LAPORAN TUGAS BESAR 1

IF2211 Strategi Algoritma

Semester II tahun 2024/2025

Pemanfaatan Algoritma Greedy dalam pembuatan bot permainan Robocode Tank Royale

Dipersiapkan oleh:

M. Rayhan Farrukh 13523035

Hanif Kalyana Aditya 13523041

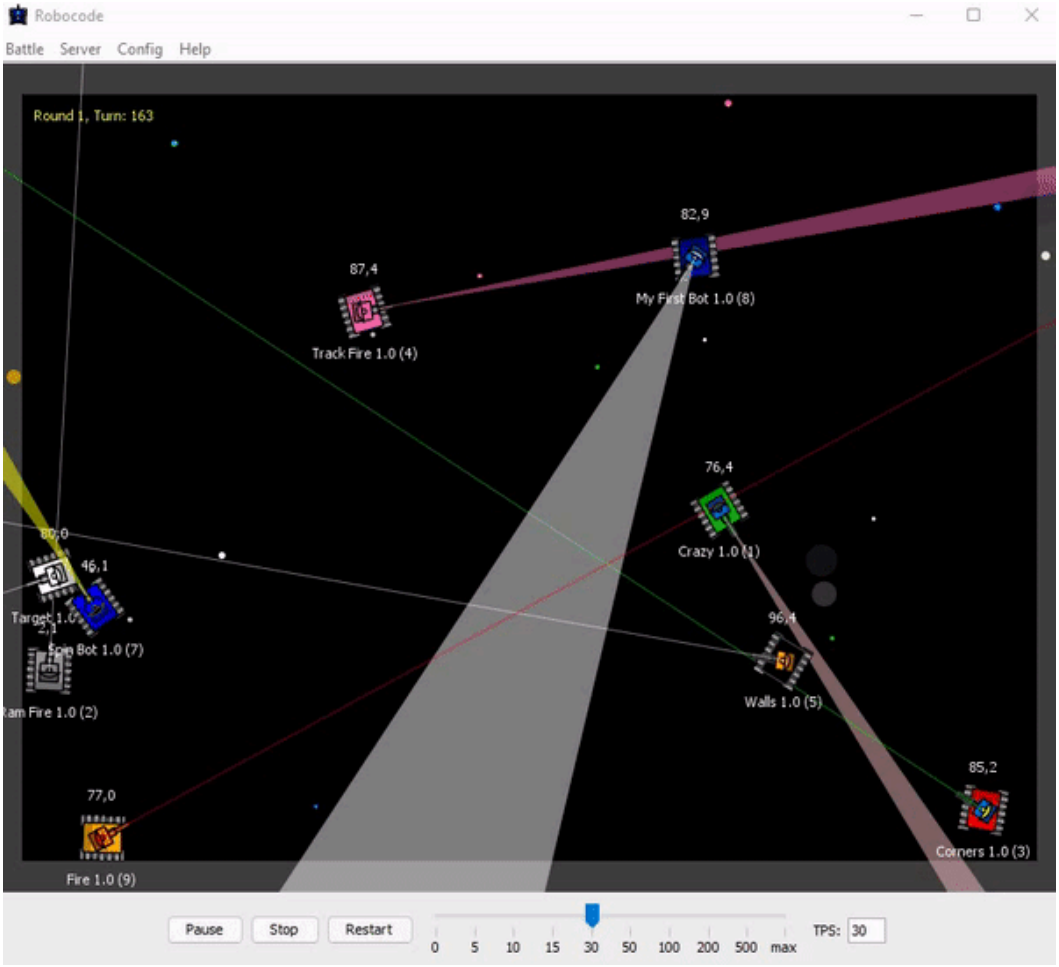
Athian Nugraha Muarajuang 13523106

Sekolah Teknik Elektro dan Informatika - Institut Teknologi Bandung

Jl. Ganesha 10, Bandung 40132

Daftar Isi

Bab 1: Spesifikasi Tugas Besar



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 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.

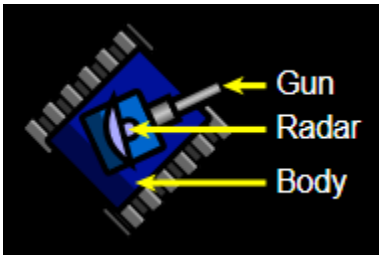
6. Tabrakan

   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 2 Anatomi 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 3 Radar lebar

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.

Gambar 4 Radar Kecil

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.

Untuk informasi lebih lengkap, silahkan buka dokumentasi Tank Royale pada link berikut.

Bab 2: Landasan Teori

2.1. Algoritma Greedy

Algoritma Greedy adalah sekumpulan algoritma dimana pengambilan keputusan dibuat berdasarkan pilihan optimal pada saat pengambilan, tanpa memikirkan hasil akhir yang akan didapat. Pilihan terbaik pada saat pengambilan disebut pilihan optimal lokal. Pilihan optimal lokal diambil dengan harapan akan membawa pilihan optimal global.

Algoritma Greedy seringkali menghasilkan proses dengan kompleksitas waktu O(n), namun seringkali membutuhkan pengurutan agar dapat melakukan algoritma dengan baik. Karena algoritma greedy hanya mengambil pilihan optimal lokal, algoritma ini tidak menjamin solusi optimal untuk semua jenis permasalahan. Meskipun begitu, algoritma greedy tetap berguna jika ingin melakukan aproksimasi solusi optimal.

2.2. Robocode

2.2.1. Overview

Robocode adalah sebuah permainan pemrograman di mana pemain memprogram robot-robot tank agar melakukan perintah-perintah untuk bertempur di dalam sebuah arena. Setiap bot dikendalikan menggunakan kode program, dan tujuan permainan adalah mengembangkan kode program dengan algoritma yang optimal untuk mendapat perolehan skor tertinggi diantara robot-robot tank musuh.

