

LAPORAN TUGAS BESAR 1

IF2211 STRATEGI ALGORITMA



Kelompok 35 : Berkah Ramadhan

Ranashahira Reztaputri	13523007
Najwa Kahani Fatima	13523043
Muhammad Izzat Jundy	13523092

PROGRAM STUDI TEKNIK INFORMATIKA
SEKOLAH TEKNIK ELEKTRO DAN INFORMATIKA
INSTITUT TEKNOLOGI BANDUNG
2025

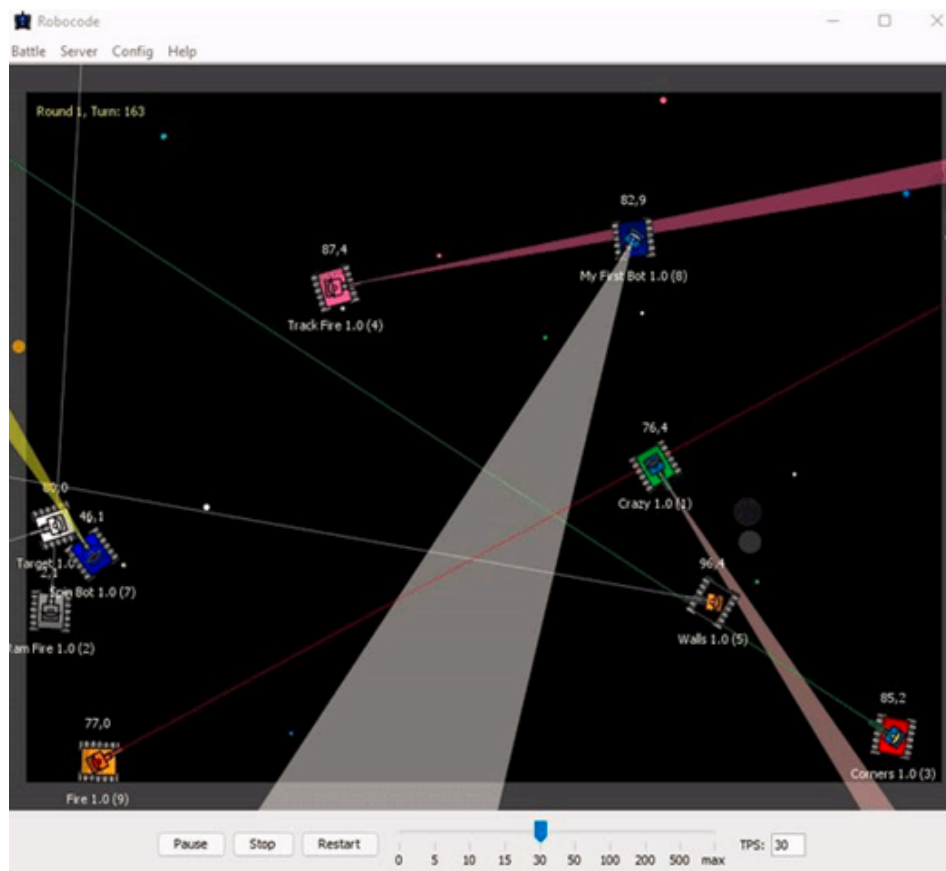
DAFTAR ISI

DAFTAR ISI.....	1
BAB I.....	3
DESKRIPSI MASALAH.....	3
BAB II.....	8
LANDASAN TEORI.....	8
2.1. Algoritma Greedy.....	8
2.2. Robocode Tank Royale.....	9
2.2.1. Aksi Bot.....	9
2.2.2. Implementasi Algoritma Greedy pada Bot.....	10
2.2.3 Cara Menjalankan Bot.....	10
BAB III.....	17
APLIKASI STRATEGI GREEDY.....	17
3.1. Alternatif Solusi Greedy.....	17
3.1.1. Greedy by Enemy Targetting: GritBot.....	17
3.1.2. Greedy by Avoiding Damage: Dreadfang.....	18
3.1.3. Greedy by Spinning, Positioning, and Evaluating: KingBot.....	19
3.1.4. Greedy by Dodging and Attack: Syzygy.....	20
3.2. Strategi Greedy yang Dipilih.....	21
BAB IV.....	22
IMPLEMENTASI DAN PENGUJIAN.....	22
4.1. Implementasi Alternatif Solusi Greedy.....	22
4.1.1. Greedy by Enemy Targetting: GritBot.....	22
4.1.2. Greedy by Avoiding Damage: Dreadfang.....	25
4.1.3. Greedy by Spinning, Positioning, and Evaluating: KingBot.....	27
4.1.4. Greedy by Dodging and Attack: Syzygy.....	28
4.2. Penjelasan Solusi Greedy yang Dipilih.....	31
4.3. Pengujian Bot.....	33
4.3.1 GritBot vs Dreadfang.....	34
4.3.2 GritBot vs KingBot.....	34
4.3.3 GritBot vs Syzygy.....	35
4.3.4 Dreadfang vs KingBot.....	35
4.3.5 Dreadfang vs Syzygy.....	36
4.3.6 Kingbot vs Syzygy.....	36
4.3.7 Semua Bot.....	37
4.4. Analisis Hasil Pengujian Bot.....	37
BAB V.....	39
KESIMPULAN DAN SARAN.....	39
5.1. Kesimpulan.....	39

5.2. Saran.....	39
LAMPIRAN.....	40
DAFTAR PUSTAKA.....	41

BAB I

DESKRIPSI MASALAH



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:

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

Pada setiap turn, bot akan secara otomatis menerima informasi terbaru tentang posisinya dan orientasinya di medan perang. Bot juga akan mendapatkan informasi tentang bot musuh ketika mereka terdeteksi oleh pemindai. API (Application Programming Interface) bot resmi secara otomatis mengirimkan niat bot ke server di balik layar. Game engine yang akan digunakan pada tugas besar ini tidak mengikuti aturan default mengenai komponen Round dan Turns.

2. Batas Waktu Giliran

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 melewati turn tersebut. Jika bot melewati 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. Berikut beberapa kondisi yang dapat berdampak pada energi bot:

- a. Bot akan kehilangan energi jika ditembak atau ditabrak oleh bot musuh.
- b. Bot juga akan kehilangan energi jika menembakkan meriamnya.

- c. 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.
- d. 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)

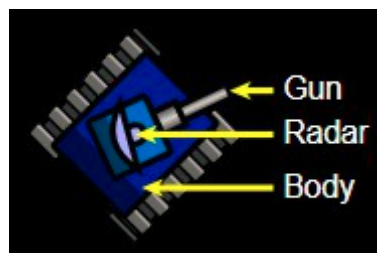
Saat menembakkan peluru, meriam akan menjadi panas. Peluru yang lebih berat menghasilkan lebih banyak panas dibandingkan peluru yang lebih ringan. Ketika meriam terlalu panas, bot tidak dapat menembak hingga suhu meriam turun ke nol. Selain itu, meriam juga sudah dalam keadaan panas di awal round dan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

6. Tabrakan

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:



- a. *Body* adalah bagian utama dari tank yang digunakan untuk menggerakkan tank.

- b. *Gun* digunakan untuk menembakkan peluru dan dapat berputar bersama *body* atau independen dari *body*.
- c. *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.

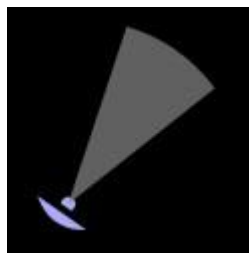
Setiap bagian tubuh memiliki kecepatan putar yang berbeda. Radar adalah bagian tercepat dan dapat berputar hingga 45 derajat per giliran, yang berarti dapat berputar 360 derajat dalam 8 giliran. Turret dan meriam dapat berputar hingga 20 derajat per giliran.

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. Tidak ada energi yang dikonsumsi saat bot bergerak atau berbelok.

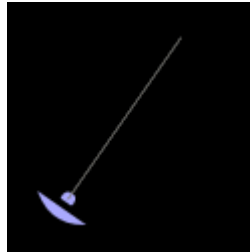
10 Pemindaian

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.

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.



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:

- a. Bullet Damage: Bot mendapatkan poin sebesar *damage* yang dibuat kepada bot musuh menggunakan peluru.
- b. Bullet Damage Bonus: Apabila peluru berhasil membunuh bot musuh, bot mendapatkan poin sebesar 20% dari *damage* yang dibuat kepada musuh yang terbunuh.
- c. Survival Score: Setiap ada bot yang mati, bot lain yang masih bertahan pada ronde tersebut mendapatkan 50 poin.
- d. Last Survival Bonus: Bot terakhir yang bertahan pada suatu ronde akan mendapatkan 10 poin dikali dengan banyaknya musuh.
- e. Ram Damage: Bot mendapatkan poin sebesar 2 kalinya *damage* yang dibuat kepada bot musuh dengan cara menabrak.
- f. Ram Damage Bonus: Apabila musuh terbunuh dengan cara ditabrak, bot mendapatkan poin sebesar 30% dari *damage* yang dibuat kepada musuh yang terbunuh.

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

BAB II

LANDASAN TEORI

2.1. Algoritma Greedy

Greedy adalah paradigma algoritmik yang membangun solusi langkah demi langkah, selalu memilih bagian berikutnya yang menawarkan manfaat paling jelas dan langsung. Pada setiap langkah, harus dibuat keputusan yang terbaik dalam menentukan pilihan sehingga tidak bisa mundur lagi (kembali) ke langkah sebelumnya. Jadi, pada setiap langkah, kita memilih optimum lokal (*local optimum*) dengan harapan bahwa langkah sisanya mengarah ke solusi optimum global (*global optimum*). Algoritma greedy seringkali digunakan untuk optimasi. Persoalan yang dapat dioptimasi dengan algoritma greedy dapat dilihat dengan mengetahui karakteristik dari algoritma greedy. Karakteristik dari algoritma greedy, diantaranya adalah:

1. Algoritma greedy sederhana dan mudah dipahami.
2. Algoritma greedy efisien dalam hal kompleksitas waktu, dan sering kali memberikan solusi yang cepat.
3. Algoritma greedy tidak mempertimbangkan kembali pilihan-pilihan sebelumnya, karena membuat keputusan berdasarkan informasi terkini tanpa melihat ke depan.

Selain memiliki karakteristik, algoritma Greedy juga memiliki elemen-elemen, yaitu:

1. Himpunan kandidat, C : berisi kandidat yang akan dipilih pada setiap langkah.
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 meminimumkan.

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

2.2. Robocode Tank Royale

Robocode adalah permainan pemrograman yang tujuannya adalah membuat kode bot. Dalam hal ini, bot merupakan tank yang harus bersaing dengan tank lain di arena pertempuran virtual. "Pemain" dalam permainan ini adalah programmer (coder) bot, yang tidak akan memiliki pengaruh langsung pada permainan saat sedang berjalan. Sebaliknya, pemain ("Robocoder") harus menulis program yang menjadi otak tank. Program (kode) tersebut memberi tahu bagaimana tank harus berperilaku dan bereaksi terhadap peristiwa yang terjadi di arena pertempuran. Kode tersebut harus memberi tahu tank cara bergerak di medan perang dan menghindari tembakan musuh saat bergerak dalam situasi yang berbeda. Pada saat yang sama, tank harus memindai tank musuh dengan memutar radarnya dan menembakkan senjatanya ke arah tank musuh. Semua bot (tank) mendapat skor dan penempatan tergantung pada seberapa baik mereka tampil di medan perang. Jadi, tujuan dari Robocode Tank Royale adalah untuk mendapatkan skor dan penempatan setinggi mungkin.

