketika pertama kali mendeteksi musuh, bot langsung menyimpan ID musuh tersebut sebagai target.
- c. Target tidak akan diganti selama musuh tersebut masih hidup.
- d. Jika target mati, bot akan mengosongkan target dan menunggu musuh lain muncul untuk dikunci sebagai target baru.

2. Pergerakan Menuju Target:

Pergerakan memiliki algoritma yang sama dengan Bot Pertama.

3. Penargetan dan Penembakan:

Penargetan dan penembakan memiliki algoritma yang sama dengan Bot Pertama

4. Penanganan Kolisi:

Penanganan kolisi memiliki algoritma yang sama dengan Bot Pertama.

3.4. Algoritma Greedy pada Robocode Keempat

Bot keempat, KepelsetPolisiTidurAlt3, mengimplementasikan strategi greedy dengan pendekatan *Energy Optimization Firing* dan *Random Move*, yaitu bot akan menyesuaikan kekuatan tembakan (FIRE) berdasarkan energi yang dimiliki oleh bot dan juga jarak dengan bot musuh. Selain itu, bot ini juga memiliki perilaku untuk bergerak dengan acak agar tidak mudah ditembak oleh bot musuh.

Tujuan utama dari bot ini adalah melakukan optimasi pada tembakan yang dikeluarkan agar dapat memberikan *damage* yang maksimal dan melakukan penghematan energi. Tahapan algoritma dari bot kedua adalah sebagai berikut:

1. Pemilihan Target:

- a. Bot melakukan pemindaian (*scan*) ke segala arah dengan memutar *turret* sebesar 360° .
- b. Ketika pertama kali mendeteksi musuh, bot akan menyimpan sudut relatif ke musuh dan jarak ke musuh.
- c. Bot akan langsung menembak tanpa mempertimbangkan apakah ada musuh lain yang lebih lemah atau lebih dekat.
- d. Jika musuh terdeteksi, bot akan menyesuaikan tembakan berdasarkan energi yang dimiliki oleh bot, kecepatan musuh dan jarak musuh.
- e. Setelah menembak, bot akan melakukan pemindaian ulang.

2. Pergerakan Menuju Target:

- a. Bot bergerak secara acak yang melingkupi maju atau mundur dalam jarak acak (50 -150 unit), belok kanan atau kiri secara ajak dalam kisaran $15-90^\circ$.
- b. Bot akan mengulangi proses ini secara terus-menerus.

3. Penargetan dan Penembakan:

- a. Saat musuh terdeteksi, bot akan menghitung sudut relatif musuh dari *gun* dan menggerakan *gun* ke arah musuh.
- b. Bot akan melakukan penyesuaian kekuatan tembakan (FIRE) dengan ketentuan sebagai berikut:
 - Jika energi yang dimiliki oleh bot tinggal sedikit (<20), bot hanya akan menembak dengan kekuatan (FIRE) 1.
 - Jika musuh diam atau jaraknya dekat (<200 unit), bot akan menembak dengan kekuatan maksimal (FIRE) 3.

- Jika musuh dalam jarak menengah ($200 < x < 500$ unit), bot akan menembak dengan kekuatan (FIRE) 2.
- Jika energi yang dimiliki masih banyak (>20) dan musuh memiliki jarak yang jauh (>500 unit), bot akan menembak dengan kekuatan (FIRE) 1.

4. Penanganan Kolisi:

- a. Terkena Tembakan: Jika bot terkena tembakan, bot akan bergerak mundur sejauh 150 unit. Setelah itu, bot akan berbelok 45 derajat ke kanan.
- b. Tabrakan dengan Dinding: Jika bot menabrak dinding, bot akan bergerak mundur sejauh 150 unit. Setelah itu, bot akan berbelok 90 derajat ke kanan.
- c. Tabrakan dengan Bot Lain: Jika bot bertabrakan dengan lawan, bot akan mundur sejauh 150 unit. Setelah itu, bot akan berbelok 45 derajat ke kanan.

3.5. Analisis Efisiensi dan Efektivitas dari Berbagai Robocode

Analisis ini bertujuan untuk mengevaluasi efisiensi dan efektivitas masing-masing bot dalam mencapai tujuan utama, yaitu memaksimalkan skor dalam pertandingan.

Bot 1: Menembak Musuh Terdekat dengan Gerakan Berpola (*Fixed Movement*)

Efisiensi

- Kecepatan Pengambilan Keputusan
Algoritma ini relatif cepat, karena hanya memperhitungkan musuh terdekat tanpa memerlukan analisis tambahan terkait energi atau kondisi lain.
- Penggunaan Energi
Bot ini tidak mempertimbangkan konservasi energi secara eksplisit, sehingga mungkin menghabiskan energi lebih cepat meskipun gerakan berpola membantu mengurangi pengeluaran energi.
- Kompleksitas Waktu
O(n), di mana **n** adalah jumlah musuh yang terlihat. Bot akan memilih musuh terdekat di antara musuh yang terdeteksi.
- Kompleksitas Ruang
O(1) untuk penyimpanan tambahan (seperti posisi dan jarak), karena bot hanya perlu informasi dasar untuk pengambilan keputusan.

Efektivitas

- Memaksimalkan Damage
Bot ini dapat memberikan damage yang cukup besar terhadap musuh yang terdekat, tetapi bisa kurang efektif jika musuh tersebut terlalu kuat atau jika bot mudah terjebak dalam pola gerak.
- Kelemahan
Tidak ada pertimbangan terhadap energi musuh atau gerakan acak dari musuh, yang bisa mengurangi keefektifan bot dalam situasi dinamis.