2.2.2. Cara Kerja Bot

Bot pada Robocode bekerja dengan menerima informasi lingkungan sekitarnya dan memproses informasi tersebut melalui API Robocode. API menyediakan *method-method* sebagai cara Bot menerima dan memproses informasi serta melaksanakan aksi sesuai keinginan pemain/*programmer*. Setiap aksi akan berkontribusi pada skor yang diperoleh Bot, dan pemilihan aksi-aksi tersebut untuk mendapat perolehan skor maksimum adalah tujuan utama permainan.

2.2.3. Cara Implementasi Algoritma Greedy

Algoritma Greedy diimplementasikan sebagai cara atau strategi Robot untuk memilih keputusan apa yang akan dilakukan setiap langkahnya berdasarkan perhitungan agar mendapat hasil yang optimal. Hasil yang ingin dioptimalkan disini adalah perolehan skor. Setiap strategi memiliki *heuristic* tertentu untuk menilai opsi mana yang paling menguntungkan pada suatu saat tertentu. Beberapa contoh strategi *greedy* pada robocode:

- Menembak musuh terdekat
- Bergerak menjauh dari robot lain

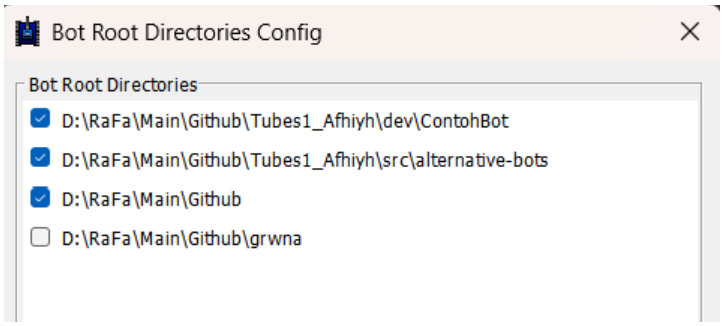
- Mengejar musuh dengan energi terendah

Dengan menerapkan strategi greedy yang sesuai, robot dapat membuat keputusan terbaik untuk memaksimalkan perolehan skor sekaligus peluang kemenangan untuk setiap pertempuran.

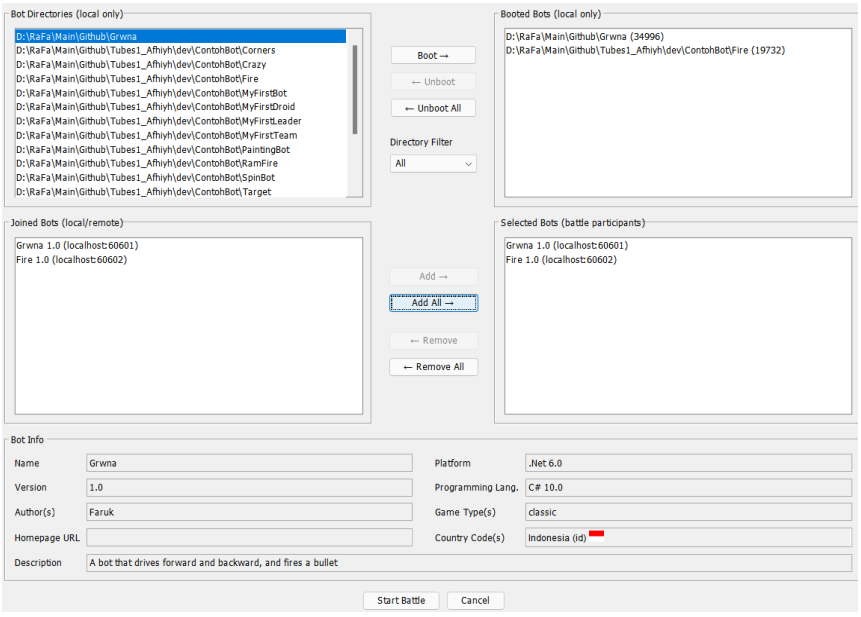
2.2.4. Cara Menjalankan Bot

Langkah-langkah menjalankan bot:

1. *Download* dan *install environment* dan aplikasi yang dibutuhkan (.NET/C# SDK, JDK, GUI Robocode)
2. Tulis kode program bot pada bahasa yang didukung seperti Java atau C#.
3. Buka GUI Robocode, lalu masukkan *directory* dimana kode robot disimpan sebagai salah satu *directory* sumber robot.



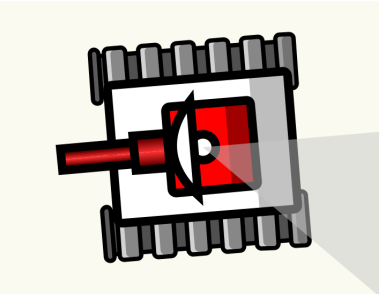
4. Tekan menu *Start Battle*, lalu *boot* robot-robot yang diinginkan
5. Tambahkan robot-robot yang telah di-*boot* sebagai robot yang ikut pertarungan



6. Tekan tombol *Start Battle* dan amati pertempuran para robot!

Bab 3: Aplikasi Strategi Greedy

3.1 GrwnaBot: Greedy by Bullet Damage Bonus



3.1.1. Overview

Pada bot alternatif ini, keputusan bot dibuat dengan memaksimalkan *output damage* dari peluru, terutama pada *Bullet Damage Bonus*, yaitu tambahan poin yang didapat ketika bot berhasil mendapatkan *kill* atau berhasil menghancurkan salah satu bot lawan. Optimalisasi ini dilakukan tanpa memikirkan atau memperhitungkan *defense* dari Bot, sehingga benar-benar memaksimalkan *output* dari serangan. Pendekatan yang agresif ini sangat cocok untuk situasi *melee* atau *free for all* karena sangat banyak robot dalam satu arena akan menyebabkan kesulitan untuk menghindari serangan robot lain.