2.2.1. Aksi Bot

Pada Robocode Tank Royale, permainan berlangsung dalam *turn-based simulation*, di mana setiap bot diberi kesempatan untuk bertindak dalam satu siklus permainan (*tick*). Dalam setiap turn, bot dapat melakukan berbagai aksi tergantung pada strategi yang diterapkan. Bot dapat bergerak maju atau mundur dengan kecepatan tertentu serta berputar ke kiri atau kanan untuk mengubah arah, yang sering digunakan untuk menghindari serangan atau mencari posisi strategis. Radar bot dapat berputar secara independen dari badan tank, memungkinkan bot untuk mendeteksi lawan dan menyesuaikan strategi, apakah itu menembak, menghindar, atau mengejar. Dalam menyerang, bot dapat menembakkan peluru dengan kekuatan tertentu, berkisar antara 1 hingga 3 unit energi, di mana peluru yang lebih kuat memberikan damage lebih besar tetapi juga menghabiskan lebih banyak energi. Jika tembakan mengenai lawan, bot mendapatkan tambahan energi, tetapi jika meleset, energi hanya terbuang percuma.

Bot beroperasi menggunakan metode utama `run()`, yang berisi *main loop* untuk menjalankan aksi bot di setiap *turn*. Metode ini harus tetap berjalan selama bot aktif, dengan menggunakan perulangan `while(isRunning())` agar tidak keluar sebelum pertandingan berakhir. Dalam loop ini, bot menjalankan berbagai metode untuk memilih target lawan, mengatur pergerakan, membidik dan menembak, memutar radar guna mendeteksi musuh, serta berbagai metode lainnya. Semua aksi ini dikirimkan ke server menggunakan metode `go()`, yang menandakan akhir dari turn saat ini dan memproses tindakan yang telah ditentukan.

Selama battle, bot dapat bergerak maju atau mundur dengan `Forward()` atau `Back()`, serta berputar menggunakan `setTurnLeft()` atau `setTurnRight()`, yang memungkinkan bot menghindari serangan atau mencari posisi strategis. Radar bot dapat berputar secara independen dari badan tank menggunakan `setTurnRadarLeft()` atau `setTurnRadarRight()`, memungkinkan bot untuk mendeteksi musuh dan menyesuaikan strategi berdasarkan informasi yang diterima. Saat menyerang, bot menggunakan `setFire(power)`, di mana kekuatan tembakan berkisar antara 1 hingga 3 unit energi; tembakan yang lebih kuat memberikan damage lebih

besar tetapi menghabiskan lebih banyak energi. Jika tembakan mengenai lawan, bot mendapatkan tambahan energi, sedangkan jika meleset, energi terbuang sia-sia.

Bot juga harus menerapkan strategi bertahan dan menyerang dengan baik. Beberapa bot menghindari tembakan lawan menggunakan gerakan zig-zag atau pola acak, sementara bot lain memilih strategi agresif dengan mendekati lawan untuk menembak dari jarak dekat. Ada pula bot yang menggunakan strategi "camping", dengan menunggu lawan di satu tempat dan menyerang dari kejauhan. Manajemen energi menjadi krusial, karena tanpa energi, bot akan dinonaktifkan dan tidak dapat bertindak lebih lanjut. Oleh karena itu, bot harus cerdas dalam mengelola energinya, baik dalam menembak maupun menghindari tabrakan, agar dapat bertahan lebih lama dalam pertempuran. Dengan menerapkan strategi yang efektif dan mengoptimalkan penggunaan metode API, bot dapat meningkatkan peluangnya untuk bertahan hingga akhir dan memenangkan pertandingan.

2.2.2. Implementasi Algoritma Greedy pada Bot

Objektif utama dari Robocode Tank Royale adalah untuk mendapatkan skor setinggi mungkin dibandingkan bot yang lain. Dengan objektif ini, kita dapat menerapkan strategi greedy dengan selalu mengambil aksi yang memberikan keuntungan terbesar dalam setiap turn. Ini sesuai dengan definisi greedy yang membangun solusi langkah demi langkah. Strategi greedy dapat diterapkan dengan cara memprioritaskan tindakan yang langsung meningkatkan skor atau memperpanjang kelangsungan hidup bot. Misalnya, bot dapat selalu memilih untuk menembak ketika musuh berada dalam jangkauan, menggunakan kekuatan tembakan maksimal yang masih memungkinkan bot untuk mempertahankan energinya. Jika lawan berada dalam jarak yang cukup dekat dan tembakan dipastikan mengenai target, bot akan langsung menembak untuk mendapatkan poin dan memulihkan energi.

Selain dalam menyerang, strategi greedy juga dapat diterapkan dalam pergerakan bot. Bot dapat memilih untuk terus bergerak ke posisi yang mengurangi kemungkinan terkena tembakan musuh, seperti dengan melakukan gerakan zig-zag atau mencari tempat yang lebih sulit dijangkau lawan. Jika bot mendeteksi bahwa tetap diam akan meningkatkan risiko terkena tembakan, ia akan segera bergerak ke arah yang lebih aman. Bot juga bisa memanfaatkan tabrakan dengan lawan untuk memberikan damage jika energinya lebih tinggi, atau sebaliknya, menghindari tabrakan jika energinya lebih rendah.

Dalam penggunaan radar, bot dapat mengadopsi strategi greedy dengan terus-menerus memutar radar dan langsung membidik serta menyerang begitu musuh terdeteksi. Jika lawan menghilang dari radar, bot akan memilih untuk bergerak atau menyesuaikan arah radar untuk meningkatkan peluang mendeteksi kembali musuh. Dengan pendekatan ini, bot selalu mengambil langkah yang memberikan keuntungan instan, baik dalam hal serangan maupun pertahanan, tanpa terlalu mempertimbangkan efek jangka panjang.

2.2.3 Cara Menjalankan Bot

1. Build game engine

```
./gradlew :gui-app:clean  
./gradlew :gui-app:build
```

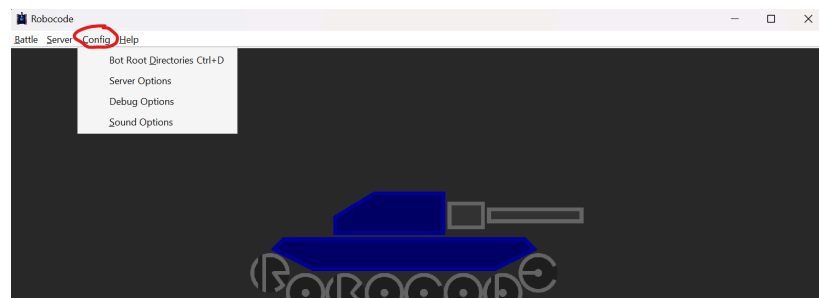
2. Jalankan file .jar aplikasi GUI

```
java -jar robocode-tankroyale-gui-0.30.0.jar
```

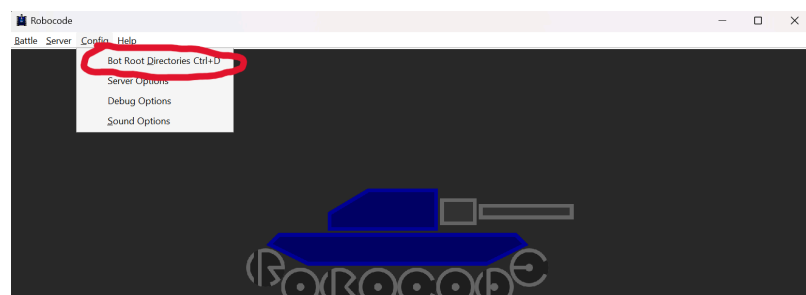
3. Build bot yang ingin dimainkan dengan menjalankan file .cmd milik bot tersebut (file .cmd terdapat pada template bot).

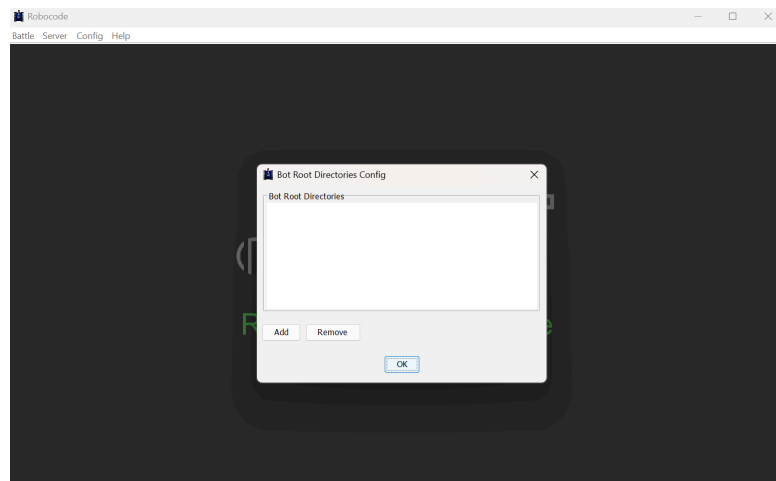
```
./<nama_bot>.cmd
```

4. Setup konfigurasi booter
 - a. Klik tombol “Config”



- b. Klik tombol “Bot Root Directories”

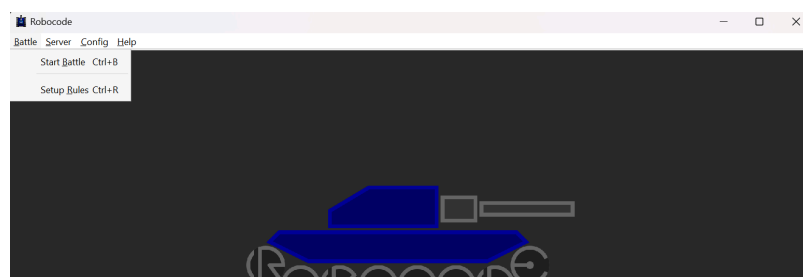




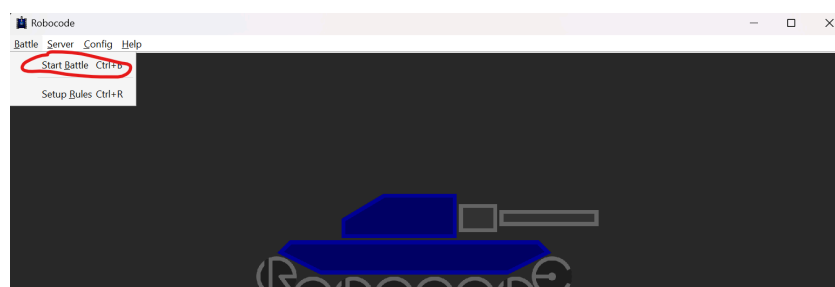
- c. Masukkan *directory* src/main-bot untuk menjalankan bot utama dan src/alternative-bots untuk menjalankan bot alternatif