Bot 2: Menembak Musuh Terdekat dan Berenergi Terendah dengan Gerakan Berpola (*Fixed Movement*)

Efisiensi

- Kecepatan Pengambilan Keputusan

Algoritma ini sedikit lebih kompleks daripada Bot 1, karena membutuhkan pengecekan energi musuh. Namun, ini masih relatif efisien karena hanya mengevaluasi dua kriteria (musuh terdekat dan energi).

- Penggunaan Energi

Bot ini lebih efisien dalam memilih musuh yang mudah dibunuh, yang berpotensi menghemat energi dalam jangka panjang.

- Kompleksitas Waktu

$O(n)$, di mana n adalah jumlah musuh yang terlihat. Bot harus memeriksa energi dan jarak dari tiap bot yang ada.

- Kompleksitas Ruang

$O(1)$ untuk penyimpanan tambahan (seperti posisi dan jarak). Meskipun ada dua kriteria yang diperiksa, informasi yang perlu disimpan untuk perhitungan tidak bertambah signifikan.

Efektivitas

- Memaksimalkan Damage

Memilih musuh dengan energi terendah dapat meningkatkan kemungkinan bot untuk membunuh musuh lebih cepat, tetapi jika jarak musuh terlalu jauh, bot mungkin kesulitan.

- Keuntungan

Memilih musuh dengan energi rendah membuat bot lebih efektif dalam pertempuran dengan mengurangi risiko kehabisan energi terlalu cepat.

- Kelemahan

Jika musuh yang memiliki energi rendah berpindah tempat atau bergerak lebih cepat, bot ini akan kesulitan untuk mempertahankan target.

Bot 3: Menembak Musuh yang Pertama Kali Dilihat, Kemudian Di-lock Hingga Mati dengan Gerakan Berpola (*Fixed Movement*)

Efisiensi

- Kecepatan Pengambilan Keputusan

Algoritma ini sedikit lebih lambat karena bot harus mengunci musuh dan mengikuti musuh hingga mati, namun proses ini cukup sederhana dan tidak memerlukan banyak pengecekan tambahan.

- Penggunaan Energi

Dengan gerakan berpola, bot cenderung lebih efisien dalam penggunaan energi meskipun tetap menghabiskan energi lebih banyak karena terus menembak satu target.

- Kompleksitas Waktu
O(1) untuk memilih musuh pertama, tetapi waktu pengambilan keputusan dalam setiap iterasi pertempuran tetap **O(1)** karena bot hanya fokus pada satu musuh yang terkunci.
- Kompleksitas Ruang
O(1) untuk penyimpanan, karena hanya musuh yang terkunci yang perlu dilacak.

Efektivitas

- Memaksimalkan Damage
Mengunci musuh hingga mati memberi keuntungan dalam mengenai musuh secara konsisten dan memastikan damage.
- Kelemahan
Bot ini kurang fleksibel dalam menghadapi musuh baru yang muncul atau situasi di mana target yang terkunci berubah posisi atau menghindar. Selain itu, jika bot menghadapi banyak musuh dalam jangkauan, bot ini bisa terjebak fokus pada satu musuh tanpa mempertimbangkan posisi musuh lain yang mungkin lebih berbahaya atau lebih mudah ditangani.

Bot 4: Menembak Musuh yang Pertama Kali Dilihat, Kemudian Dilakukan Scan Setelah Penembakan untuk Penggantian Target, dengan Firepower Adaptif dan Gerakan Acak (*Random Movement*)

Efisiensi

- Kecepatan Pengambilan Keputusan
Algoritma ini lebih kompleks dan mungkin memerlukan lebih banyak waktu untuk memindai target dan menyesuaikan firepower, tetapi dengan gerakan acak yang membuatnya lebih sulit diprediksi oleh musuh.
- Penggunaan Energi
Menggunakan *firepower* adaptif dan gerakan acak dapat mengurangi efisiensi energi, karena tidak ada pola yang konsisten dan bot bisa menghabiskan banyak energi untuk terus menyesuaikan firepower dan bergerak.
- Kompleksitas Waktu
O(1) untuk memilih musuh pertama, tetapi waktu pengambilan keputusan dalam setiap iterasi pertempuran tetap **O(1)** karena bot hanya fokus pada satu musuh yang terkunci.
- Kompleksitas Ruang
O(1) untuk penyimpanan, karena hanya musuh yang terkunci yang perlu dilacak.

Efektivitas

- Memaksimalkan Damage
Dengan firepower adaptif, bot dapat menyesuaikan tembakan untuk mengalahkan musuh dengan lebih efektif berdasarkan posisi dan energi musuh.
- Keunggulan

Gerakan acak membuat bot lebih sulit diprediksi, memberi keuntungan dalam menghindari tembakan musuh. Namun, ini juga bisa membuat bot tidak fokus pada target utama.

- Kelemahan
Bot ini bisa kehilangan fokus dalam pertempuran jika terlalu sering berganti target atau menghabiskan energi terlalu cepat dengan gerakan acak.
- Kompleksitas Waktu
O(n) untuk memilih target terbaik (karena perlu memindai beberapa musuh), dan **O(1)** untuk memodifikasi firepower atau gerakan. Secara keseluruhan, waktu pengambilan keputusan mungkin lebih tinggi karena banyak **komputasi adaptif**.
- Kompleksitas Ruang
O(n) untuk menyimpan status setiap musuh yang terlihat, serta beberapa parameter untuk pengaturan **firepower adaptif** dan **gerakan acak**.

3.6. Pemilihan Algoritma Greedy

Setelah melakukan analisis dari keempat bot, kami membuat tabel untuk membandingkan efektivitas dari tiap bot yang telah dirumuskan.