3.1.2. Elemen-elemen Greedy

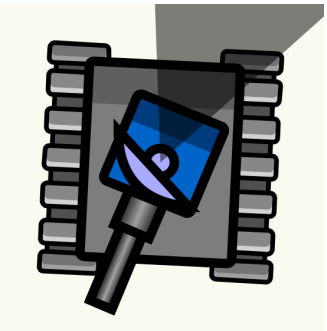
- Himpunan Kandidat: Musuh-musuh yang terdeteksi radar
- Himpunan Solusi: Musuh pertama yang terdeteksi
- Fungsi Solusi: Berhasil menembak dan mengenai musuh
- Fungsi Seleksi: Pilih musuh yang terdeteksi saat ini (agar cepat bertindak)
- Fungsi Kelayakan: Tidak ada
- Fungsi Objektif: Memaksimalkan *output damage* peluru

3.1.3. Evaluasi Efisiensi dan Efektivitas

Walaupun mengabaikan *defense* tampak seperti strategi yang tidak bagus, namun pada Robocode, hal terpenting untuk menjadi juara adalah memperoleh skor tertinggi. Ini berarti tank tidak perlu menjadi tank terakhir yang bertahan hidup, dalam kata lain, tidak perlu menjadi peringkat satu pada suatu ronde. Strategi ini efisien karena berfokus langsung pada tindakan yang memperoleh skor tanpa harus membuang waktu dengan bertahan atau menghindar. Selain itu, strategi ini

dinilai sangat efektif bahkan untuk mendapat *Survival Bonus* karena penggunaan *defense by offence*, dimana dengan membunuh bot lain, bot ini juga akan mendapat *Survival Bonus*.

3.2 Kabur: Greedy by Survival and Last Survival



3.2.1 Overview

Bot ini bekerja dengan cara menghindari musuh dan berusaha bertahan hidup selama mungkin. Pendekatan yang digunakan adalah menggunakan *anti-gravity*, atau gaya tolak-menolak (repulsive force), di mana setiap musuh yang terdeteksi akan "memberikan" gaya menjauh terhadap bot. Semakin dekat atau semakin tinggi energi musuh, maka gaya tolak yang dihasilkan akan semakin besar. Dengan strategi ini, bot secara konsisten menjaga jarak dari musuh, memaksimalkan peluang untuk bertahan hidup selama mungkin

3.2.2. Elemen-elemen Greedy

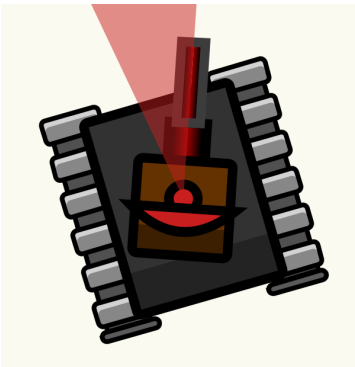
- Himpunan Kandidat: Semua musuh yang terdeteksi oleh radar yang akan memberikan arah bahaya yang akan dihindari
- Himpunan Solusi: Musuh yang dianggap paling mengancam saat ini
- Fungsi Solusi: Bot berhasil menentukan arah gerak yang menjauh dari musuh dan menghindari posisi berbahaya
- Fungsi Seleksi: Pilih nilai ancaman tertinggi sebagai solusi yang dipilih.
- satu musuh dari himpunan kandidat dengan Fungsi Kelayakan: Tidak ada
- Fungsi Objektif: Meminimumkan *damage* dan memaksimalkan waktu bertahan hidup

3.2.3. Evaluasi Efisiensi dan Efektivitas

Strategi bertahan hidup yang digunakan sangat efisien karena hanya mengandalkan perhitungan *anti-gravity* untuk menjauhi musuh, tanpa perlu logika tembak-menembak yang kompleks. Strategi ini juga dinilai efektif karena mampu menghindari kerusakan dan bertahan lebih lama, sehingga

berpeluang besar mendapatkan Survival Bonus dan posisi akhir yang tinggi tanpa harus mengeliminasi musuh secara langsung.

3.3 Starath : Greedy by Targeting the Weakest



3.3.1. Overview

Starath menerapkan strategi *greedy* berupa menarget bot dengan *energy* terendah yang ada di arena (tidak termasuk Bot ini sendiri tentunya). Saat pertama di-*deploy* ke arena, bot akan *scan* seluruh arena untuk menyimpan seluruh data bot musuh dan hanya menembak bot ber-*energy* terrendah. Adapun kalkulasi firepower berdasarkan *energy* milik bot musuh, memperbesar kemungkinan untuk mendapat kill bot danmendapatkan bonus bullet damage. Starath memiliki pergerakan melingkar untuk memperkecil peluang terkena peluru musuh.

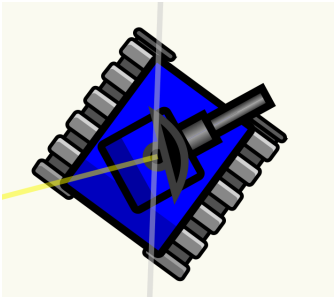
3.3.2. Elemen-elemen Greedy

- Himpunan Kandidat: *Scan* seluruh bot yang berada di arena. Hasil *scan* berupa data *energy* musuh disimpan dalam data struktur Dictionary.
- Himpunan Solusi: satu bot dengan energy terrendah (disimpan dalam variabel targetedId)
- Fungsi Solusi: Memastikan bot yang terpilih merupakan bot ber-*energy* terrendah dan Id dari target valid
- Fungsi Seleksi: Iterasi seluruh data dalam *dictionary* untuk mendapatkan bot Id dengan *energy* paling sedikit
- Fungsi Kelayakan: Memastikan hanya menyeleksi bot yang hidup dengan menghapus data bot yang sudah hancur.
- Fungsi Objektif: memperoleh skor sebanyak mungkin dengan menarget bot lemah, pergerakan yang sulit untuk kena tembak, dan penyesuaian *firepower* untuk manajemen energy.

