

LAPORAN TUGAS BESAR 1
IF2211 STRATEGI ALGORITMA



Kelompok 8 Infinitea

Anella Utari Gunadi 13523078
Nayla Zahira 13523079
Diyah Susan Nugrahani 13523080

SEKOLAH TEKNIK ELEKTRO DAN INFORMATIKA
INSTITUT TEKNOLOGI BANDUNG
2025

DAFTAR ISI

DAFTAR ISI	1
BAB I	3
DESKRIPSI TUGAS	3
BAB II	8
LANDASAN TEORI	8
2.1. Dasar Teori	8
2.2. Cara Kerja Program	8
2.3. Cara Menjalankan Program	9
2.4. Alur Pengembangan Algoritma Greedy pada Program	12
BAB III	13
APLIKASI STRATEGI GREEDY	13
3.1. Alternatif Solusi Greedy	13
3.1.1. Alternatif Solusi 1	13
3.1.1.1. Proses Mapping Alternatif Solusi 1	13
3.1.1.2. Eksplorasi Alternatif Solusi 1	14
3.1.1.3. Analisis Efisiensi dan Efektivitas Alternatif Solusi 1	14
3.1.2. Alternatif Solusi 2	15
3.1.2.1. Proses Mapping Alternatif Solusi 2	15
3.1.2.2. Eksplorasi Alternatif Solusi 2	15
3.1.2.3. Analisis Efisiensi dan Efektivitas Alternatif Solusi 2	16
3.1.3. Alternatif Solusi 3	17
3.1.3.1. Proses Mapping Alternatif Solusi 3	17
3.1.3.2. Eksplorasi Alternatif Solusi 3	18
3.1.3.3. Analisis Efisiensi dan Efektivitas Alternatif Solusi 3	18
3.1.4. Alternatif Solusi 4	19
3.1.4.1. Proses Mapping Alternatif Solusi 4	19
3.1.4.2. Eksplorasi Alternatif Solusi 4	20
3.1.4.3. Analisis Efisiensi dan Efektivitas Alternatif Solusi 4	21
3.2. Strategi Greedy Solusi Utama	21
BAB IV	22
IMPLEMENTASI DAN PENGUJIAN	22
4.1. Implementasi Solusi	22
4.1.1. Implementasi Bot Utama	22
4.1.2. Implementasi Bot Alternatif 1	23
4.1.3. Implementasi Bot Alternatif 2	24
4.1.4. Implementasi Bot Alternatif 3	25

4.2. Penjelasan Program Bot Utama	26
4.2.1 Struktur Data Bot Utama	26
4.2.2 Fungsi dan Prosedur Bot Utama	26
4.3. Hasil Pengujian Bot Utama	27
4.3.1 Bot Utama Melawan Alternatif Bot 1	27
4.3.1.1 Analisis Hasil Pengujian	29
4.3.2 Bot Utama Melawan Alternatif Bot 2	29
4.3.2.1 Analisis Hasil Pengujian Melawan Alternatif Bot 2	30
4.3.3 Bot Utama Melawan Alternatif Bot 3	31
4.3.3.1 Analisis Hasil Pengujian Melawan Alternatif Bot 3	32
4.3.4 Bot Utama Melawan Seluruh Alternate Bot	32
4.3.4.1 Analisis Hasil Pengujian Melawan Seluruh Alternatif Bot	35
BAB V	36
KESIMPULAN DAN SARAN	36
5.1. Kesimpulan	36
5.2. Saran	36
LAMPIRAN	37
DAFTAR PUSTAKA	38

BAB I

DESKRIPSI TUGAS

Robocode adalah permainan pemrograman yang bertujuan untuk membuat kode bot dalam bentuk tank virtual untuk berkompetisi melawan bot lain di arena. Pertempuran Robocode berlangsung hingga bot-bot bertarung hanya tersisa satu seperti permainan Battle Royale, karena itulah permainan ini dinamakan Tank Royale. Nama Robocode adalah singkatan dari "Robot code," yang berasal dari [versi asli/pertama permainan ini](#). Robocode Tank Royale adalah evolusi/versi berikutnya dari permainan ini, di mana bot dapat berpartisipasi melalui Internet/jaringan. Dalam permainan ini, pemain berperan sebagai programmer bot dan tidak memiliki kendali langsung atas permainan. Pemain hanya bertugas untuk membuat program yang menentukan logika atau "otak" bot. Program yang dibuat akan berisi instruksi tentang cara bot bergerak, mendeteksi bot lawan, menembakkan senjatanya, serta bagaimana bot bereaksi terhadap berbagai kejadian selama pertempuran.