5. Jalankan sebuah battle

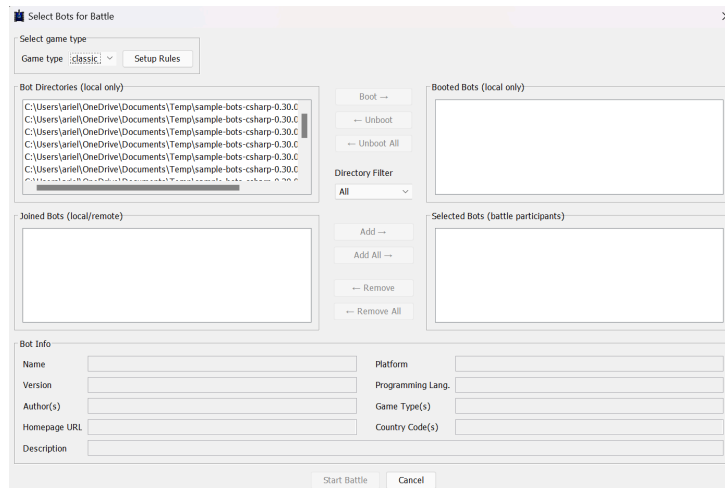
- a. Klik tombol “Battle”



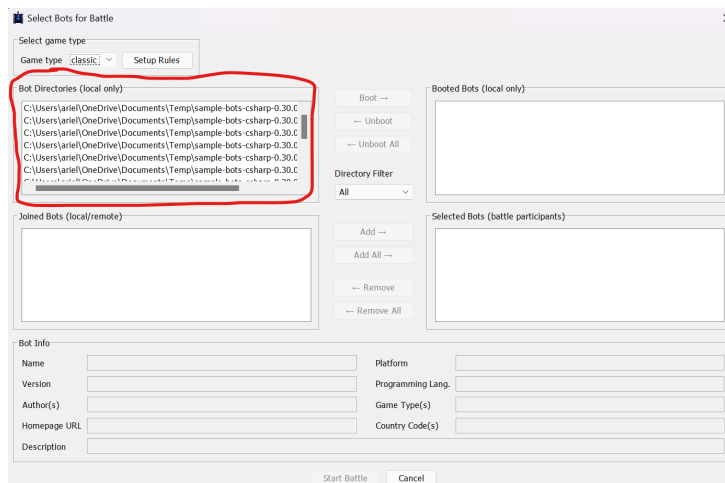
- b. Klik tombol “Start Battle”



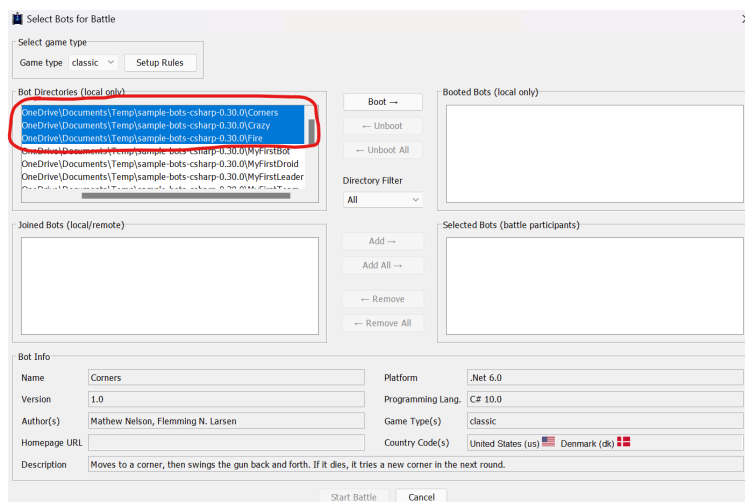
- c. Akan muncul panel konfigurasi permainan



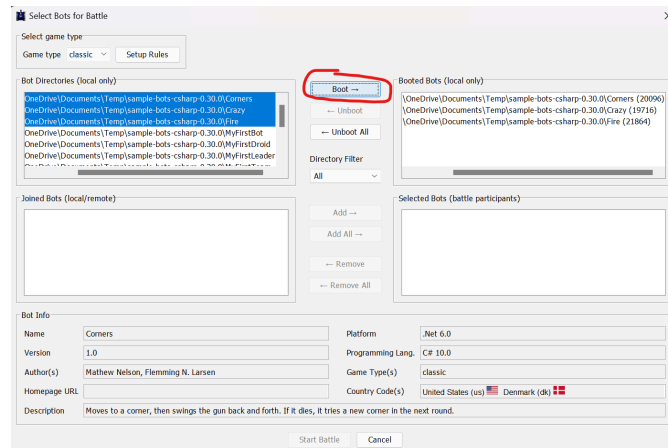
- d. Bot-bot di dalam directory yang telah disetup pada proses konfigurasi akan otomatis muncul pada kotak kiri-atas



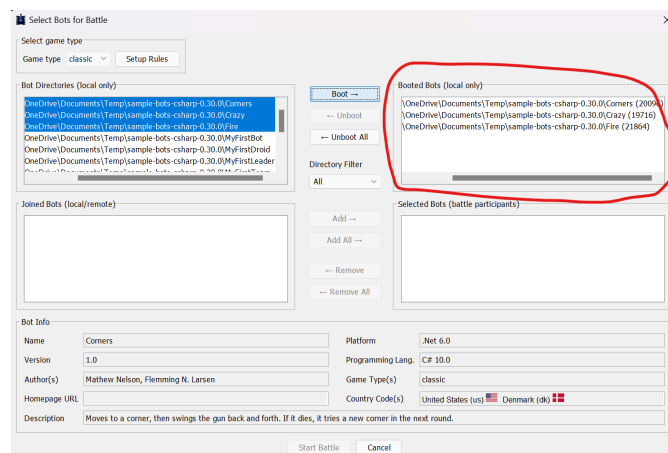
- e. Boot bot yang ingin Anda mainkan dengan memilih bot yang ingin dimainkan pada kotak kiri-atas



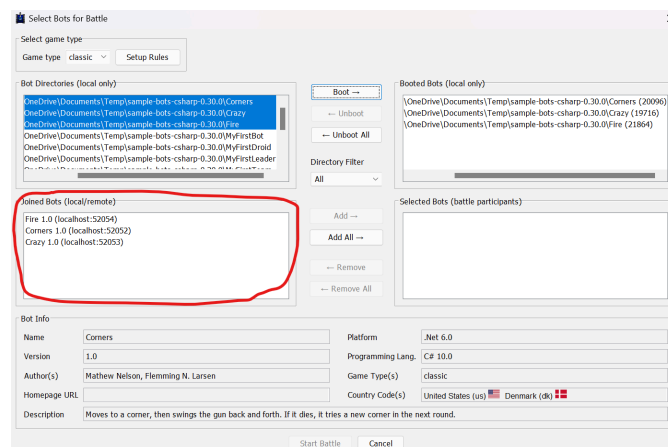
- f. Klik tombol “Boot →”



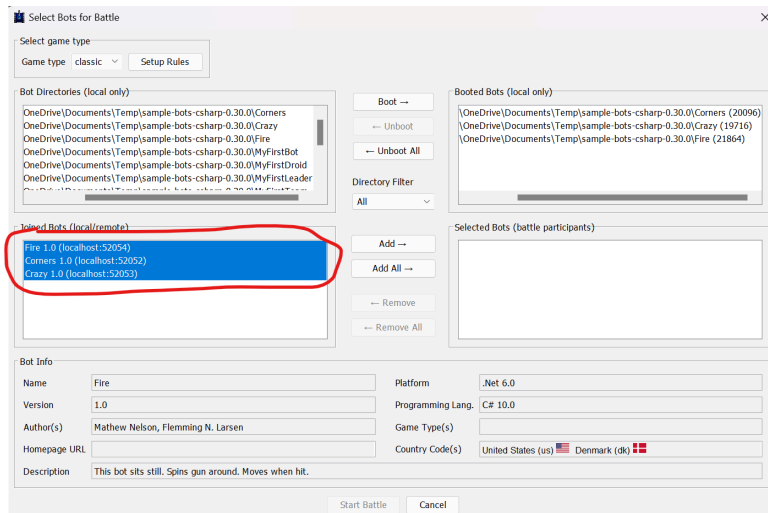
g. Bot yang telah dipilih akan muncul pada kotak kanan-atas



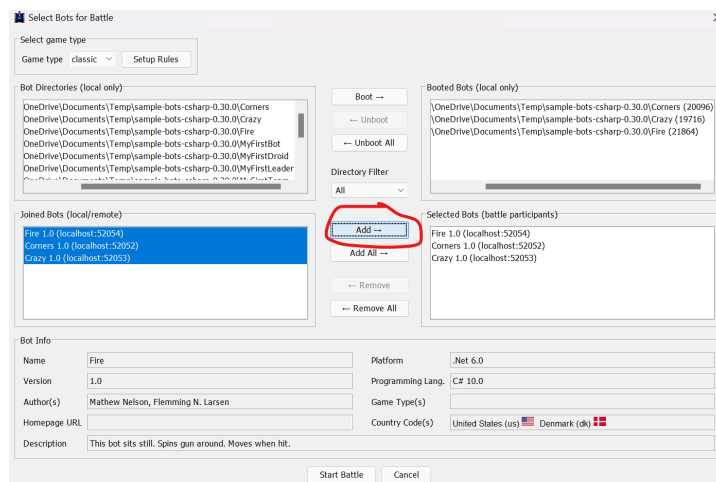
h. Bot yang berhasil di-boot akan muncul pada kotak kiri-bawah



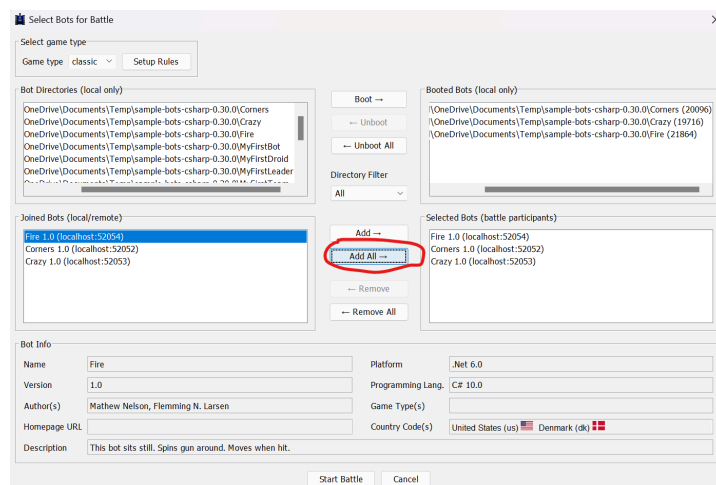
i. Tambahkan bot ke dalam permainan: pilih bot yang ingin ditambahkan ke dalam permainan pada kotak kiri-bawah



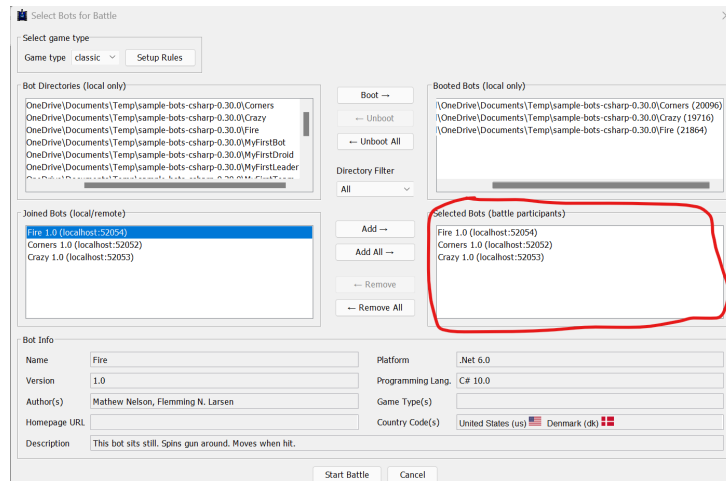
j. Klik tombol “Add →”



k. Apabila Anda ingin menambahkan semua bot, klik tombol “Add All →”



l. Bot yang telah ditambahkan akan otomatis muncul pada kotak kanan-bawah



m. Mulai permainan dengan menekan tombol “Start Battle”

BAB III

APLIKASI STRATEGI GREEDY

Seperti yang telah disampaikan pada Bab II, algoritma greedy memiliki elemen-elemen. Dalam implementasi ini, kami menggunakan berbagai pendekatan greedy. Pada subbab 3.1, akan dijelaskan bagaimana setiap pendekatan tersebut dikaitkan dengan elemen-elemen greedy melalui proses pemetaan yang sistematis.