3.3.3. Evaluasi Efisiensi dan Efektivitas

Starath sudah cukup efektif dalam targeting bot ber-*energy* terrendah. Penggunaan *energy* untuk tembakan pun sudah dibuat untuk efisiensi penggunaan *energy* dan memperbesar peluang dalam mendapatkan *bonus bullet damage.* Namun masih memiliki beberapa kekurangan. Diantaranya yaitu tidak mempertimbangkan posisi dinding. Dengan faktor dinding yang tidak dipertimbangkan, Starath rentan untuk tertabrak tembok yang membuatnya terkena kerusakan. Starath pun tidak bisa membuat tembakan prediksi, sehingga tidak jarang pelurunya meleset.

3.4 Hnfadtya: Greedy by the Nearest Enemy



3.4.1 Overview

Hnfadtya menerapkan strategi *greedy* berupa menarget bot dengan jarak terdekat yang ada di arena dengan cara memutar radar 360 derajat. Ketika memutar radar, bot ini akan mendata seluruh musuh yang ada masing-masing mengenai posisinya (dalam X dan Y), *direction*, dan juga Id bot tersebut. Setelah mendapatkan ListOfEnemy dari seluruh bot musuh, akan dicari bot dengan jarak terdekat. Bot tersebut lah yang akan ditembak. Kekuatan dari tembakan bot ini didasari atas energi yang masih dimilikinya. Semakin besar energi yang masih ada, semakin maksimal pula tembakan yang diberikan.

3.4.2. Elemen-elemen Greedy

- Himpunan Kandidat: Musuh-musuh yang terdeteksi radar
- Himpunan Solusi: Musuh dengan jarak terdekat dari bot
- Fungsi Solusi: Memastikan bot yang terpilih merupakan bot dengan jarak terdekat dalam sekali pemutaran radar
- Fungsi Seleksi: Mencari bot dengan jarak paling dekat dalam ListOfEnemy dengan iterasi
- Fungsi Kelayakan: Memastikan hanya menyeleksi bot yang masih hidup serta meng-*update* posisi setiap bot dan menghapus bot yang telah mati

- Fungsi Objektif: Memaksimalkan skor yang diperoleh dan *output damage* peluru dengan mempertimbangkan energi yang masih tersisa

3.4.3. Evaluasi Efisiensi dan Efektivitas

Bot ini masih belum konsisten dalam memilih musuh dengan jarak yang terpendek. Contohnya, pada pemutaran radar pertama bot berhasil memilih musuh dengan benar tetapi salah pada pemutaran radar selanjutnya. Masalah lain yang masih terjadi adalah bot ini akan menyerang satu bot yang telah dipilih secara terus menerus dengan posisi tembakan mengarah ke arah yang sama. Hal ini disebabkan posisi musuh tidak ter-*update*. Selain itu, masalah yang masih belum terpecahkan adalah kasus bot ini menabrak tembok. Bisa terjadi kasus ketika bot ini menabrak tembok lalu malah terus menerus di posisi yang sama dan menembak asal-asalan. Implikasi dari masalah-masalah tersebut adalah keberhasilan bot dalam suatu permainan sangat tidak dapat ditentukan atau diprediksi.

3.5 Strategi Terpilih

Strategi yang dipilih sebagai *heuristic* utama robot untuk kelompok ini adalah strategi *Greedy by Bullet Damage* & *Bullet Damage Bonus*. Alasan pemilihan strategi ini sebagai perwakilan kelompok adalah berdasarkan evaluasi efisiensi dan efektivitas strategi ini, kami menilai strategi ini berpotensi paling tinggi untuk mendapat skor tertinggi, dan berdasarkan hasil pengujian yang akan dibahas lebih lanjut pada Bab 4, strategi inilah yang berhasil mendapat skor tertinggi.

Bab 4: Implementasi dan Pengujian

4.1 Implementasi

4.2.1. GrwnaBot

Struktur Data, Fungsi, dan Prosedur

Data

- double X, Y → Posisi bot
- double Direction → Arah hadap bot
- double GunDirection → Arah hadap meriam
- double Energy → Energi bot
- bool isIdle → Status idle bot (bergerak atau tidak)
- ScannedBotEvent → Data musuh terdeteksi
- HitBotEvent → Data musuh yang di-*damage*
- HitWallEvent → Data saat bot menabrak dinding
- HitByBulletEvent → Data saat bot terkena tembakan

Fungsi dan Prosedur

- getFirepower(distance) → Menghitung power tembakan optimal berdasarkan jarak dan energi
- chaseEnemy(distance, direction) → Gerak menabrak atau mengitari musuh sesuai jarak
- OnScannedBot(e) → Reaksi saat mendeteksi musuh
- OnHitBot(e) → Reaksi saat menabrak atau menembak musuh
- OnHitWall(e) → Reaksi saat menabrak tembok
- OnHitByBullet(e) → Reaksi saat tertembak oleh bot lain
- Run() → *Main loop* untuk robot

Pseudocode

- Main Loop

```
RUN():
   SET warna bot, meriam, radar, peluru, dll

   WHILE bot masih hidup:
       IF kecepatan == 0 THEN:
           isIdle ← true
```

```
        IF energi < 7 THEN:          // Gerakan melingkar untuk menghindar
            ULANGI 5 KALI:
                BELOK(45)
                MAJU(50)
                BELOK(45)
                MAJU(50)
                BELOK(45)
                MAJU(50)

        IF isIdle == true THEN:      // Jika Bot diam,
            BELOK(90)
            MUNDUR(100)
            PUTAR RADAR(360)
```