Pada Tugas Besar pertama Strategi Algoritma ini, mahasiswa diminta untuk membuat sebuah bot yang nantinya akan dipertandingkan satu sama lain. Tentunya mahasiswa harus menggunakan strategi *greedy* dalam membuat bot ini.

Komponen-komponen dari permainan ini antara lain:

1. Rounds dan Turns

Pertempuran dapat terdiri dari beberapa rounds. Secara default, satu pertempuran berisi 10 rounds, di mana setiap rounds akan memiliki pemenang dan yang kalah.

Setiap round dibagi menjadi beberapa turns, yang merupakan unit waktu terkecil. Satu turn adalah satu ketukan waktu dan satu putaran permainan. Jumlah turn dalam satu round tergantung pada berapa lama waktu yang dibutuhkan hingga hanya tersisa bot terakhir yang bertahan.

Pada setiap turn, sebuah bot dapat:

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

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 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.

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

4. Peluru

Semakin banyak energi (daya tembak) yang digunakan untuk menembakkan peluru, semakin berat peluru tersebut dan semakin lambat gerakannya. Namun, peluru yang lebih

berat juga menghasilkan lebih banyak kerusakan dan memungkinkan bot mendapatkan lebih banyak energi saat mengenai bot musuh.

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

5. Panas Meriam (Gun Heat)

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

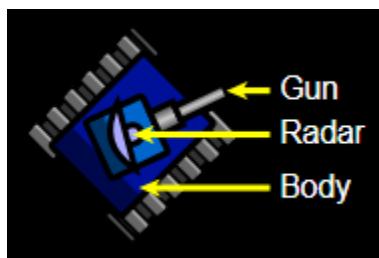
6. Tabrakan

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

Jika bot menabrak bot musuh dengan bergerak maju, ini disebut ramming (menabrak dengan sengaja), yang akan memberikan sedikit skor tambahan bagi bot yang menyerang.

7. Bagian Tubuh Tank

Tubuh tank terdiri dari 3 bagian:



Body adalah bagian utama dari tank yang digunakan untuk menggerakkan tank.

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

Radar digunakan untuk memindai posisi musuh dan dapat berputar bersama *body* atau independen dari *body*.

8. Pergerakan

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

9. Berbelok

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

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

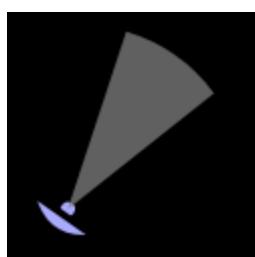
Bagian paling lambat adalah tubuh tank, yang dalam kondisi terbaik dapat berputar hingga 10 derajat per giliran. Namun, ini bergantung pada kecepatan bot saat ini. Semakin cepat bot bergerak, semakin lambat kemampuannya untuk berbelok.

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

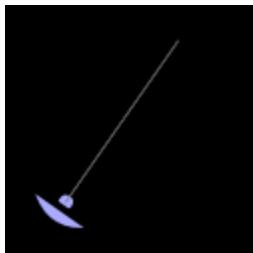
10. Pemindaian

Aspek penting dalam Robocode adalah memindai bot musuh menggunakan radar. Radar dapat mendeteksi bot dalam jangkauan hingga 1200 piksel. Musuh yang berada lebih dari 1200 piksel dari bot tidak dapat terdeteksi atau dipindai oleh radar.

Penting untuk diperhatikan bahwa sebuah bot hanya dapat memindai bot musuh yang berada dalam jangkauan sudut pemindaian (scan arc)-nya. Sudut pemindaian ini merupakan "sapuan radar" dari arah radar sebelumnya ke arah radar saat ini dalam satu giliran.