3.1. Alternatif Solusi Greedy

3.1.1. Greedy by Enemy Targetting: GritBot

Bot ini dijalankan dengan algoritma Greedy by Enemy Targetting untuk dapat memaksimalkan poin akhir. Secara default, bot akan bergerak maju setiap turn datang dan bergerak maju lebih jauh jika bot sudah memilih target. Pemilihan target dilakukan secara acak bergantung pada urutan terdeteksi oleh radar. Jika belum menemukan target, maka radar bot akan berputar sejauh 360 derajat. Jika telah menemukan target, secara default radar bot akan berputar pada derajat yang lebih kecil searah dan berlawanan arah jarum jam secara berulang untuk dapat mengetahui lokasi terbaru dari target. Setiap musuh yang terdeteksi akan masuk ke dalam list musuh yang berisi informasi-informasi umum. Data ini berguna untuk menyimpan lokasi dan energi terakhir target untuk penentuan gerakan bot selanjutnya: yaitu menghindari atau menembak. Jika energi musuh berkurang, maka ada kemungkinan musuh menembakkan peluru sehingga bot akan menghindari. Saat bot mendeteksi musuh, bot akan menembak dengan mengarahkan gun ke posisi akurat musuh lalu menembak terus menerus sesuai dengan jarak musuh dan energi sisa bot.

1. Mapping Elemen Greedy

- a. Himpunan kandidat: Aksi-aksi yang dapat diambil bot dalam setiap langkah, seperti menghindari dan menyerang (menembak) musuh.
- b. Himpunan solusi: Serangkaian aksi yang dipilih dalam pertempuran berdasarkan kondisi tertentu.
- c. Fungsi solusi: Memeriksa apakah bot masih bertahan dan berhasil memperoleh skor setinggi mungkin.
- d. Fungsi seleksi: Memilih aksi yang memaksimalkan poin dari menembak lawan dengan mengunci target dan terus menerus menembaknya secara akurat berdasarkan lokasi dan jarak musuh.
- e. Fungsi kelayakan: Memastikan aksi yang dipilih tidak menyebabkan bot kehilangan banyak energi atau mati lebih cepat sehingga bot bertahan lebih lama.
- f. Fungsi obyektif: Memaksimalkan skor dengan cara mendapatkan poin dari penembakan dengan menarget dan mengikuti (tracking) lokasi musuh yang sama.

2. Analisis Efisiensi Solusi

Selama game berjalan, bot mendeteksi sejumlah bot lawan, misalkan bot lawan berjumlah sebanyak n . Pada kasus terburuk, yaitu sampai bot ini mati, semua bot lain tetap hidup sehingga harus terus-menerus melakukan scanning pada n bot untuk menyimpan data bot musuh. Hal ini terus dilakukan pada setiap turn-nya, jika turn-nya sejumlah n , maka iterasi akan dilakukan sebanyak $n*n$ untuk setiap rondonya. Karena pada tugas besar ini, jumlah ronde adalah 10, maka terdapat $10*n*n$ iterasi. Maka, kompleksitas waktu dari algoritma ini adalah sebesar $O(n^2)$.

3. Analisis Efektivitas Solusi

Algoritma ini cukup efektif untuk dapat menyerang dengan menarget satu lawan karena radar tidak perlu melakukan pemindaian secara terus menerus ke seluruh arena, sehingga pendeteksian dan penembakan bisa lebih efektif. Namun jika bot bertarung di arena dengan banyak lawan, kemungkinan bertahan bot ini akan kecil karena hanya berfokus pada satu bot pada waktu tertentu saja dan baru melakukan pemindaian ulang atau mengubah target jika target lawan sebelumnya sudah berada di luar jangkauan pemindaian radar yang mungkin saja terjadi ketika bot melakukan pertahanan diri (menghindar).

3.1.2. Greedy by Avoiding Damage: Dreadfang

Algoritma ini berfokus pada "Greedy by Avoiding Damage" untuk memaksimalkan skor dengan bertahan selama mungkin. Bot terus bergerak dalam pola berputar untuk menghindari tembakan lawan, menyesuaikan arah belok jika dekat dengan tembok, dan meningkatkan kecepatan saat terkena serangan. Bot baru akan menyerang dengan kekuatan tembakan dan tabrakan maksimal ketika bot lawan yang hidup tersisa satu. Dengan strategi ini, bot berusaha mengamankan Survival Score dari setiap musuh yang mati dan Last Survival Bonus jika berhasil menjadi yang terakhir bertahan. Selain itu, respons cepat terhadap tembakan lawan membantu mengurangi damage yang diterima, memastikan bot tetap hidup lebih lama dan memiliki peluang lebih besar untuk mendapatkan skor tinggi.

1. Mapping Elemen Greedy

- Himpunan kandidat: Aksi-aksi yang dapat diambil bot dalam setiap langkah, seperti menghindar, menyerang, bertahan, atau menabrak musuh.
- Himpunan solusi: Serangkaian aksi yang dipilih dalam pertempuran berdasarkan kondisi tertentu.
- Fungsi solusi: Memeriksa apakah bot masih bertahan dan berhasil memperoleh skor setinggi mungkin.
- Fungsi seleksi: Memilih aksi yang meminimalkan kemungkinan terkena damage, seperti berputar secara terus-menerus dan mengatur jarak dari musuh. Jika hanya tersisa satu musuh atau bot ditabrak, strategi berubah menjadi menyerang dengan daya serang yang sesuai berdasarkan energi musuh.

- e. Fungsi kelayakan: Memastikan aksi yang dipilih tidak menyebabkan bot kehilangan banyak energi atau mati lebih cepat sehingga bot bertahan lebih lama
- f. Fungsi obyektif: Memaksimalkan skor dengan cara bertahan selama mungkin dan mengambil kesempatan menyerang ketika risikonya rendah.

2. Analisis Efisiensi Solusi

Selama game berjalan, bot mendeteksi sejumlah bot lawan, misalkan bot lawan berjumlah sebanyak n . Worst-casenya, sampai bot ini mati, semua bot lain tetap hidup sehingga harus terus-menerus melakukan scanning pada n bot untuk menyimpan data bot musuh. Hal ini terus dilakukan pada setiap turn-nya, jika turn-nya sejumlah n , maka iterasi akan dilakukan sebanyak $n*n$ untuk setiap rondanya. Karena pada tugas besar ini, jumlah ronde adalah 10, maka terdapat $10*n*n$ iterasi. Maka, kompleksitas waktu dari algoritma ini adalah sebesar $O(n^2)$.

3. Analisis Efektivitas Solusi

Algoritma ini cukup efektif untuk membuat bot bertahan lebih lama dalam ronde pertandingan jika melawan banyak bot secara sekaligus. Dengan algoritma ini, bot lain akan menyerang satu sama lain ataupun menghabiskan energinya dengan terus menembak ke arah bot ini. Dari segi survivability, bot ini memiliki kemungkinan survival yang besar. Namun, karena bot ini lebih berfokus pada strategi defensif, poin yang dapat diperoleh bot menjadi lebih kecil. Hal ini disebabkan perolehan poin dari strategi ofensif jauh lebih besar daripada strategi defensif. Jadi, bot ini cukup efektif untuk melawan bot dengan jumlah yang banyak yang mana kemampuan survival sangat dibutuhkan.

3.1.3. Greedy by Spinning, Positioning, and Evaluating: KingBot

Bot ini merupakan penyempurnaan dari Spin Bot yang terdapat pada sampel. Pada Spin Bot, dilakukan evaluasi strategi algoritma berupa penyesuaian posisi memutar, penyesuaian reaksi terhadap terkena tembakan, dan terhadap tabrakan dengan bot lain.

1. Mapping Elemen Greedy

- a. Himpunan kandidat: Aksi-aksi yang dapat diambil bot dalam setiap langkah, seperti menghindar dan menembak musuh.
- b. Himpunan solusi: Serangkaian aksi yang dipilih dalam pertempuran berdasarkan kondisi tertentu.
- c. Fungsi solusi: Memeriksa apakah bot masih bertahan dan berhasil memperoleh skor setinggi mungkin.
- d. Fungsi seleksi: Memilih aksi yang memaksimalkan posisi bot agar dapat mendeteksi musuh dengan maksimal, meminimalkan kemungkinan terkena

tembakan dari sumber yang sama lebih dari sekali, dan memaksimalkan penabrakan ke bot lain.

- e. Fungsi kelayakan: Memastikan aksi yang dipilih tidak menyebabkan bot kehilangan banyak energi atau mati lebih cepat sehingga bot bertahan lebih lama.
- f. Fungsi obyektif: Memaksimalkan skor dengan cara bertahan di arena dengan menghindari tembakan lawan sambil menembak lawan.

2. Analisis Efisiensi Solusi

Bot ini menggunakan algoritma yang sangat efisien. Di luar looping yang dilakukan dalam `IsRunning()`, komputer mengeksekusi fungsi-fungsi yang sederhana untuk setiap event yang terjadi dengan kompleksitas $O(1)$.

3. Analisis Efektivitas Solusi

Algoritma ini efektif untuk melakukan penyerangan dengan penyesuaian lokasi yang optimal pada arena. Pergerakan bot yang terus berputar juga efektif untuk mendeteksi lawan di seluruh bagian arena. Selain itu, keunggulan lain dari strategi ini adalah kemampuannya untuk menghindari tembakan lawan dengan pola gerakan yang sulit diprediksi. Dengan terus bergerak dan melakukan rotasi besar, bot mempersulit musuh untuk melakukan tembakan yang akurat. Bot ini lebih fokus pada strategi menghindar dan mempertahankan posisi sehingga perolehan skor dari menembak lawan cenderung lebih sedikit dibandingkan bot yang lebih agresif.