- chaseEnemy

```
CIRCLE ENEMY LOGIC(jarak, arahMusuh):
    sudutPutar ← NormalizeRelativeAngle(arahMusuh - arahBot)
    BELOK(sudutPutar)

    IF jarak < 50 THEN:    // Mengitari musuh jika dekat
        BELOK(45)
        MAJU(100)
        BELOK(45)
    ELSE:
        MAJU(500)          // Nge-ram ke musuh

    // Jika bot terjebak maka bergerak belok
    IF speed == 0 DAN posisi aman:
        BELOK(90)
```

- getFirepower

```
GET FIREPOWER(jarak):
    power ← Energy * 2 / (jarak * 0.1)
    IF power < 1 THEN power ← 1
    IF power > 3 THEN power ← 3
    RETURN power
```

- OnHitBot & OnScannedBot

```
BOT(e):
    arahMusuh ← DirectionTo(e.X, e.Y)
    sudutPutar ← NormalizeRelativeAngle(arahMusuh - arahMeriam)
    PUTAR_MERIAM(sudutPutar)

    jarak ← DistanceTo(e.X, e.Y)
    power ← getFirepower(jarak)

    TEMBAK(power)
    chaseEnemy(jarak, arahMusuh)
    PUTAR RADAR(0)
```

- OnHitWall

```
ON HIT WALL:              // Bergerak menjauhi tembok
    BEL0K_KANAN(45)
    MAJU(20)
    BEL0K_KANAN(45)
    MAJU(30)
```

- OnHitByBullet

```
ON HIT BY BULLET:         // Bergerak menjauhi peluru
    isIdle ← false
    BEL0K(90)
    MAJU(200)
    isIdle ← true
```

4.2.2. KaburBot

Struktur Data, Fungsi, dan Prosedur

Data

- double Width → Menyimpan data lebar arena
- double Height → Menyimpan data panjang arena
- struct EnemyBot → Menyimpan data-data bot musuh yang ter-*scan*
- ScannedBotEvent → Data musuh terdeteksi
- HitBotEvent → Data musuh yang di-*damage*
- HitWallEvent → Data reaksi saat menabrak tembok
- BotDeathEvent → Data musuh yang telah hancur
- Dictionary<int, EnemyBot> enemies → Penyimpanan data bot-bot yang hidup
- Random rand → membantu membuat gerakan random saat "gaya" mendekati 0
- const MOVE_DISTANCE → jarak konstan

Fungsi dan Prosedur

- int findWeakestBot() → Fungsi seleksi mencari bot dengan *energy* terrendah
- calculatedFirepower(double enemyEnergy) → Proses kalkulasi daya tembak peluru
- OnScannedBot(e) → Reaksi saat mendeteksi musuh. menyimpan data musuh ke dictionary
- OnHitBot(e) → Reaksi saat menabrak musuh, mencari posisi aman dan menembak musuh.
- OnHitWall(e) → Reaksi saat menabrak tembok
- Run() → *Main loop* untuk robot

Pseudocode

- **Main Loop**

```
Run:

    SET AdjustGunForBodyTurn ← true
    SET AdjustRadarForGunTurn ← true
    SET currEnemies ← EnemyCount

    SET BodyColor ← abu-abu

    WHILE IsRunning DO
        SetTurnRadarRight(360)

        IF currEnemies > EnemyCount THEN

            SET currEnemies ← EnemyCount

        (totalFx, totalFy) ← CalculateTotalEnemyForce()
```

```
        SET smoothing ← 0.8
        SET prevFx ← smoothing × prevFx + (1 - smoothing) × totalFx
        SET prevFy ← smoothing × prevFy + (1 - smoothing) × totalFy

        SET forceMagnitude ← sqrt(prevFx² + prevFy²)

        IF forceMagnitude < 0.1 THEN
            BELOK(45 + random(0, 90))
            SetForward(MOVE_DISTANCE ÷ 2)
            Go()
            CONTINUE

        SET angle ← atan2(prevFy, prevFx) dalam derajat
        SET angle ← NormalizeRelativeAngle(angle)

        IF |angle| < 90 THEN
            BELOK(angle)
            SetForward(MOVE_DISTANCE)
        ELSE
            BELOK(NormalizeRelativeAngle(angle + 180))
            MUNDUR(MOVE_DISTANCE)

        IF X < 50 OR X > 750 OR Y < 50 OR Y > 550 THEN
            BELOK(90)
            MUNDUR(MOVE_DISTANCE)

        JALANKAN semua perintah pergerakan dan tembakan
```

- **getFirePower**

```
getFirepower(enemyDistance : real) → real
    RETURN Min(3, Max(Energy × 2 ÷ (enemyDistance ×
0.1)), 1)
```

- **FiringLogic**

```
FiringLogic(X : real, Y : real):
    SET eDirection ← DirectionTo(X, Y)
```

```
    SET turnAngle ← NormalizeRelativeAngle(eDirection - GunDirection)

    SetTurnGunLeft(turnAngle)

    SET firepower ← getFirepower(DistanceTo(X, Y))

    IF Energy > 30 THEN
        SetFire(firepower)

    JALANKAN semua perintah pergerakan dan tembakan
```