Kriteria	Bot 1	Bot 2	Bot 3	Bot 4
Kecepatan Pengambilan Keputusan	Cepat, sederhana	Sedikit lebih lambat (butuh pengecekan energi)	Cukup cepat, namun bot terkunci pada satu musuh	Lebih lambat karena perlu scan dan adaptasi firepower
Penggunaan Energi	Efisien, namun bisa kehabisan energi jika tidak dikendalikan	Lebih efisien dalam memilih musuh yang lemah	Penggunaan energi mungkin lebih tinggi karena fokus pada satu musuh	Penggunaan energi tinggi, terutama dengan gerakan acak dan firepower adaptif
Memaksimalkan Damage	Cukup maksimal dalam menyerang musuh terdekat	Potensi lebih tinggi, tapi kesulitan jika musuh terlalu jauh	Damage konsisten pada satu target, namun bisa kurang efektif jika musuh menghindar	Memaksimalkan damage, tetapi bisa menghabiskan energi terlalu cepat
Adaptasi terhadap Musuh	Kurang adaptif terhadap musuh yang menghindar	Kurang adaptif terhadap musuh yang menghindar atau bergerak cepat	Kurang fleksibel terhadap musuh lain yang muncul atau bergerak cepat	Sangat adaptif, namun terkadang bisa kehilangan fokus pada target
Fleksibilitas Gerakan	Gerakan berpola, mudah diprediksi	Gerakan berpola, mudah diprediksi	Gerakan berpola, mudah diprediksi	Gerakan acak memberikan fleksibilitas, namun kurang efektif dalam memfokuskan

				serangan
Kelemahan Umum	Kurang fleksibel dalam menghadapi musuh lebih kuat	Terlalu bergantung pada musuh dengan energi rendah, rentan jika jarak terlalu jauh	Kurang efisien dalam menghadapi banyak musuh	Kehilangan fokus pada target utama, menghabiskan energi terlalu cepat
Komplekitas Waktu	$O(n)$	$O(n)$	$O(1)$	$O(n)$
Kompleksitas Ruang	$O(1)$	$O(1)$	$O(1)$	$O(n)$

Dari perbandingan tersebut, **Bot 1** unggul di banyak aspek, berbeda tipis dengan **Bot 2**, terutama dalam **kecepatan pengambilan keputusan** (meskipun **Bot 3** lebih cepat) dan **efisiensi energi**. Dengan demikian, kami memutuskan untuk menggunakan **Bot 1** sebagai bot utama/*main bot* dan **Bot 2-4** sebagai bot alternatif/*alternative bots*.

BAB IV

IMPLEMENTASI DAN PENGUJIAN

4.1. Struktur Program

Proyek ini memiliki struktur direktori dan file sebagai berikut:

```
Tubes1_KepelesetPolisiTidur
├── doc/
└── src/
    ├── alternative_bots
    │   ├── KopelesetPolisiTidurAlt1
    │   │   ├── bin/
    │   │   ├── obj/
    │   │   ├── KopelesetPolisiTidurAlt1.cmd
    │   │   ├── KopelesetPolisiTidurAlt1.cs
    │   │   ├── KopelesetPolisiTidurAlt1.csproj
    │   │   ├── KopelesetPolisiTidurAlt1.json
    │   │   └── KopelesetPolisiTidurAlt1.sh
    │   ├── KopelesetPolisiTidurAlt2
    │   │   ├── bin/
    │   │   ├── obj/
    │   │   ├── KopelesetPolisiTidurAlt2.cmd
    │   │   ├── KopelesetPolisiTidurAlt2.cs
    │   │   ├── KopelesetPolisiTidurAlt2.csproj
    │   │   ├── KopelesetPolisiTidurAlt2.json
    │   │   └── KopelesetPolisiTidurAlt2.sh
    │   ├── KopelesetPolisiTidurAlt3
    │   │   ├── bin/
    │   │   ├── obj/
    │   │   ├── KopelesetPolisiTidurAlt3.cmd
    │   │   ├── KopelesetPolisiTidurAlt3.cs
    │   │   ├── KopelesetPolisiTidurAlt3.csproj
    │   │   ├── KopelesetPolisiTidurAlt3.json
    │   │   └── KopelesetPolisiTidurAlt3.sh
    └── main-bot
        ├── KopelesetPolisiTidur
        │   ├── bin/
        │   ├── obj/
        │   ├── KopelesetPolisiTidur.cmd
        │   ├── KopelesetPolisiTidur.cs
        │   ├── KopelesetPolisiTidur.csproj
        │   ├── KopelesetPolisiTidur.json
        │   └── KopelesetPolisiTidur.sh
    └── .gitignore/
    └── README.md
    └── Tubes1_KepelesetPolisiTidur.sln
```

Keterangan

- doc/: Berisi laporan tugas besar 1 strategi algoritma dalam bentuk .pdf
- src/: Berisi kode sumber program
 - a. alternative-bots/: Berisi folder dari 3 bot alternatif yang dikembangkan dengan greedy yang berbeda-beda.
Setiap bot alternatif memiliki folder masing-masing yang berisi:
 - 1) bin/: Berisi file hasil *build* dari bot setelah di-*compile*.
 - 2) obj/: Berisi file sementara selama proses *build*.
 - 3) KepolesetPolisiTidurAltX.cmd: Berisi *script batch* agar bot dapat dijalankan di Windows.
 - 4) KepolesetPolisiTidurAltX.cs: Berisi *source code* dalam bahasa C#.
 - 5) KepolesetPolisiTidurAltX.csproj: Berisi file proyek C# yang digunakan oleh VS Code untuk mengelola dependensi dan konfigurasi proyek
 - 6) KepolesetPolisiTidurAltX.json: Berisi konfigurasi untuk bot
 - 7) KepolesetPolisiTidurAltX.sh: Berisi *script* untuk menjalankan bot di sistem berbasis Unix/Linux
 - b. main-bots/: Berisi folder dari bot utama yang dikembangkan dengan strategi greedy terbaik.
Folder KepolesetPolisiTidur/ berisi:
 - 1) bin/: Berisi file hasil *build* dari bot setelah di-*compile*.
 - 2) obj/: Berisi file sementara selama proses *build*.
 - 3) KepolesetPolisiTidur.cmd: Berisi *script batch* agar bot dapat dijalankan di Windows.
 - 4) KepolesetPolisiTidur.cs: Berisi *source code* dalam bahasa C#.
 - 5) KepolesetPolisiTidur.csproj: Berisi file proyek C# yang digunakan oleh VS Code untuk mengelola dependensi dan konfigurasi proyek
 - 6) KepolesetPolisiTidur.json: Berisi konfigurasi untuk bot
 - 7) KepolesetPolisiTidur.sh: Berisi *script* untuk menjalankan bot di sistem berbasis Unix/Linux
- .gitignore/: Berisi daftar file atau folder yang akan diabaikan oleh Git, seperti bin/ dan obj/
- README.md : Berisi file readme yang mencakup deskripsi singkat, persyaratan, instalasi, kompilasi, cara menjalankan program dan nama author
- Tubes1_KepolesetPolisiTidur.sln:

4.2. Robocode 1

- a. Pseudocode

```

// Inisialisasi bot
BEGIN KepolesetPolisiTidur

    SET closestDistance = MaxValue // Menyimpan jarak ke target terdekat
    SET closestBotId = -1 // Menyimpan ID bot target
    SET targetX = 0, targetY = 0 // Menyimpan koordinat dari bot target
    SET rotateGunRight = TRUE // Menyimpan nilai boolean dari arah rotasi gun

    START BOT

    WHILE IsRunning DO
        IF closestBotId ≠ -1 THEN
            // Menghitung perbedaan sudut antara arah bot saat ini dengan arah ke target
            SET turnAngle = NormalizeRelativeAngle(BearingTo(targetX, targetY))
            TurnRight(turnAngle) // Arahkan bot ke bot musuh

            // Atur move berdasarkan jarak ke target
            SET distance = DistanceTo(targetX, targetY)
            IF distance > 150 THEN
                Forward(100)
            ELSE IF distance < 50 THEN
                Back(100)
            ELSE
                Forward(50)
            END IF

            // Melakukan pemindaian bot musuh
            IF rotateGunRight THEN
                TurnGunRight(360)
            ELSE
                TurnGunLeft(360)
            END IF
            rotateGunRight = NOT rotateGunRight

        ELSE
            IF rotateGunRight THEN
                TurnGunRight(360)
            ELSE
                TurnGunLeft(360)
            END IF
            rotateGunRight = NOT rotateGunRight
        END IF

    END WHILE

END KepolesetPolisiTidur

// Event ketika mendeteksi bot musuh
ON EVENT OnScannedBot(evt)
    SET distance = DistanceTo(evt.X, evt.Y)

    // Jika ada bot musuh lebih dekat, jadikan target utama
    IF distance < closestDistance THEN
        closestDistance = distance
        closestBotId = evt.ScannedBotId
        targetX = evt.X

```

```

targetY = evt.Y
END IF

// Jika target ini adalah yang terdekat, tembak dengan kekuatan maksimal
IF evt.ScannedBotId == closestBotId THEN
    Fire(3)
END IF
END EVENT

// Event ketika menabrak dinding
ON EVENT OnHitWall(evt)
    Back(50)
    TurnRight(90)
END EVENT

// Event ketika bertabrakan dengan bot lain
ON EVENT OnHitBot(evt)
    Back(100)
    TurnRight(45)
END EVENT

// Event ketika bot musuh yang ditargetkan mati
ON EVENT OnBotDeath(evt)
    IF evt.VictimId == closestBotId THEN
        closestDistance = infinity // Reset agar mencari target baru
        closestBotId = -1 // Hapus target saat ini
    END IF
END EVENT

```

b. Struktur Data

Bot ini menggunakan struktur data bawaan bahasa pemrograman (primitif) untuk menyimpan informasi *battle*:

- closestDistance → Menyimpan jarak dengan bot target terdekat.
- closestBotId → Menyimpan ID dari bot target terdekat.
- targetX, targetY → Menyimpan koordinat bot target
- rotateGunRight → Menyimpan boolean dari arah rotasi *gun*.

c. Fungsi dan Prosedur yang Digunakan dalam Solusi Greedy

Pendekatan greedy dalam bot ini dilakukan dengan selalu memilih target dengan jarak terdekat secara lokal tanpa mempertimbangkan kondisi jangka panjang. Fungsi-fungsi yang digunakan dikelompokkan berdasarkan proses utama dalam strategi greedy ini:

1. Pemindaian & Pemilihan Target (Greedy Selection)

Bot selalu mencari target terbaik di setiap iterasi pemindaian dengan membandingkan jarak.

- TurnGunRight() / TurnGunLeft(): Memutar turret untuk mencari target baru.

- OnScannedBot(evt): Event scanning yang memindai musuh, menghitung jarak, dan memperbarui target jika lebih dekat dari sebelumnya.
- DistanceTo(x, y): Mengukur jarak ke target.
- BearingTo(x, y): Menentukan arah ke target agar bot bisa berputar ke sana.

2. Menyerang Target (Greedy Attack)

Jika sudah ada target terbaik, bot langsung menembak.

- Fire(3): Menembak dengan kekuatan penuh jika bot yang ter-scan merupakan target saat ini.
- NormalizeRelativeAngle(): Menyesuaikan sudut agar bot bisa mengarahkan turret dengan lebih akurat.

3. Pergerakan Bot (Greedy Movement)

Bot menyesuaikan jarak ke target berdasarkan kondisi saat ini tanpa mempertimbangkan pergerakan jangka panjang.

- TurnRight(angle): Memutar bot agar mendekati target secara optimal.
- Forward(distance) / Back(distance): Bergerak maju atau mundur berdasarkan jarak ke target.