Maka dari itu, bot ini sangat efektif dalam menghadapi banyak lawan karena dapat bertahan lebih lama dibandingkan bot lain yang lebih agresif tetapi kehabisan energi lebih cepat. Namun, dalam aspek perolehan skor, strategi ini kurang optimal karena tidak cukup agresif dalam menyerang lawan secara aktif.

3.1.4. Greedy by Dodging and Attack: Syzygy

Bot ini dijalankan dengan algoritma Greedy by Dodging and Attack untuk dapat memaksimalkan poin akhir. Bot bergerak secara random untuk menghindari serangan lawan dan bergerak lagi jika mendeteksi energi bot berkurang yang mungkin terjadi karena bot menembakkan peluru. Radar bot berputar dan memindai seluruh arena dengan terus menerus dan menembak ke arah lawan jika bot mendeteksi bot lain.

1. Mapping Elemen Greedy

- a. Himpunan kandidat: Aksi-aksi yang dapat diambil bot dalam setiap langkah, seperti menghindar dan menembak musuh.
- b. Himpunan solusi: Serangkaian aksi yang dipilih dalam pertempuran berdasarkan kondisi tertentu.

- c. Fungsi solusi: Memeriksa apakah bot masih bertahan dan berhasil memperoleh skor setinggi mungkin.
- d. Fungsi seleksi: Memilih aksi yang meminimalkan bot terkena damage, yaitu dengan bergerak secara acak dan menghindari jika energi bot musuh berkurang, dengan asumsi energi bot musuh berkurang karena menembakkan peluru, atau dengan menembak jika mendeteksi musuh.
- e. Fungsi kelayakan: Memastikan aksi yang dipilih tidak menyebabkan bot kehilangan banyak energi atau mati lebih cepat sehingga bot bertahan lebih lama.
- f. Fungsi obyektif: Memaksimalkan skor dengan cara bertahan di arena dengan menghindari tembakan lawan sambil menembak lawan.

2. Analisis Efisiensi Solusi

Selama game berjalan, bot mendeteksi sejumlah bot lawan, misalkan bot lawan berjumlah sebanyak n . Pada kasus terburuk, yaitu sampai bot ini mati, semua bot lain tetap hidup sehingga harus terus-menerus melakukan scanning pada n bot untuk menyimpan data bot musuh. Hal ini terus dilakukan pada setiap turn-nya, jika turn-nya sejumlah n , maka iterasi akan dilakukan sebanyak $n*n$ untuk setiap rondanya. Karena pada tugas besar ini, jumlah ronde adalah 10, maka terdapat $10*n*n$ iterasi. Maka, kompleksitas waktu dari algoritma ini adalah sebesar $O(n^2)$.

3. Analisis Efektivitas Solusi

Algoritma ini cukup efektif untuk dapat bertahan pada arena karena gerakannya yang acak dan kemampuannya untuk menghindari jika mendeteksi lawan menembakkan peluru. Bot juga menembak ke arah lawan jika bot lain terpindai untuk mendapatkan poin tembakan.

3.2. Strategi Greedy yang Dipilih

Strategi yang dipilih untuk bot utama adalah strategi Greedy by Dodging and Attack: Syzygy. Strategi ini dipilih karena kemampuannya untuk dapat menghindari dari serangan lawan dengan gerakannya yang acak dan kemampuannya mendeteksi kemungkinan serangan dari musuh. Oleh karena itu, bot dengan strategi ini memiliki kemungkinan terbesar untuk dapat menang dalam arena. Selain itu, bot ini juga melakukan penyerangan terhadap bot lain. Kombinasi ini membuat bot ini memiliki posisi kuat di arena untuk mengumpulkan skor terbanyak.

BAB IV

IMPLEMENTASI DAN PENGUJIAN

4.1. Implementasi Alternatif Solusi Greedy

4.1.1. Greedy by Enemy Targetting: GritBot

```
function Main()
{ Memulai eksekusi bot }
Deklarasi:
    bot : GritBot

Algoritma:
    bot ← instance GritBot
    Jalankan bot
endfunction

function Run()
{ Loop utama bot untuk bergerak dan menembak }
Algoritma:
    InitializeRound()
    while IsRunning do
        HandleMove()
        HandleGun()
    endwhile
endfunction

function HandleMove()
{ Mengatur pergerakan bot }
Algoritma:
    Maju(5)
    if setTarget then
        Maju(100)
    endif
endfunction

function HandleGun()
{ Mengatur gerakan senjata bot }
Algoritma:
    if not setTarget then
        PutarSenjataKiri(360 * turnDirection)
    else
        PutarSenjataKanan(20 * turnDirection)
        PutarSenjataKiri(45 * turnDirection)
    endif
endfunction

function OnScannedBot(e)
{ Menangani musuh yang terdeteksi }
Deklarasi:
    scannedEnemy : Enemy
```

```

    enemyDistance, enemyDirection : real

Algoritma:
    setTarget ← true

    enemyDistance ← JarakKe(e.X, e.Y)
    enemyDirection ← ArahKe(e.X, e.Y)

    scannedEnemy ← musuh dengan id e.ScannedBotId dalam daftar_musuh
    if scannedEnemy = null then
        Tambahkan musuh baru ke daftar_musuh
    else
        if scannedEnemy.IsEnergyEnemyDec(e.Energy) then
            DodgeMovement()
        endif
    endif
    FireTarget(enemyDistance, enemyDirection)
endfunction

function FireTarget(distance, enemyDirection)
{ Menembak musuh berdasarkan jarak dan arah }
Deklarasi:
    gunBearing : real

Algoritma:
    gunBearing ← NormalisasiSudut(enemyDirection - ArahSenjata)
    PutarSenjataKiri(gunBearing)

    if distance > 200 or Energi < 20 then
        for i ← 1 to 7 do
            Tembak(1)
        endfor
    else if distance > 100 then
        Tembak(2)
        Tembak(2)
    else
        Tembak(3)
    endif
endfunction

function TurnToFaceTarget(x, y)
{ Memutar bot menghadap target }
Deklarasi:
    bearing : real

Algoritma:
    bearing ← ArahKe(x, y)
    turnDirection ← if bearing ≥ 0 then 1 else -1
    BelokKiri(bearing)
endfunction

function DodgeMovement()
{ Menghindari serangan dengan bergerak menjauh }
Algoritma:
    BelokKanan(45)
    Maju(50)
endfunction

```



```

function OnBotDeath(e)
{ Menangani musuh yang mati }
Algoritma:
    if e.VictimId = targetID then
        setTarget ← false
    endif
endfunction

function OnHitBot(e)
{ Menangani tabrakan dengan bot lain }
Algoritma:
    BelokKanan(90)
    Maju(30)
endfunction

function OnHitWall(e)
{ Menangani tabrakan dengan dinding }
Algoritma:
    BelokKanan(90)
    Mundur(10)
endfunction

function InitializeRound()
{ Inisialisasi warna dan variabel sebelum pertandingan dimulai }
Algoritma:
    Atur warna bot
    setTarget ← false
    targetID ← -1
endfunction

class Enemy
{ Menyimpan informasi musuh yang terdeteksi }
Deklarasi:
    id : integer
    energy, x, y : real

Konstruktor:
    function Enemy(id, energy, x, y)
        this.id ← id
        this.energy ← energy
        this.x ← x
        this.y ← y
    endfunction

    function Update(energy, x, y)
    { Memperbarui informasi musuh }
    Algoritma:
        this.energy ← energy
        this.x ← x
        this.y ← y
    endfunction

    function IsEnergyEnemyDec(curEnergy)
    { Memeriksa apakah energi musuh berkurang }
    Algoritma:
        return curEnergy < this.energy
    endfunction
endclass

```

4.1.2. Greedy by Avoiding Damage: Dreadfang

```
function Main()
{ Memulai eksekusi bot }
Deklarasi:
    bot : instance Bot

Algoritma:
    bot ← BuatInstanceBot()
    bot.Jalankan()
endfunction

function Run()
{ Loop utama bot untuk menghindari damage }
Deklarasi:
    sudut_belokan : integer

Algoritma:
    AturWarnaBot()

    while IsRunning do
    { Menentukan sudut belokan berdasarkan jumlah musuh dan posisi
    terhadap dinding }
        if jumlah_musuh > 10 then
            if IsNearWall() then
                sudut_belokan ← 20
            else
                sudut_belokan ← 40
            endif
        else if jumlah_musuh > 5 then
            if IsNearWall() then
                sudut_belokan ← 40
            else
                sudut_belokan ← 80
            endif
        else
            if IsNearWall() then
                sudut_belokan ← 60
            else
                sudut_belokan ← 120
            endif
        endif

        BelokKiri(sudut_belokan * arah_belokan)
        Maju(100)
        KecepatanTarget ← 5
        PutarRadar(MaksRadarPutar)
    endwhile
endfunction

function OnScannedBot(e)
{ Menangani musuh yang terdeteksi }
Deklarasi:
    target : integer
```

```

Algoritma:
    PerbaruiEnergiMusuh(e.id, e.energi)

    { Pilih target dengan energi terendah }
    if target_id = -1 or target_id tidak ada dalam daftar_musuh or e.energi
    daftar_musuh[target_id] then
        target_id ← e.id
    endif

    { Jika hanya tersisa satu musuh, fokus menyerang }
    if target_id ≠ -1 and target_id ada dalam daftar_musuh and jumlah_musuh
    then
        KecepatanTarget ← 2
        target ← daftar_musuh[target_id]
        TurnToFaceTarget(e.x, e.y)

        if JarakKe(e.x, e.y) > 300 then
            Tembak(0.2)
        else if target > 200 then
            Tembak(0.5)
        else if target > 150 then
            Tembak(1)
        else if target > 80 then
            Tembak(2)
        else
            Tembak(3)
        endif
    endif
endfunction

function OnBotDeath(e)
{ Menghapus musuh yang sudah mati dari daftar }
Deklarasi:
    id : integer

Algoritma:
    id ← e.id
    Hapus daftar_musuh[id]
endfunction

function OnHitByBullet(e)
{ Menghindari peluru dengan mengubah arah dan kecepatan }
Algoritma:
    KecepatanTarget ← 8
    BelokKiri(90 * arah_belokan)
    Maju(50)
endfunction

function OnHitWall(e)
{ Menyesuaikan arah setelah menabrak dinding }
Algoritma:
    arah_belokan ← -arah_belokan
    KecepatanTarget ← -KecepatanTarget
    BelokKiri(90)
    Maju(50)
endfunction