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



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

11. Skor

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

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

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

BAB II

LANDASAN TEORI

2.1. Dasar Teori

Algoritma *greedy* adalah metode pemecahan masalah yang memilih pilihan terbaik di setiap langkahnya dengan harapan bahwa pilihan tersebut akan mengarah pada solusi optimal secara keseluruhan, tanpa mempertimbangkan konsekuensi jangka panjang. Dalam algoritma greedy, hanya terdapat dua macam persoalan optimasi, yaitu maksimasi (*maximization*) dan minimasi (*minimization*). Metode ini sering digunakan karena sifatnya yang sederhana dan efisien, terutama untuk masalah yang memiliki struktur tertentu yang memungkinkan solusi optimal ditemukan secara lokal.

Algoritma *greedy* banyak digunakan dalam berbagai kasus optimasi. Contoh penerapan algoritma ini adalah algoritma Prim dan algoritma Kruskal yang digunakan untuk menemukan pohon merentang minimum dalam graf berbobot. Selain itu, algoritma Dijkstra juga dapat digunakan untuk mencari jalur terpendek dalam graf berbobot. Dalam bidang kompresi data, kode Huffman memanfaatkan prinsip *greedy* untuk menyusun kode optimal berdasarkan frekuensi kemunculan karakter. Dalam pengembangan program bot yang menggunakan algoritma *greedy*, strategi yang diterapkan biasanya berfokus pada pengambilan keputusan berdasarkan kondisi saat ini tanpa memperhitungkan dampak jangka panjang.

Keunggulan utama dari algoritma ini adalah efisiensinya, karena umumnya memiliki kompleksitas $O(n)$ atau $O(n \log n)$, serta kemudahan implementasinya. Namun, algoritma *greedy* tidak selalu menghasilkan solusi global yang optimal. Dalam beberapa kasus, keputusan yang tampak terbaik dalam satu langkah justru dapat menyebabkan hasil akhir yang tidak optimal. Jika kondisi yang diperlukan terpenuhi, maka algoritma *greedy* dapat menjadi solusi yang cepat dan efektif dalam pemrograman.

2.2. Cara Kerja Program

Untuk dapat menjalankan game, diperlukan starter pack game robocode yang dapat diunduh melalui link berikut [tubes1-if2211-starter-pack](#). Dalam memainkan game ini diperlukan sebuah game engine dan bot yang akan saling bertanding dalam arena.

- Game engine :
<https://github.com/Ariel-HS/tubes1-if2211-starter-pack/releases/download/v1.0/robocode-tankroyale-gui-0.30.0.jar>
- Template bot :

<https://github.com/Ariel-HS/tubes1-if2211-starter-pack/releases/download/v1.0/TemplateBot.zip>

Dalam tugas besar ini, game engine yang digunakan untuk game ini adalah Robocode TankRoyal berbasis Java yang memungkinkan pemain membuat tank virtual dan mengadu strategi dalam pertempuran. Game engine ini bekerja dengan cara seperti berikut :

a. Struktur Dasar

- Robot, merupakan program yang ditulis dengan bahasa C# dan mendukung API yang dimiliki oleh Tank Royale. Bot bisa bergerak, menembak, dan berinteraksi dengan bot lain.
- Arena, tempat pertempuran antar robot terjadi.
- Server, game ini menggunakan server untuk mengelola pertempuran.

b. Komponen Utama

- GUI, antarmuka grafis untuk memvisualisasikan pertempuran pada game ini menggunakan file JAR.
- Bot API yang telah disediakan untuk memprogram bot. Pemain menggunakan API ini untuk mengontrol gerakan, serangan, radar, dan strategi bot.

c. Alur Permainan

- Inisialisasi, sebelum memulai permainan, bot perlu untuk di boot terlebih dahulu ke dalam game engine. Setelah boot berhasil, bot yang telah ditambahkan dapat bertempur dalam arena yang telah disediakan.
- Pertempuran, bot dapat bergerak, menembak, dan bereaksi berdasarkan program yang telah dibuat.
- Event Handling, game engine akan memicu event yang kemudian dapat digunakan oleh pemain untuk mengatur strategi dalam program bot yang dibuat.
- Penilaian, tujuan objektif dari permainan ini adalah mengumpulkan skor sebanyak-banyaknya dengan cara menghancurkan lawan atau menghindar.