- **CalculateTotalEnemyForce**

```
CalculateTotalEnemyForce → (real, real):
    SET totalFx ← 0
    SET totalFy ← 0

    FOR setiap enemy ∈ enemies DO
        SET dx ← enemy.X - X
        SET dy ← enemy.Y - Y
        SET distanceSq ← dx² + dy²
        IF distanceSq < 1 THEN
            SET distanceSq ← 1

        SET energyFactor ← enemy.Energy ÷ 100
        SET force ← 9000 × energyFactor ÷ distanceSq

        SET angle ← atan2(dy, dx)
        SET totalFx ← totalFx + force × cos(angle)
        SET totalFy ← totalFy + force × sin(angle)

    // Gaya tambahan dari dinding (wall repulsion)
    SET wallMargin ← 60
    SET wallForce ← 10000

    IF X < wallMargin THEN
        totalFx ← totalFx + wallForce ÷ (X²)

    ELSE IF X > Width - wallMargin THEN
        totalFx ← totalFx - wallForce ÷ ((Width - X)²)
```

```
    IF Y < wallMargin THEN
        totalFy ← totalFy + wallForce ÷ (Y²)
    ELSE IF Y > Height - wallMargin THEN
        totalFy ← totalFy - wallForce ÷ ((Height - Y)²)

    RETURN (totalFx, totalFy)
```

- **OnScannedBot**

```
OnScannedBot(e):

    IF enemies berisi key e.ScannedBotId THEN
        SET enemies[e.ScannedBotId] ← EnemyBot(e.X, e.Y, e.Energy, e.Direction)
    ELSE
        Tambahkan ke enemies:
            key   ← e.ScannedBotId
            value ← EnemyBot(e.X, e.Y, e.Energy, e.Direction)

    FiringLogic(e.X, e.Y)
```

- **OnHitBot**

```
OnHitBot(e):
    FiringLogic(e.X, e.Y)
```

- **OnHitWall**

```
OnHitWall(e):

    BELOK(90)
    MUNDUR(100)

    JALANKAN semua perintah pergerakan dan tembakan
```

- **OnBotDeath**

```
OnHitWall(e):
    BELOK(90)

    MUNDUR(100)
    JALANKAN semua perintah pergerakan dan tembakan
```

4.2.3. Starath

Struktur Data, Fungsi, dan Prosedur

Data

- double Energy → Energi bot
- double targetedId → Id bot yang dijadikan target
- struct ScannedBot → Menyimpan data-data bot musuh yang ter-*scan*
- ScannedBotEvent → Data musuh terdeteksi
- HitBotEvent → Data musuh yang di-*damage*
- BotDeathEvent → Data musuh yang telah hancur
- Dictionary<int, ScannedBot> scannetBots → Penyimpanan data bot-bot yang hidup

Fungsi dan Prosedur

- int findWeakestBot() → Fungsi seleksi mencari bot dengan *energy* terrendah
- calculatedFirepower(double enemyEnergy) → Proses kalkulasi daya tembak peluru
- OnScannedBot(e) → Reaksi saat mendeteksi musuh. menyimpan data musuh ke dictionary
- OnHitBot(e) → Reaksi saat menabrak musuh, mencari posisi aman dan menembak musuh.
- OnHitWall(e) → Reaksi saat menabrak tembok
- Run() → *Main loop* untuk robot

Pseudocode

- **Main Loop**

```
Run:
    SET warna tubuh bot menjadi abu-abu gelap
    SET warna meriam dan turret menjadi coklat tua
    SET warna radar, scan, dan peluru menjadi merah

    WHILE bot masih hidup:
        SET MaxSpeed ← 6.0
        SET targetedId ← findWeakestBot()
        MAJU(40000 unit)
        BELOK(10 derajat)
        BELOKMERIAM(20 derajat)
```

```
        JALANKAN semua perintah pergerakan dan tembakan
```

- findWeakestBot

```
findWeakestBot → integer:
    SET weakestId ← -1
    SET lowestEnergy ← tak hingga (∞)

    FOR setiap entry dalam dictionary scannedBots
        IF entry.Value.energy < lowestEnergy THEN
            SET lowestEnergy ← entry.Value.energy
            SET weakestId ← entry.Key
    RETURN weakestId
```

- calculatedFirepower

```
calculatedFirepower(enemyEnergy):
    SET ratio ← (enemyEnergy - 20) / (100 - 20)
    SET rawFirepower ← 3.0 - ratio × (3.0 - 1.01)
    SET firepower ← MIN(rawFirepower, 3.0)

    SetFire(firepower)
    JALANKAN semua perintah pergerakan dan tembakan
```

- OnScannedBot

```
OnScannedBot(e):

    IF scannedBots berisi kunci e.ScannedBotId THEN
            SET scannedBots[e.ScannedBotId] ← ScannedBot(e.X, e.Y, e.Energy,
DistanceTo(e.X, e.Y))
    ELSE
            key   ← e.ScannedBotId
            value ← ScannedBot(e.X, e.Y, e.Energy, DistanceTo(e.X, e.Y))
```

```
    IF targetedId = e.ScannedBotId THEN
        calculatedFirepower(scannedBots[e.ScannedBotId].energy)
```

- OnHitBot

```
OnHitBot(e):

    SET deltaX ← e.X - X
    SET deltaY ← e.Y - Y

    SET absoluteBearing ← arctangent(deltaX / deltaY) // dalam derajat
    SET absoluteBearing ← (absoluteBearing + 360) modulo 360

    SET relativeBearing ← NormalizeRelativeAngle(absoluteBearing - Direction)

    IF |relativeBearing| < 90 THEN
        MUNDUR(150 unit)
        BELOK(90 - relativeBearing derajat)
    ELSE
        MAJU(150 unit)
        BELOK(90 + relativeBearing derajat)
    ENDIF

    TEMBAK(3)
    JALANKAN semua perintah pergerakan dan tembakan
```

- OnBotDeath

```
OnBotDeath(e):

    Hapus entri dengan key e.VictimId dari scannedBots
    SET targetedId ← -1
```