```

```

function OnHitBot(e)
{ Menyerang balik jika bertabrakan dengan musuh }
Algoritma:
    TurnToFaceTarget(e.x, e.y)
    { Tabrak musuh lebih keras lagi dan serang dengan kekuatan tembakan yang
    besar jika energi musuh banyak }
    if e.energi > 16 then
        Tembak(3)
    else if e.energi > 10 then
        Tembak(2)
    else if e.energi > 4 then
        Tembak(1)
    else if e.energi > 2 then
        Tembak(0.5)
    else if e.energi > 0.4 then
        Tembak(0.1)
    endif

    Maju(40)
endfunction

function TurnToFaceTarget(x, y)
{ Memutar turret ke arah musuh }
Deklarasi:
    bearing : real

Algoritma:
    bearing ← ArahKe(x, y)
    arah_belokan ← if bearing ≥ 0 then 1 else -1
    BelokKiri(bearing)
endfunction

function IsNearWall()
{ Memeriksa apakah bot dekat dengan dinding arena }
Algoritma:
    return (x < batas_aman or x > lebar_arena - batas_aman or
            y < batas_aman or y > tinggi_arena - batas_aman)
endfunction

```

4.1.3. Greedy by Spinning, Positioning, and Evaluating: KingBot

```

function Main()
{ Memulai eksekusi bot }
Deklarasi:
    bot : KingBot

Algoritma:
    bot ← instance KingBot
    Jalankan bot
endfunction

function Run()
{ Loop utama bot untuk pergerakan dan strategi }
Deklarasi:
    i : integer

```

```

    div : integer ← 2

Algoritma:
    Atur warna bot
    { Pindah ke titik tengah arena }
    BelokKiri(ArahKe(ArenaWidth/div, ArenaHeight/div))
    Maju(JarakKe(ArenaWidth/div, ArenaHeight/div))

    i ← 0
    while IsRunning do
        if i mod 200 = 0 then
            BelokKiri(ArahKe(ArenaWidth/div, ArenaHeight/div))
            Maju(JarakKe(ArenaWidth/div, ArenaHeight/div))
        else
            AturBelokKiri(50_000)
            MaxSpeed ← 5
            Maju(50_000)
        endif
        i ← i + 1
    endwhile
endfunction

function OnHitBot(e)
{ Menangani tabrakan dengan bot lain }
Algoritma:
    Mundur(50)
    BelokKiri(ArahKe(ArenaWidth/div, ArenaHeight/div))
    Maju(90)
endfunction

function OnHitByBullet(e)
{ Menangani bot yang terkena peluru }
Algoritma:
    Maju(50)
endfunction

function OnScannedBot(e)
{ Menangani musuh yang terdeteksi }
Algoritma:
    Berhenti()
    Tembak(3)
    Lanjutkan()
endfunction

```

4.1.4. Greedy by Dodging and Attack: Syzygy

```

function Main()
{ Memulai eksekusi bot }
Deklarasi:
    bot : Syzygy

Algoritma:
    bot ← instance dari Syzygy
    Jalankan bot
endfunction

```

```

function Run()
{ Loop utama bot untuk pergerakan dan pengendalian radar }
Algoritma:
    InitializeRound()
    while IsRunning do
        GunTurnRate  $\leftarrow$  MaxGunTurnRate
        HandleRadar()
        HandleMove()
    endwhile
endfunction

function HandleRadar()
{ Menggerakkan radar untuk mendeteksi musuh }
Algoritma:
    PutarSenjataKanan(360)
endfunction

function HandleMove()
{ Menggerakkan bot dengan pola acak }
Deklarasi:
    randomForwBack : integer
    randomTurn : integer

Algoritma:
    randomForwBack  $\leftarrow$  Nilai acak antara -75 hingga 200
    randomTurn  $\leftarrow$  Nilai acak antara -90 hingga 90
    Maju(randomForwBack)
    BelokKanan(randomTurn)
endfunction

function OnScannedBot(e)
{ Menangani musuh yang terdeteksi }
Deklarasi:
    scannedEnemy : Enemy
    enemyDistance, enemyGunBearing : real

Algoritma:
    enemyDistance  $\leftarrow$  JarakKe(e.X, e.Y)
    enemyGunBearing  $\leftarrow$  ArahSenjataKe(e.X, e.Y)

    scannedEnemy  $\leftarrow$  musuh dengan id e.ScannedBotId dalam daftar_musuh
    if scannedEnemy = null then
        Tambahkan musuh baru ke daftar_musuh
    else
        if scannedEnemy.IsEnergyEnemyDec(e.Energy) then
            DodgeMovement()
        endif
    endif

    FireTarget(enemyDistance, enemyGunBearing)
endfunction

function DodgeMovement()
{ Menghindari serangan dengan bergerak acak }
Algoritma:
    BelokKanan(45)
    Maju(50)
endfunction

```

```

function OnHitByBullet(e)
{ Menangani bot yang terkena tembakan }
Algoritma:
    BelokKanan(45)
    Maju(50)
endfunction

function OnHitWall(e)
{ Menangani bot yang menabrak dinding }
Algoritma:
    Mundur(20)
    BelokKanan(90)
endfunction

function OnHitBot(e)
{ Menangani bot yang bertabrakan dengan musuh }
Algoritma:
    Mundur(10)
    BelokKiri(90)
endfunction

function FireTarget(distance, gunBearing)
{ Menentukan kekuatan tembakan berdasarkan jarak musuh }
Algoritma:
    if distance > 200 or Energi < 20 then
        Tembak(1)
    else if distance > 100 then
        Tembak(2)
    else
        Tembak(3)
    endif
endfunction

function InitializeRound()
{ Mengatur warna dan menghapus daftar musuh sebelum pertandingan }
Algoritma:
    Atur warna bot
    Hapus semua musuh dari daftar_musuh
endfunction

class Enemy
{ Menyimpan informasi musuh yang terdeteksi }
Deklarasi:
    id : integer
    energy, x, y : real

Konstruktor:
    function Enemy(id, energy, x, y)
        this.id ← id
        this.energy ← energy
        this.x ← x
        this.y ← y
    endfunction

    function Update(energy, x, y)
    { Memperbarui informasi musuh }
    Algoritma:

```

```

        this.energy ← energy
        this.x ← x
        this.y ← y
    endfunction

    function IsEnergyEnemyDec(curEnergy)
    { Periksa apakah energi musuh berkurang }
    Algoritma:
        return curEnergy < this.energy
    endfunction
endclass

```

4.2. Penjelasan Solusi Greedy yang Dipilih

Bot yang dipilih sebagai bot utama adalah Syzygy. Struktur data utama yang digunakan dalam bot ini adalah list, yang berfungsi untuk menyimpan daftar musuh yang telah terdeteksi selama permainan. Dengan menggunakan list ini, bot dapat memantau perubahan energi musuh untuk menentukan apakah musuh sedang diserang oleh bot lain. Jika energi musuh berkurang, bot akan segera menghindar untuk mengurangi risiko terkena tembakan. Selain itu, list ini juga memudahkan bot dalam mengambil keputusan secara greedy, baik untuk menyerang atau menghindar. Setiap musuh yang terdeteksi akan disimpan dalam objek Enemy, yang memiliki atribut id (sebagai identitas unik musuh), energy (jumlah energi musuh), serta x dan y (posisi musuh di arena). Fungsi Update(energy, x, y) digunakan untuk memperbarui informasi musuh setiap kali terdeteksi kembali. Selain itu, metode IsEnergyEnemyDec(curEnergy) memungkinkan bot untuk mengecek apakah energi musuh berkurang, yang menandakan bahwa musuh sedang dalam pertempuran.

Selain memanfaatkan struktur data list dan juga menggunakan class, bot ini juga memiliki berbagai fungsi dan prosedur yang mendukung implementasi solusi greedy, yaitu:

1. Run()

Fungsi ini merupakan loop utama dari bot yang terus berjalan selama permainan berlangsung. Pertama, bot akan menjalankan InitializeRound() untuk mengatur ulang warna bot serta mengosongkan daftar musuh. Selanjutnya, bot akan memasuki loop utama, di mana ia akan mengatur pergerakan radar, mendeteksi musuh, serta bergerak secara acak.

Strategi greedy terlihat dalam penggunaan GunTurnRate = MaxGunTurnRate, yang memastikan radar selalu aktif berputar untuk mencari musuh. Fungsi HandleRadar() digunakan untuk menggerakkan radar, sedangkan HandleMove() membuat bot bergerak secara acak, yang membuatnya lebih sulit ditembak oleh lawan.

2. OnScannedBot(ScannedBotEvent e)

Ketika bot mendeteksi musuh menggunakan radar, fungsi OnScannedBot(e) akan dipanggil. Fungsi ini pertama-tama mencari apakah musuh yang terdeteksi sudah ada

dalam daftar musuh (enemies). Jika belum ada, musuh tersebut akan ditambahkan ke daftar. Jika musuh sudah ada, bot akan mengecek apakah energi musuh berkurang.

Jika energi musuh berkurang (`IsEnergyEnemyDec(e.Energy)` bernilai true), itu berarti musuh sedang dalam pertempuran dan ada kemungkinan bot juga akan menjadi target. Dalam situasi ini, bot akan langsung menghindar menggunakan `DodgeMovement()`, yang memutar bot 45 derajat ke kanan dan maju 50 unit untuk keluar dari posisi berbahaya. Setelah itu, bot akan menyerang menggunakan `FireTarget(distance, gunBearing)`, yang menentukan seberapa kuat bot akan menembak musuh berdasarkan jaraknya.

3. DodgeMovement()

Ketika bot mendeteksi musuh menggunakan radar, fungsi `OnScannedBot(e)` akan dipanggil. Fungsi ini akan mencari apakah musuh yang terdeteksi sudah ada dalam daftar musuh (enemies). Jika belum ada, musuh tersebut akan ditambahkan ke daftar. Jika musuh sudah ada, bot akan mengecek apakah energi musuh berkurang.

Jika energi musuh berkurang (`IsEnergyEnemyDec(e.Energy)` bernilai true), itu berarti musuh sedang dalam pertempuran dan ada kemungkinan bot juga akan menjadi target. Dalam situasi ini, bot akan langsung menghindar menggunakan `DodgeMovement()`, yang memutar bot 45 derajat ke kanan dan maju 50 unit untuk keluar dari posisi berbahaya. Setelah itu, bot akan menyerang menggunakan `FireTarget(distance, gunBearing)`, yang menentukan seberapa kuat bot akan menembak musuh berdasarkan jaraknya.

4. HandleMove()

Fungsi `HandleMove()` bertujuan untuk mengubah posisi bot secara acak, sehingga sulit bagi musuh untuk memprediksi pergerakannya. Bot akan memilih angka acak antara -75 hingga 200 untuk menentukan apakah ia akan maju atau mundur, serta angka acak antara -90 hingga 90 untuk menentukan sudut belokan.

Strategi greedy dalam fungsi ini terletak pada pemilihan langkah terbaik untuk menghindari serangan dalam satu langkah waktu, tanpa mempertimbangkan pergerakan jangka panjang. Dengan bergerak secara acak, bot memanfaatkan kemungkinan terbaik untuk tetap aman dan tidak menjadi target mudah.

5. FireTarget(distance, gunBearing)

Fungsi `FireTarget()` bertanggung jawab dalam menentukan kekuatan tembakan bot berdasarkan jarak musuh. Jika musuh berada jauh (>200) atau energi bot rendah (<20), bot hanya akan menembak dengan kekuatan 1 untuk menghemat energi. Jika jarak musuh sedang (100-200), bot akan menembak dengan kekuatan 2. Namun, jika musuh sangat dekat (<100), bot akan menembak dengan kekuatan maksimal (3) untuk memastikan serangan mengenai target dan memberikan damage tinggi.