4. Penanganan Tabrakan & Target Hilang (Greedy Recovery)

Jika terjadi tabrakan atau target mati, bot segera menangani dengan cara berikut:

- OnHitWall(evt): Jika menabrak dinding, bot langsung mundur dan belok untuk keluar dari posisi buruk.
- OnHitBot(evt): Jika bertabrakan dengan musuh, bot mundur dan berputar agar tidak stuck..
- OnBotDeath(evt): Jika target utama mati, bot langsung mencari target baru dengan reset skor.

4.3. Robocode 2

a. Pseudocode

```
// Inisialisasi bot
BEGIN KepolesetPolisiTidurAlt1

SET bestScore = infinity // Simpan skor target terbaik (HP + jarak)
SET bestTargetId = -1 // ID target yang dipilih
SET targetX = 0, targetY = 0 // Koordinat target yang terdeteksi
SET rotateGunRight = TRUE // Arah rotasi gun untuk scanning

START BOT

WHILE IsRunning DO
  IF bestTargetId ≠ -1 THEN // Jika ada target terpilih
    SET turnAngle = NormalizeRelativeAngle(BearingTo(targetX, targetY))
    TurnRight(turnAngle) // Arahkan bot ke target

    // Menyesuaikan pergerakan berdasarkan jarak ke target
    SET distance = DistanceTo(targetX, targetY)
```

```

IF distance > 150 THEN
    Forward(100) // Maju cepat jika terlalu jauh
ELSE IF distance < 50 THEN
    Back(100) // Mundur jika terlalu dekat
ELSE
    Forward(50) // Maju dengan kecepatan sedang jika dalam jangkauan ideal

// Rotasi senjata 360 derajat untuk terus melakukan scanning
IF rotateGunRight THEN
    TurnGunRight(360)
ELSE
    TurnGunLeft(360)
END IF
rotateGunRight = NOT rotateGunRight // Ubah arah rotasi

ELSE // Jika belum ada target
// Scan area dengan memutar turret 360 derajat
IF rotateGunRight THEN
    TurnGunRight(360)
ELSE
    TurnGunLeft(360)
END IF
rotateGunRight = NOT rotateGunRight
END IF
END WHILE

END KepesetPolisiTidurAlt1

// Event ketika mendeteksi bot musuh
ON EVENT OnScannedBot(evt)
SET distance = DistanceTo(evt.X, evt.Y)
SET enemyHP = evt.Energy
SET score = enemyHP + distance // Hitung skor target

// Jika musuh ini memiliki skor lebih baik (lebih kecil), jadikan target utama
IF score < bestScore THEN
    bestScore = score
    bestTargetId = evt.ScannedBotId
    targetX = evt.X
    targetY = evt.Y
END IF

// Jika target ini adalah target terbaik yang sudah dipilih, tembak dengan kekuatan penuh
IF evt.ScannedBotId == bestTargetId THEN
    Fire(3)
END IF
END EVENT

// Event ketika menabrak dinding
ON EVENT OnHitWall(evt)
Back(50) // Mundur untuk menghindari tabrakan
TurnRight(90) // Belok 90 derajat untuk keluar dari sudut
END EVENT

// Event ketika bertabrakan dengan bot lain

```

```

ON EVENT OnHitBot(evt)
    Back(100) // Mundur untuk menghindari stuck
    TurnRight(45) // Putar sedikit agar tidak menabrak kembali
END EVENT

// Event ketika bot musuh yang ditargetkan mati
ON EVENT OnBotDeath(evt)
    IF evt.VictimId == bestTargetId THEN
        bestScore = infinity // Reset skor agar bisa mencari target baru
        bestTargetId = -1 // Hapus target
    END IF
END EVENT

```

b. Struktur Data

Bot menggunakan struktur data bawaan bahasa pemrograman (primitif) untuk menyimpan informasi *battle*:

- bestScore (Float) → Skor target terbaik (terendah) berdasarkan HP + jarak (default: ∞).
- bestTargetId (Integer) → ID target yang dipilih (default: -1 jika belum ada target).
- targetX, targetY (Float) → Koordinat target yang terdeteksi.
- rotateGunRight (Boolean) → Menentukan arah rotasi turret saat scanning.
- distance (Float) → Jarak bot ke target yang sedang diproses.
- enemyHP (Float) → HP target yang sedang diproses.
- score (Float) → Skor target yang dihitung untuk memilih target terbaik.

c. Fungsi dan Prosedur yang Digunakan dalam Solusi Greedy

Pendekatan greedy dalam bot ini dilakukan dengan selalu memilih target dengan skor terbaik (terdekat & HP terendah) secara lokal tanpa mempertimbangkan kondisi jangka panjang. Fungsi-fungsi yang digunakan dikelompokkan berdasarkan proses utama dalam strategi greedy ini:

1. Pemindaian & Pemilihan Target (Greedy Selection)

Bot selalu mencari target terbaik di setiap iterasi pemindaian dengan membandingkan skor (HP + jarak).

- TurnGunRight() / TurnGunLeft(): Memutar turret untuk mencari target baru.
- OnScannedBot(evt): Event scanning yang memindai musuh, menghitung skor, dan memperbarui target jika lebih baik dari sebelumnya.
- DistanceTo(x, y): Mengukur jarak ke target untuk perhitungan skor.
- BearingTo(x, y): Menentukan arah ke target agar bot bisa berputar ke sana.

2. Menyerang Target (Greedy Attack)

Jika sudah ada target terbaik, bot langsung menembak.

- Fire(3): Menembak dengan kekuatan penuh jika bot yang ter-scan merupakan target saat ini.
- NormalizeRelativeAngle(): Menyesuaikan sudut agar bot bisa mengarahkan turret dengan lebih akurat.

3. Pergerakan Bot (Greedy Movement)

Bot menyesuaikan jarak ke target berdasarkan kondisi saat ini tanpa mempertimbangkan pergerakan jangka panjang.