4.2.4. Hnfadtya

Struktur Data, Fungsi, dan Prosedur

Data

- List<EnemyBot> ListOfEnemy → tempat penyimpanan data bot-bot musuh
- ScannedBotEvent → Data musuh terdeteksi
- HitBotEvent → Data musuh yang di-*damage*
- BotDeathEvent → Data musuh yang telah hancur
- HitWallEvent → Data saat bot menabrak dinding
- EnemyBot → Kelas yang berisi konstruktor untuk menyimpan data musuh

Fungsi dan Prosedur

- ChasingTarget(double TargetDirection, double TargetDistance) → Prosedur untuk urusan pergerakan bot agar tidak mudah tertembak
- OnScannedBot(e) → Reaksi saat mendeteksi musuh. menyimpan data musuh ke ListOfEnemy dan menembaknya
- OnHitBot(e) → Reaksi saat menabrak musuh, mengarahkan tembakan ke arah musuh dengan kekuatan maksimum.
- OnBotDeath(e) → Reaksi saat ada bot yang mati, menghapus data musuh dalam ListOfEnemy
- OnHitWall(e) → Reaksi saat menabrak tembok, berbalik arah
- Run() → *Main loop* untuk robot

Pseudocode

- Main Loop

```
RUN():
    SELAMA bot masih hidup:
        PUTAR RADAR(360)            // terus scanning
        MAJU(100)
        BELOK(30)
        JALANKAN perintah
```

- OnScannedBot

```
ON_SCANNED_BOT(e):
    DAPATKAN jarak, arah, posisi musuh (X, Y), dan ID
```

```
    CARI index musuh berdasarkan ID dalam daftar musuh
    JIKA musuh belum ada:
        TAMBAH musuh ke daftar
    JIKA musuh sudah ada:
        UPDATE info musuh di daftar
    target ← musuh dengan jarak terdekat
    JIKA jumlah musuh terdeteksi == total musuh:
        ARAHKAN GUN ke musuh
        JIKA sudut ke musuh kecil dan senjata siap:
            TEMBAK

        // Mengejar
        panggil CHASING_TARGET(target.Direction, target.Distance)
        RESCAN()
```

- ChasingTarget

```
CHASING_TARGET(arahMusuh, jarakMusuh):
    sudutPutar ← NormalizeRelativeAngle(arahMusuh - arahBot)
    BELOK(sudutPutar)
    MAJU(jarakMusuh)
```

- OnBotDeath

```
ON_BOT_DEATH(e):
    HAPUS musuh yang sudah mati dari daftar musuh
```

- OnhitBot

```
ON_HIT_BOT(e):
    ARAHKAN GUN ke musuh
    JIKA sudut kecil DAN senjata siap:
        TEMBAK
    JIKA tepat di depan:
        RESCAN()
```

- OnHitWall

```
ON_HIT_WALL(e):
    MUNDUR(10)
    BEL0K(90)
    MAJU(20)
```

4.2 Pengujian

4.2.1. Hasil Pengujian

Rank	Name	Total Score	Survival	Surv. Bonus	Bullet Dmg.	Bullet Bonus	Ram Dmg.	Ram Bonus	1sts	2nds	3rds
1	Grwna 1.0	2712	850	90	1415	170	187	0	5	3	0
2	Starath 1.0	1871	800	120	739	109	103	0	2	4	1
3	Kabur 1.0	699	300	30	297	11	61	0	2	0	4
4	Hnfadtya 1.0	622	450	0	148	7	17	0	0	2	4

Rank	Name	Total Score	Survival	Surv. Bonus	Bullet Dmg.	Bullet Bonus	Ram Dmg.	Ram Bonus	1sts	2nds	3rds
1	Kabur 1.0	1053	500	60	431	22	28	12	3	1	0
2	Starath 1.0	889	550	30	286	11	11	0	1	4	0
3	Hnfadtya 1.0	577	300	30	214	6	26	0	1	0	3
4	Grwna 1.0	487	100	0	325	0	62	0	0	0	2

Rank	Name	Total Score	Survival	Surv. Bonus	Bullet Dmg.	Bullet Bonus	Ram Dmg.	Ram Bonus	1sts	2nds	3rds
1	Grwna 1.0	1699	500	0	930	86	182	0	3	3	1
2	Hnfadtya 1.0	1622	850	150	532	34	55	0	3	2	1
3	Starath 1.0	1041	450	30	408	27	125	0	1	1	4
4	Kabur 1.0	994	450	30	463	12	37	0	1	2	2

Rank	Name	Total Score	Survival	Surv. Bonus	Bullet Dmg.	Bullet Bonus	Ram Dmg.	Ram Bonus	1sts	2nds	3rds
1	Starath 1.0	1462	750	90	484	44	65	29	3	2	2
2	Grwna 1.0	1387	400	0	803	77	107	0	1	2	2
3	Kabur 1.0	1291	550	60	600	72	10	0	3	1	1
4	Hnfadtya 1.0	444	350	30	44	0	20	0	0	2	2

Rank	Name	Total Score	Survival	Surv. Bonus	Bullet Dmg.	Bullet Bonus	Ram Dmg.	Ram Bonus	1sts	2nds	3rds
1	Grwna 1.0	1055	300	30	556	59	110	0	1	1	2
2	Kabur 1.0	1024	500	60	410	19	35	0	2	2	0
3	Starath 1.0	715	350	0	270	27	67	0	0	2	2
4	Hnfadtya 1.0	676	350	60	216	16	35	0	2	0	1

Rank	Name	Total Score	Survival	Surv. Bonus	Bullet Dmg.	Bullet Bonus	Ram Dmg.	Ram Bonus	1sts	2nds	3rds
1	Grwna 1.0	2580	800	90	1243	209	238	0	5	2	1
2	Starath 1.0	1701	900	120	507	73	101	0	3	3	1
3	Kabur 1.0	972	300	0	558	13	100	0	0	3	4
4	Hnfadtya 1.0	445	350	0	55	0	40	0	0	0	2