Strategi greedy dalam fungsi ini terletak pada pemanfaatan energi secara efisien, di mana bot hanya menggunakan energi besar ketika memiliki peluang tinggi untuk memberikan damage besar kepada musuh.

6. OnHitByBullet(e)

Saat bot terkena tembakan, fungsi OnHitByBullet(e) akan langsung memindahkannya dari posisi sebelumnya. Dengan memutar bot 45 derajat ke kanan dan maju 50 unit, bot akan segera keluar dari jalur tembakan, mengurangi kemungkinan terkena serangan beruntun.

7. OnHitBot(e)

Fungsi OnHitBot(e) menangani kasus di mana bot bertabrakan dengan bot lain. Dalam situasi ini, bot akan mundur 10 unit dan berbelok 90 derajat ke kiri, agar tidak terjebak dalam posisi yang bisa dimanfaatkan lawan.

8. OnHitWall(e)

Fungsi OnHitWall(e) menangani kasus ketika bot menabrak dinding arena. Dalam kondisi ini, bot akan mundur 20 unit dan berbelok 90 derajat ke kanan, untuk kembali ke area yang lebih terbuka dan tidak terjebak di sudut arena.

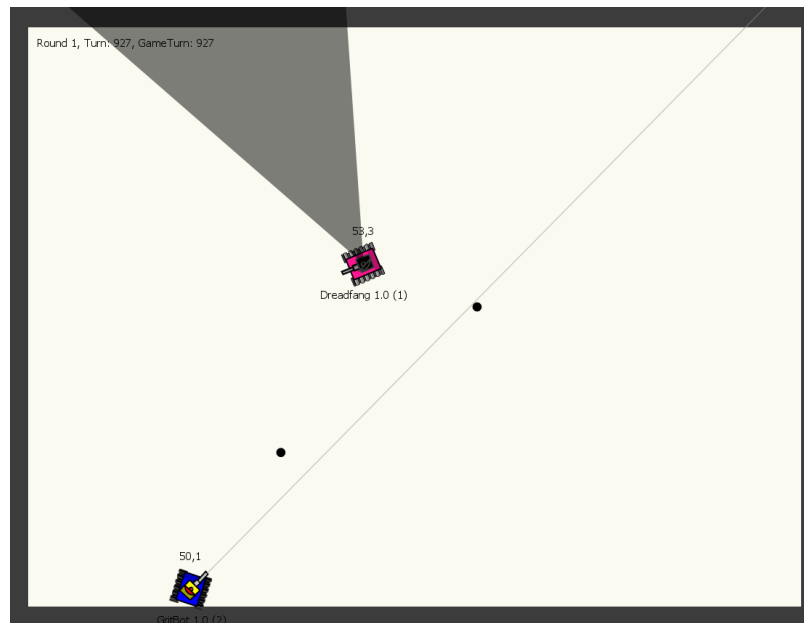
9. InitializeRound()

Sebelum permainan dimulai, bot akan menjalankan fungsi InitializeRound(). Fungsi ini digunakan untuk mengatur warna bot agar lebih mudah dikenali dalam pertandingan, serta mengosongkan daftar musuh dari ronde sebelumnya.

4.3. Pengujian Bot

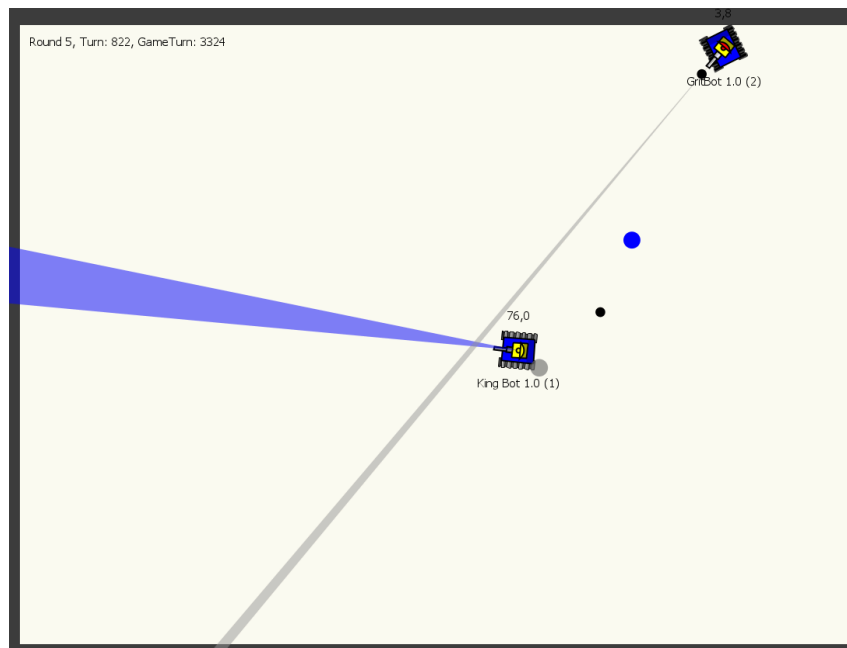
Berikut adalah hasil pengujian bot-bot yang dibuat.

4.3.1 GritBot vs Dreadfang



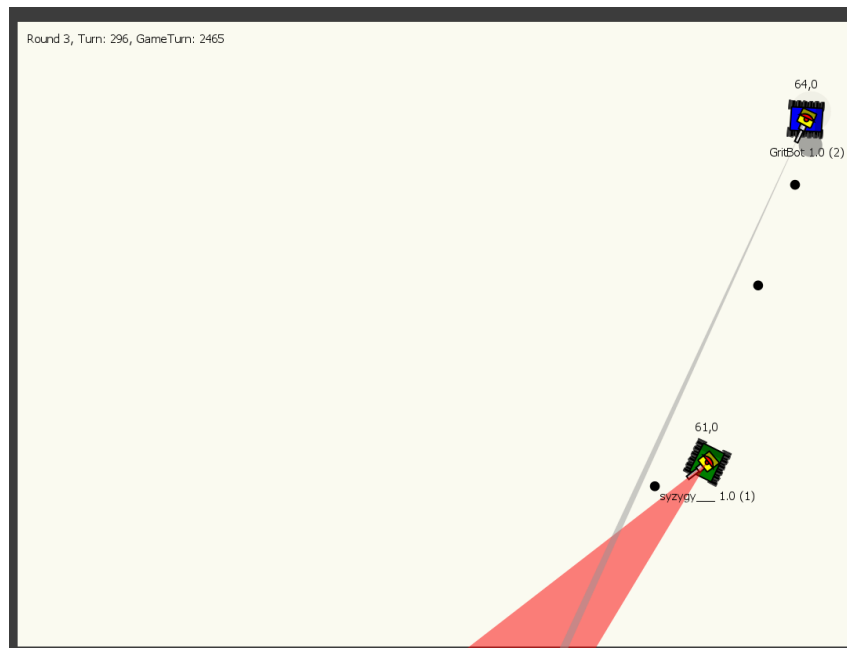
Results for 10 rounds												
Rank	Name	Total Score	Survival	Surv. Bonus	Bullet Dmg.	Bullet Bo...	Ram Dmg.	Ram Bonus	1sts	2nds	3rds	
1	GritBot 1.0	508	150	30	276	36	16	0	4	0	0	
2	Dreadfang 1.0	58	0	0	48	0	10	0	0	4	0	

4.3.2 GritBot vs KingBot



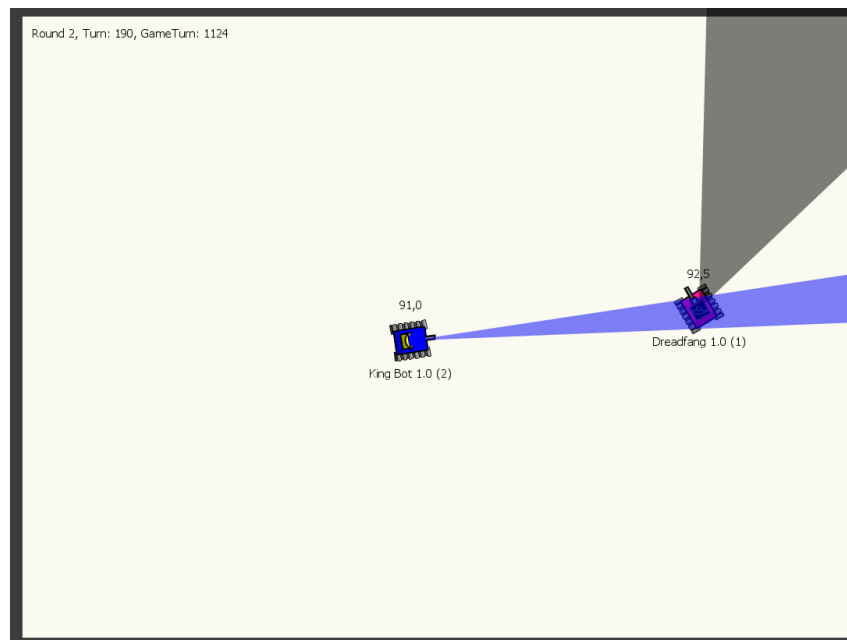
Results for 10 rounds												
Rank	Name	Total Score	Survival	Surv. Bonus	Bullet Dmg.	Bullet Bo...	Ram Dmg.	Ram Bonus	1sts	2nds	3rds	
1	King Bot 1.0	1118	300	60	640	115	2	0	7	0	0	
2	GritBot 1.0	173	0	0	172	0	1	0	0	7	0	

4.3.3 GritBot vs Syzygy



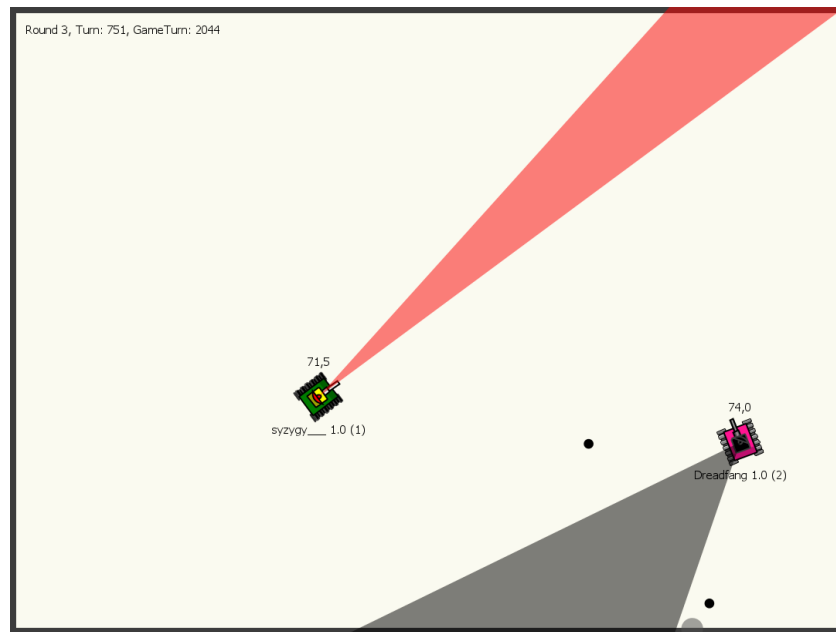
Results for 10 rounds											
Rank	Name	Total Score	Survival	Surv. Bonus	Bullet Dmg.	Bullet Bo...	Ram Dmg.	Ram Bonus	1sts	2nds	3rds
1	syzygy__ 1.0	789	200	40	492	52	5	0	5	2	0
2	GritBot 1.0	440	100	20	276	39	4	0	2	5	0