- TurnRight(angle): Memutar bot agar mendekati target secara optimal.
- Forward(distance) / Back(distance): Bergerak maju atau mundur berdasarkan jarak ke target.

4. Penanganan Tabrakan & Target Hilang (Greedy Recovery)

Jika terjadi tabrakan atau target mati, bot segera menangani dengan cara berikut:

- OnHitWall(evt): Jika menabrak dinding, bot langsung mundur dan belok untuk keluar dari posisi buruk.
- OnHitBot(evt): Jika bertabrakan dengan musuh, bot mundur dan berputar agar tidak stuck..
- OnBotDeath(evt): Jika target utama mati, bot langsung mencari target baru dengan reset skor.

4.4. Robocode 3

a. Pseudocode

```
// Inisialisasi bot
BEGIN KepolesetPolisiTidurAlt2

SET targetId = null // Menyimpan ID bot target
SET targetX = 0, targetY = 0 // Menyimpan koordinat dari bot target
SET rotateGunRight = TRUE // Menyimpan nilai boolean dari arah rotasi gun

START BOT

WHILE IsRunning DO
    IF targetId ≠ null THEN // Jika sudah ada target yang dikunci
        SET turnAngle = NormalizeRelativeAngle(BearingTo(targetX, targetY))
        TurnRight(turnAngle) // Arahkan bot ke target

        // // Atur move berdasarkan jarak ke target
        SET distance = DistanceTo(targetX, targetY)
        IF distance > 150 THEN
            Forward(100)
        ELSE IF distance < 50 THEN
            Back(100)
        ELSE
            Forward(50)
        END IF
    END IF
```

```

// Melakukan pemindaian
IF rotateGunRight THEN
    TurnGunRight(360)
ELSE
    TurnGunLeft(360)
END IF
rotateGunRight = NOT rotateGunRight

END WHILE

END KepesetPolisiTidurAlt2

// Event ketika mendeteksi bot musuh
ON EVENT OnScannedBot(evt)
    // Jika belum ada target, ambil bot pertama yang terdeteksi
    IF targetId = null THEN
        targetId = evt.ScannedBotId
        targetX = evt.X
        targetY = evt.Y
    END IF

    // Jika yang di scan adalah target, update posisi & tembak
    IF evt.ScannedBotId == targetId THEN
        targetX = evt.X
        targetY = evt.Y
        Fire(3)
    END IF
END EVENT

// Event ketika target mati
ON EVENT OnBotDeath(evt)
    IF evt.VictimId == targetId THEN
        targetId = null // Reset target agar bisa mencari yang baru
    END IF
END EVENT

// Event ketika menabrak dinding
ON EVENT OnHitWall(evt)
    Back(50)
    TurnRight(90)
END EVENT

// Event ketika bertabrakan dengan bot lain
ON EVENT OnHitBot(evt)
    Back(100)
    TurnRight(45)
END EVENT

```

b. Struktur Data

Bot Menggunakan struktur data bawaan bahasa pemrograman (primitif) untuk menyimpan informasi *battle*:

- targetId → Menyimpan ID dari bot musuh yang sedang ditargetkan. (Default = null)

- targetX, targetY → Menyimpan koordinat dari bot target yang sedang ditargetkan.
 - rotateGunRight → Menyimpan nilai boolean dari arah rotasi *gun*.
- c. Fungsi dan Prosedur yang Digunakan dalam Solusi Greedy

Pendekatan greedy dalam bot ini dilakukan dengan pendekatan *first-scanned* target, yaitu memilih dan mengunci musuh pertama yang terdeteksi, lalu mempertahankan target tersebut sampai musuh itu mati. Fungsi dan prosedur yang digunakan adalah sebagai berikut:

- OnScannedBot(evt): Digunakan untuk melakukan memindai dan mendeteksi musuh. Jika target belum ada, nilai dari targetId = null. Jika terdapat target yang terdeteksi, bot akan memperbarui posisinya dan menembak dengan kekuatan maksimal FIRE(3).
- OnBotDeath(evt): Digunakan untuk me-reset target jika sudah mati.
- OnHitWall(evt): Digunakan untuk melakukan penyesuaian gerakan ketika menabrak dinding.
- OnHitBot(evt): Digunakan untuk melakukan penyesuaian gerakan ketika menabrak bot lain.

4.5. Robocode 4

a. Pseudocode

```

// Inisialisasi bot
BEGIN KepesetPolisiTidurAlt3

SET lastEnemyAngle = 0 // Menyimpan sudut relatif musuh yang terdeteksi
SET lastEnemyDistance = 0 // Menyimpan jarak relatif musuh yang terdeteksi

START BOT

WHILE IsRunning DO
    CALL MoveBot() // Menggerakkan bot dengan pola acak
    TurnGunRight(360) // Melakukan pemindaian
END WHILE

END KepesetPolisiTidurAlt3

// Fungsi untuk menggerakkan bot dengan pola acak
FUNCTION MoveBot()
    Forward(RANDOM(50, 151)) // Maju dengan jarak acak

    IF RANDOM(0, 1) == 0 THEN
        TurnRight(RANDOM(15, 91)) // Putar ke kanan secara acak
    ELSE
        TurnLeft(RANDOM(15, 91)) // Putar ke kiri secara acak
    END IF