Rank	Name	Total Score	Survival	Surv. Bonus	Bullet Dmg.	Bullet Bonus	Ram Dmg.	Ram Bonus	1sts	2nds	3rds
1	Grwna 1.0	2012	650	60	1017	98	170	16	2	5	0
2	Kabur 1.0	1788	800	90	749	73	71	5	4	1	2
3	Starath 1.0	1362	600	60	524	52	126	0	2	2	3
4	Hnfadtya 1.0	405	300	0	73	0	28	5	0	0	3

Rank	Name	Total Score	Survival	Surv. Bonus	Bullet Dmg.	Bullet Bonus	Ram Dmg.	Ram Bonus	1sts	2nds	3rds
1	Hnfadtya 1.0	1409	750	90	447	17	104	0	3	1	4
2	Starath 1.0	1266	700	90	396	48	32	0	3	1	1
3	Grwna 1.0	1264	400	0	612	82	154	15	2	2	2
4	Kabur 1.0	951	400	30	401	31	89	0	0	4	1

Rank	Name	Total Score	Survival	Surv. Bonus	Bullet Dmg.	Bullet Bonus	Ram Dmg.	Ram Bonus	1sts	2nds	3rds
1	Grwna 1.0	3413	1100	210	1641	276	185	0	7	1	0
2	Starath 1.0	1210	600	30	418	30	131	0	1	4	2
3	Kabur 1.0	1039	350	0	579	9	101	0	1	2	2
4	Hnfadtya 1.0	706	600	0	77	0	29	0	0	2	5

Rank	Name	Total Score	Survival	Surv. Bonus	Bullet Dmg.	Bullet Bonus	Ram Dmg.	Ram Bonus	1sts	2nds	3rds
1	Grwna 1.0	1656	550	90	777	87	142	10	3	1	1
2	Starath 1.0	1221	650	30	450	26	65	0	1	3	3
3	Hnfadtya 1.0	1068	600	60	300	23	78	6	2	2	2
4	Kabur 1.0	804	250	0	479	19	56	0	1	1	1

4.2.2. Analisis Hasil Pengujian

Grwna

Bot ini berhasil mendapat skor tertinggi pada 7 dari 10 percobaan, yang berarti bot ini unggul dan berhasil mendapat nilai yang optimal untuk sebagian besar waktu. Adapun situasi dimana bot ini tidak mendapat hasil optimal adalah jika terjadi 1v1 dengan bot yang bergerak berputar, sehingga bot ini tidak dapat mengejar dan tidak dapat menembak dengan akurat.

Starath

Walau masih kalah dengan bot Grwna, bot ini masih cenderung mendapatkan posisi ke-2 skor tertinggi. Selain itu, bot ini dapat menang terhadap bot Grwna jika terjadi 1v1 dan *infinite loop* bot Grwna mengejar Starath dan tembakannya tidak akurat, namun jika bot Grwna menge-*ram*, maka bot ini akan kalah.

Hnfaditya

Bot berhasil mendapat nilai optimal sesuai *greedy* yang digunakan, walau masih kerap kalah dari bot lain. Bot ini unggul ketika dapat *pick-off* musuh secara satu persatu, atau saat terjadi 1v1 lokal, karena bot ini bekerja baik terhadap satu target dan tidak baik dalam meng-*handle* banyak musuh sekaligus.

Kabur

Bot ini cenderung tidak mendapat tertinggi karena bot Grwna yang bersifat sangat agresif, sehingga mengalahkan strategi menghindarnya. Namun ada saat dimana bot lain cenderung menyerang satu sama lain sehingga bot ini tidak *overwhelmed* dan dapat menghindari serangan, dan berhasil meng-*snipe* bot lain dari jauh.

Bab 5: Kesimpulan dan Saran

5.1 Kesimpulan

Pada tugas besar ini kami berhasil mengimplementasikan beberapa *heuristic* algoritma greedy untuk merancang beberapa robot tank pada robocode. Implementasi robocode dengan menggunakan algoritma greedy dapat membantu membuat strategi perancangan robot yang lebih baik sekaligus membantu pemahaman tentang algoritma greedy dengan mendalami bagaimana algoritma greedy diaplikasikan untuk mengoptimalkan pengambilan keputusan.

5.2 Saran

a. Optimasi lebih lanjut: Selain penggunaan *greedy* Robot juga dapat dioptimalkan perolehan skornya dengan menggunakan strategi umum atau algoritma khusus dari Robocode, seperti *Wave Surfing* dan *Guess Factor*.

b. Memahami Engine secara mendalam: Seringkali implementasi yang ditulis tidak sesuai harapan sebab Engine bekerja tidak sesuai ekspektasi, seperti gerak *zigzag* yang tidak dapat diimplementasikan menggunakan *Set methods*.

Lampiran

Repository: https://github.com/grwna/Tubes1_Afhiyh

Link Video: https://youtu.be/lA1JgXBxvck

Tabel Progress:

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.		

Referensi

[1]. R. Munir, "Algoritma Greedy (Bagian 1)" Institut Teknologi Bandung, 2025. [Online]. Available: https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2024-2025/04-Algoritma-Greedy-(2025)-Bag1.pdf. [Accessed: Mar. 5, 2025].

[2]. R. Munir, "Algoritma Greedy (Bagian 2)" Institut Teknologi Bandung, 2025. [Online]. Available: https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2024-2025/05-Algoritma-Greedy-(2025)-Bag2.pdf. [Accessed: Mar. 5, 2025].

[3]. R. Munir, "Algoritma Greedy (Bagian 3)" Institut Teknologi Bandung, 2025. [Online]. Available: https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2024-2025/06-Algoritma-Greedy-(2025)-Bag3.pdf. [Accessed: Mar. 5, 2025].