4.3.4 Dreadfang vs KingBot



Results for 10 rounds											
Rank	Name	Total Score	Survival	Surv. Bonus	Bullet Dmg.	Bullet Bo...	Ram Dmg.	Ram Bonus	1sts	2nds	3rds
1	King Bot 1.0	563	150	30	320	49	13	0	4	1	0
2	Dreadfang 1.0	165	50	10	80	0	25	0	1	4	0

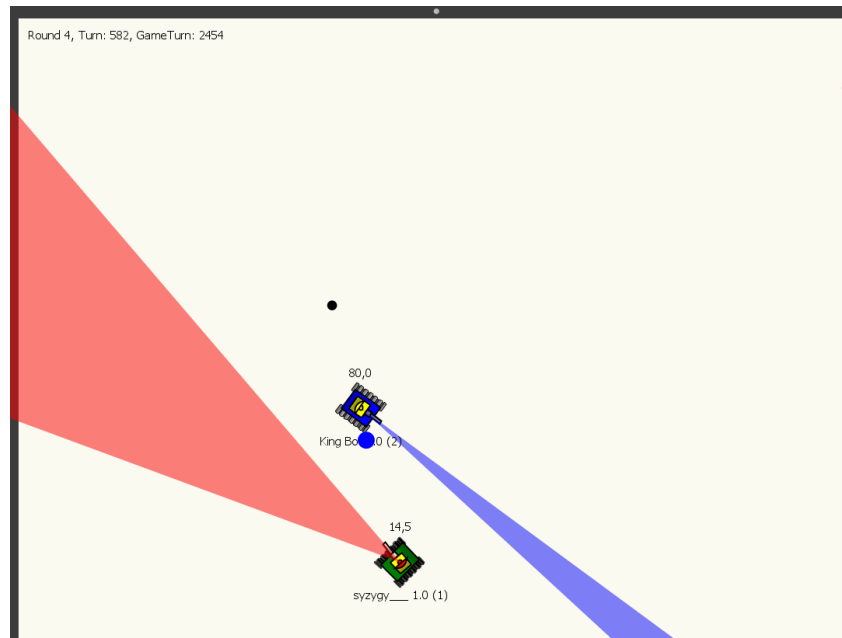
4.3.5 Dreadfang vs Syzygy



Results for 10 rounds

Rank	Name	Total Score	Survival	Surv. Bonus	Bullet Dmg.	Bullet Bo...	Ram Dmg.	Ram Bonus	1sts	2nds	3rds
1	syzygy___ 1.0	950	250	50	510	76	32	31	6	0	0
2	Dreadfang 1.0	95	0	0	64	0	31	0	0	6	0

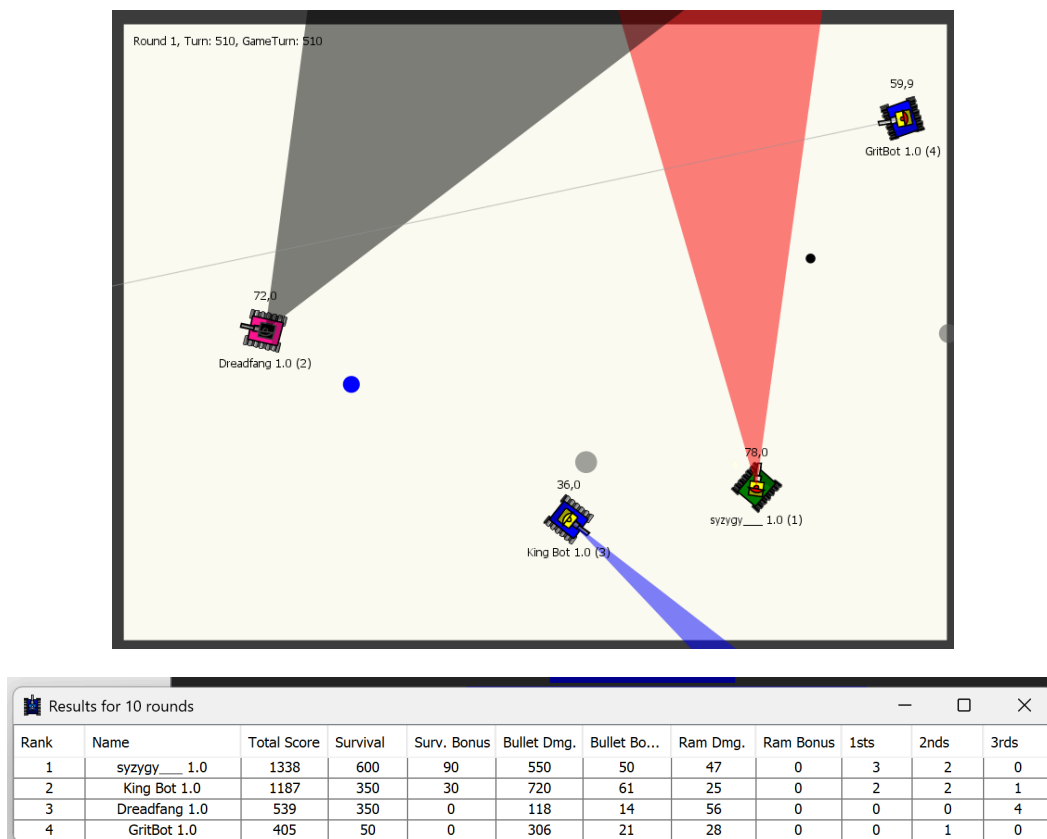
4.3.6 Kingbot vs Syzygy



Results for 10 rounds

Rank	Name	Total Score	Survival	Surv. Bonus	Bullet Dmg.	Bullet Bo...	Ram Dmg.	Ram Bonus	1sts	2nds	3rds
1	King Bot 1.0	969	250	50	576	81	12	0	6	0	0
2	syzygy___ 1.0	377	0	0	358	0	19	0	0	6	0

4.3.7 Semua Bot



4.4. Analisis Hasil Pengujian Bot

Pada pengujian pertama, yaitu GritBot vs Dreadfang, GritBot berhasil memenangkan seluruh ronde. Hal ini kemungkinan besar terjadi karena strategi GritBot yang menarget dan menyerang bot Dreadfang sambil melakukan *tracking*. Strategi ini dapat menyerang strategi Dreadfang yang berfokus pada melakukan pertahanan.

Dari hasil pengujian GritBot vs KingBot, didapatkan bahwa KingBot memenangkan pertandingan. Penyerangan KingBot dengan energi peluru yang besar membuat damage yang besar pula terhadap GritBot. Hal ini tentunya akan memperbesar *gap* poin pada arena satu lawan satu. Gerakan KingBot yang memutar juga membuat GritBot terkadang kesulitan dalam melakukan *tracking*.

Pada pertandingan GritBot vs Syzygy, diperoleh hasil Syzygy sebagai pemenang 5 dari 7 ronde yang ada. Gerakan Syzygy yang acak dan tidak menentu tentunya mempersulit Gritbot dalam melakukan *tracking* lokasi dan melakukan penyerangan. Hal ini kemungkinan besar menjadi faktor kemenangan Syzygy pada pertandingan ini.

KingBot memenangkan pertandingan pengujian melawan Dreadfang. Energi peluru KingBot yang besar diduga menjadi faktor utama kemenangannya. Strategi KingBot yang merupakan penyempurnaan dari startegi bot sampel, SpinBot, yang diketahui sebagai salah satu strategi

bot yang paling optimal, membuatnya bisa mendapatkan poin yang optimal dalam pertandingan.

Pada pertandingan keenam, Syzygy memenangkan seluruh ronde melawan Dreadfang. Gerakan acak Syzygy dan kemampuannya untuk menghindari serangan Dreadfang mampu menghasilkan perolehan poin optimal. Pada pertandingan terakhir, KingBot memperoleh poin yang lebih banyak dibanding Syzygy. Hal ini kemungkinan besar terjadi karena strategi KingBot yang melakukan penyerangan pada posisi optimal dengan energi yang besar sehingga mempercepat pertumbuhan *gap* poin dalam pertandingan.

BAB V

KESIMPULAN DAN SARAN

5.1. Kesimpulan

Greedy merupakan paradigma algoritmik yang membangun solusi langkah demi langkah, selalu memilih bagian berikutnya yang menawarkan manfaat paling jelas dan langsung. Pada setiap langkah, harus dibuat keputusan yang terbaik dalam menentukan pilihan sehingga tidak bisa mundur lagi (kembali) ke langkah sebelumnya. Algoritma greedy adalah salah satu pendekatan yang dapat digunakan untuk mencari solusi optimal dari suatu permasalahan. Permasalahan yang ditinjau saat ini adalah memenangkan Robocode Tank Royale dengan mengumpulkan skor sebanyak-banyaknya dibandingkan bot lain dalam pertandingan sepuluh ronde. Tentunya, terdapat banyak pendekatan greedy yang dapat diimplementasikan untuk menyelesaikan persoalan ini. Setiap strategi greedy yang digunakan memiliki kelebihan dan kekurangannya masing-masing. Maka, dari berbagai strategi yang sudah dicoba, ditemukan bot paling optimal, yaitu Syzygy. Bot Syzygy bergerak secara random untuk menghindari serangan lawan dan bergerak lagi jika mendeteksi energi bot berkurang yang mungkin terjadi karena bot menembakkan peluru. Radar bot berputar dan memindai seluruh arena dengan terus menerus dan menembak ke arah lawan jika bot mendeteksi bot lain. Bot ini menggunakan strategi Greedy by Dodging and Attacking.


5.2. Saran

Terdapat beberapa saran yang dapat dilakukan untuk pengembangan lebih lanjut sekaligus untuk evaluasi proses pengerjaan, antara lain:

1. Lebih fokus memikirkan strategi greedy yang lebih sederhana, tetapi efektif.
2. Mengoptimalkan struktur pembagian tugas serta meningkatkan kerjasama tim untuk mengoptimalkan hasil program.
3. Meningkatkan manajemen waktu selama pengerjaan tugas.

LAMPIRAN

Tautan Github: https://github.com/najwakahanifatima/Tubes1_Berkah-Ramadhan

Tautan Video:  Video (sementara) (Link YouTube akan dikirim menyusul ke asisten)

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

- GeeksforGeeks. (t.t.). *Pengenalan algoritma greedy - Tutorial struktur data dan algoritma*. Diakses pada 24 Maret 2025, dari <https://www.geeksforgeeks.org/introduction-to-greedy-algorithm-data-structures-and-algorithm-tutorials/>
- Munir, R. (2025). *Algoritma greedy - Bagian 1*. Diakses pada 24 Maret 2025, dari [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)