```

```

Back(RANDOM(50, 151)) // Mundur dengan jarak acak

IF RANDOM(0, 1) == 0 THEN
    TurnRight(RANDOM(15, 91)) // Putar ke kanan secara acak
ELSE
    TurnLeft(RANDOM(15, 91)) // Putar ke kiri secara acak
END IF
END FUNCTION

// Event ketika mendeteksi bot musuh
ON EVENT OnScannedBot(evt)
lastEnemyAngle = NormalizeRelativeAngle(evt.Direction - GunDirection)
lastEnemyDistance = DistanceTo(evt.X, evt.Y)

// Menentukan kekuatan tembakan berdasarkan jarak dan energi bot
IF Energy < 20 THEN
    SET firePower = 1
ELSE IF evt.Speed == 0 OR DistanceTo(evt.X, evt.Y) < 200 THEN
    SET firePower = 3
ELSE IF DistanceTo(evt.X, evt.Y) < 500 THEN
    SET firePower = 2
ELSE
    SET firePower = 1
ENDIF

Fire(firePower) // Tembak dengan kekuatan yang telah disesuaikan

// Arahkan gun ke musuh
TurnGunRight(NormalizeRelativeAngle(evt.Direction - GunDirection))
END EVENT

// Event ketika terkena peluru musuh
ON EVENT OnHitByBullet(evt)
Back(150) // Mundur untuk menghindar
IF RANDOM(0, 1) == 0 THEN
    TurnRight(45)
ELSE
    TurnLeft(45)
ENDIF
END EVENT

// Event ketika menabrak dinding
ON EVENT OnHitWall(evt)
Back(150)
TurnRight(90)
END EVENT

// Event ketika menabrak bot lain
ON EVENT OnHitBot(evt)
Back(150)
TurnRight(45)
END EVENT

```

b. Struktur Data

Bot menggunakan struktur data bawaan bahasa pemrograman (primitif) untuk menyimpan informasi *battle*:

- lastEnemyAngle (double) → Menyimpan sudut relatif musuh terhadap *gun* dari bot.
- lastEnemyDistance (double) → Menyimpan jarak dari bot musuh.
- firePower (double) → Menyimpan nilai kekuatan dari *gun* ketika akan ditembak (FIRE).

c. Fungsi dan Prosedur yang Digunakan dalam Solusi Greedy

Pendekatan greedy dalam bot ini dilakukan dengan menyesuaikan kekuatan tembakan (FIRE) berdasarkan energi yang dimiliki oleh bot dan juga jarak dengan bot musuh. Fungsi dan prosedur yang digunakan dikelompokkan berdasarkan proses utama dalam strategi greedy ini:

1. OnScannedBot(evt): Bot melakukan pemindaian dan menembak musuh setiap kali terdeteksi. Kekuatan tembakan yang dihasilkan dihitung berdasarkan kondisi dari energi bot dan jarak terhadap musuh. Fokus dari bot ini hanya lah menyerang musuh yang ada saat ini. Dampak dari strategi ini adalah dapat dengan efektif memberikan damage dan menghemat energi dari bot. Akan tetapi, kekurangannya adalah musuh yang ditembak adalah musuh yang pertama kali dideteksi, bukan musuh dengan energi terendah atau yang memiliki jarak terdekat.
2. MoveBot(): Bot akan bergerak secara acak, baik maju atau mundur, maupun belok kanan atau belok kiri. Bot berfokus untuk membuat pergerak yang sulit ditebak. Akan tetapi, dampak negatifnya adalah bisa saja pergerakan yang dibuat oleh robot tidak aman atau tidak strategis
3. OnHitByBullet(evt): Bot akan mundur dan mengubah arah ketika terkena tembakan. Bot berfokus untuk menghindari serangan langsung secara terus-menerus.
4. OnHitWall(evt) dan OnHitBot(evt): Bot akan mengubah arah gerak ketika menabrak tembok atau menabrak bot lain.

4.6. Pengujian Semua Robocode

Gambar di bawah merupakan hasil pengujian bot sebanyak empat kali.

Results for 10 rounds													
Rank	Name	Total Score	Survival	Surv. Bonus	Bullet Dmg.	Bullet Bonus	Ram Dmg.	Ram Bonus	1sts	2nds	3rds	4ths	5ths
1	KepelasetPolisiTidur 1.0	967	350	30	528	57	1	0	1	1	1	2	
2	KepelasetPolisiTidurAlt3 1.0	921	450	30	414	24	2	0	1	3	0		
3	KepelasetPolisiTidurAlt2 1.0	882	300	30	496	51	5	0	2	0	1		
4	KepelasetPolisiTidurAlt 1.0	845	350	30	416	48	1	0	1	1	1	2	

Results for 10 rounds

Rank	Name	Total Score	Survival	Surv. Bonus	Bullet Dmg.	Bullet Bonus	Ram Dmg.	Ram Bonus	1sts	2nds	3rds
1	KepolesetPolisiTidurAlt1 1.0	1554	650	90	688	118	7	0	3	2	1
2	KepolesetPolisiTidur 1.0	1164	400	30	672	57	5	0	2	2	0
3	KepolesetPolisiTidurAlt3 1.0	683	400	30	240	8	5	0	1	1	3
4	KepolesetPolisiTidurAlt2 1.0	630	200	0	384	44	1	0	0	2	1

Results for 10 rounds

Rank	Name	Total Score	Survival	Surv. Bonus	Bullet Dmg.	Bullet Bonus	Ram Dmg.	Ram Bonus	1sts	2nds	3rds
1	KepolesetPolisiTidurAlt2 1.0	1318	550	90	608	70	0	0	4	0	2
2	KepolesetPolisiTidur 1.0	1037	400	30	560	41	5	0	2	2	1
3	KepolesetPolisiTidurAlt1 1.0	967	400	30	496	60	0	0	2	2	1
4	KepolesetPolisiTidurAlt3 1.0	361	150	0	198	6	7	0	0	1	1

Results for 10 rounds

Rank	Name	Total Score	Survival	Surv. Bonus	Bullet Dmg.	Bullet Bonus	Ram Dmg.	Ram Bonus	1sts	2nds	3rds
1	KepolesetPolisiTidurAlt1 1.0	1521	650	90	688	89	4	0	3	1	2
2	KepolesetPolisiTidur 1.0	1464	550	30	816	60	7	0	2	3	1
3	KepolesetPolisiTidurAlt2 1.0	1204	450	60	624	60	10	0	2	1	1
4	KepolesetPolisiTidurAlt3 1.0	630	300	0	312	6	12	0	0	2	3

Untuk hasil lebih lengkap, dapat dilihat pada bagian lampiran, ada *video testing* dan juga tampilan pada Youtube.

4.7. Analisis Pengujian

Untuk analisis pengujian, sistem penilaian digunakan untuk menentukan ranking setiap bot berdasarkan hasil battle. Poin diberikan dengan skema sebagai berikut: bot yang berada di posisi pertama mendapatkan 4 poin, posisi kedua mendapatkan 3 poin, posisi ketiga mendapatkan 2 poin, dan posisi keempat mendapatkan 1 poin. Setelah beberapa battle dilakukan, total poin dari setiap bot dijumlahkan untuk menentukan performa keseluruhan dari masing-masing strategi.

Hasil Akumulasi Poin

- Bot 1 (KepolesetPolisiTidur): 13 poin
- Bot 2 (KepolesetPolisiTidurAlt1): 11 poin
- Bot 3 (KepolesetPolisiTidurAlt2): 9 poin
- Bot 4 (KepolesetPolisiTidurAlt3): 7 poin

Analisis Hasil

1. Bot 1 (KepolesetPolisiTidur) – 13 Poin

- Strategi: Greedy berbasis jarak (memilih target terdekat untuk diserang).
- Performa terbaik dalam battle ini.
- Dengan hanya mempertimbangkan jarak, bot ini dapat lebih fokus dalam menyerang tanpa harus melakukan perhitungan tambahan yang dapat mengganggu eksekusi strategi.
- Keunggulan bot ini terletak pada akurasi tinggi karena target selalu berada dalam jarak tembak optimal, sehingga lebih mudah menghabisi lawan.

2. Bot 2 (KepalaPolisiTidurAlt1) – 11 Poin

- Strategi: Greedy berbasis kombinasi HP (Energy) dan jarak (memilih target berdasarkan keseimbangan HP dan jarak).
- Performa cukup baik, tetapi kalah dari Bot 1.
- Kekalahan ini kemungkinan disebabkan oleh kompleksitas strategi yang lebih tinggi. Dengan mempertimbangkan dua faktor sekaligus (HP dan jarak), bot ini harus melakukan lebih banyak perhitungan sebelum menembak. Hal ini dapat menyebabkan bot kehilangan fokus dan sedikit memperlambat eksekusi serangan, sehingga efektivitasnya dalam battle berkurang dibandingkan dengan Bot 1 yang lebih sederhana namun lebih terfokus.

3. Bot 3 (KepalaPolisiTidurAlt2) – 9 Poin

- Strategi: *First-scanned target* (mengunci target pertama yang terdeteksi dan terus menyerangnya sampai target mati).
- Kelemahan utama dari strategi ini adalah bot bisa saja memilih target yang berada jauh, sehingga tembakan menjadi kurang akurat dan peluang terkena lebih rendah.

4. Bot 4 (KepalaPolisiTidurAlt3) – 7 Poin

- Strategi: Energy Optimization Firing dan Random Move (menyesuaikan kekuatan tembakan berdasarkan energi serta bergerak secara acak).
- Meskipun strategi penghematan energi bisa efektif dalam jangka panjang, dalam battle skala kecil ini, sering kali yang lebih penting adalah akurasi tembakan dalam menyerang.
- Pergerakan acak yang diterapkan juga bisa menjadi pedang bermata dua—meskipun membuat bot lebih sulit ditembak, namun juga dapat menyebabkan bot berada dalam posisi yang tidak menguntungkan atau kehilangan posisi strategis dalam menembak.

BAB V

KESIMPULAN DAN SARAN

Kesimpulan

Dalam tugas besar ini, telah berhasil dibuat empat bot, yaitu satu bot utama dan tiga bot alternatif yang dipilih dari berbagai alternatif yang telah diuji. Setiap bot dirancang dengan pendekatan greedy dan heuristik yang berbeda. Pengujian dilakukan untuk mengevaluasi efektivitas strategi yang diterapkan, dan hasilnya menunjukkan bahwa solusi greedy dapat memberikan kinerja yang cukup kompetitif dalam pertarungan, meskipun memiliki keterbatasan dalam menghadapi strategi yang lebih kompleks.

Dari empat bot yang telah diuji dengan heuristik yang berbeda, bot yang memiliki algoritma kompleks belum tentu merupakan bot yang terbaik. Kemudian juga, masing-masing pendekatan *greedy* memberikan hasil yang sub-optimum; dalam artian kondisi yang berbeda menghasilkan luaran kemenangan yang berbeda juga. Hal ini telah sesuai dengan prinsip algoritma greedy, yakni kita berharap hasilnya optimum dengan mengambil optimum lokal, walaupun belum tentu mengantarkan kita ke optimum global.

Saran

Untuk pengembangan lebih lanjut, pendekatan greedy dapat diperluas dengan variasi lainnya, seperti menggabungkan beberapa heuristik greedy untuk meningkatkan fleksibilitas dalam pengambilan keputusan. Selain itu, pengujian lebih dalam terhadap skenario di mana greedy kurang optimal dapat membantu merancang strategi yang lebih efektif dalam berbagai kondisi pertarungan.

LAMPIRAN

- Tautan Repository:
https://github.com/wiwekaputra/Tubes1_KepelesetPolisiTidur
- Tautan Repository Video:
<https://youtu.be/gvH3zpNUMfI>
- Drive Terpusat:
▣ Tubes1
- Tabel Pengecekan Fitur:

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.	✓	

DAFTAR PUSTAKA

[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)

[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)

[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)