2.3. Cara Menjalankan Program

a. Prasyarat

Sebelum memainkan game ini, pastikan telah menginstall Java Runtime Environment (JRE) atau Java Development Kit (JDK) pada device. Selain itu, pastikan terdapat .Net 8 pada device.

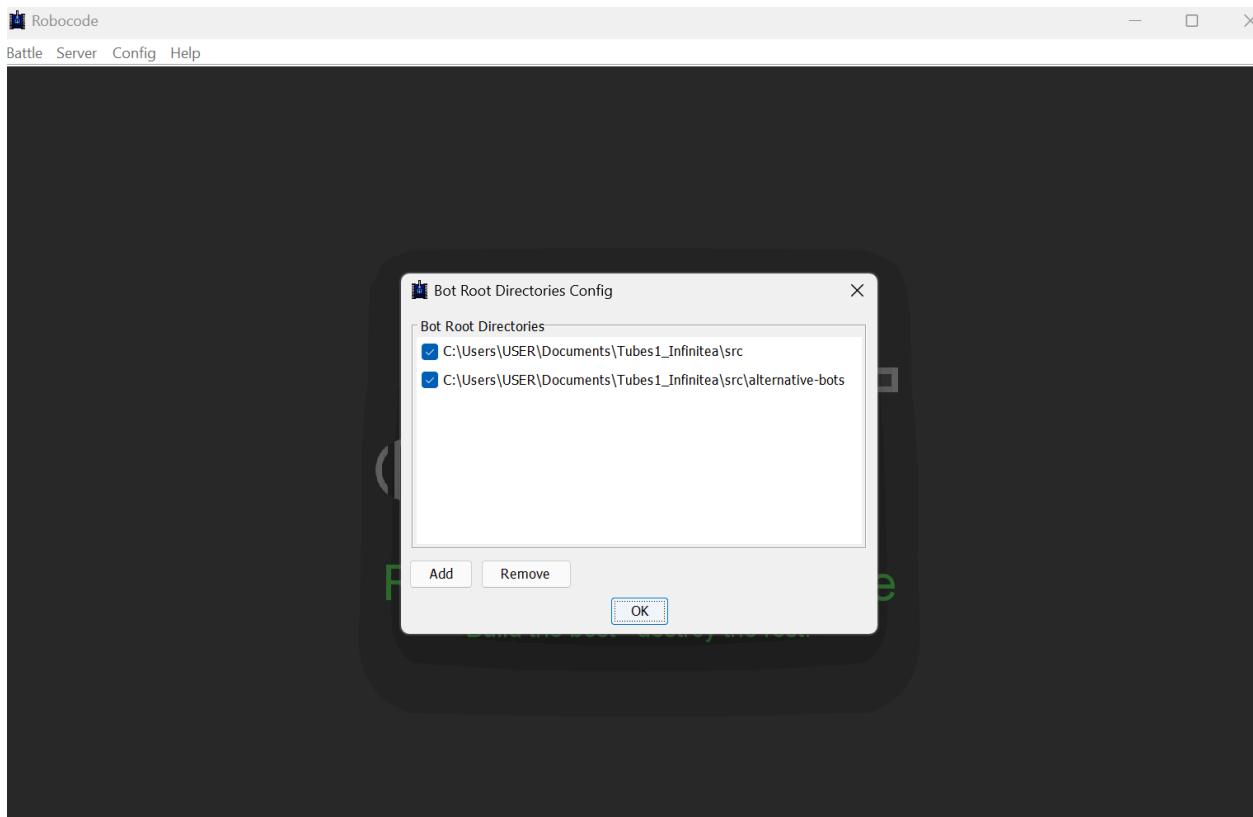
b. Menjalankan GUI

Setelah mengunduh file robocode-tankroyale-gui-0.30.0.jar GUI dapat langsung dibuka dengan *double click* pada file tersebut. Namun apabila file tidak dapat dibuka maka GUI dapat diakses dengan menjalankan perintah berikut di terminal :

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

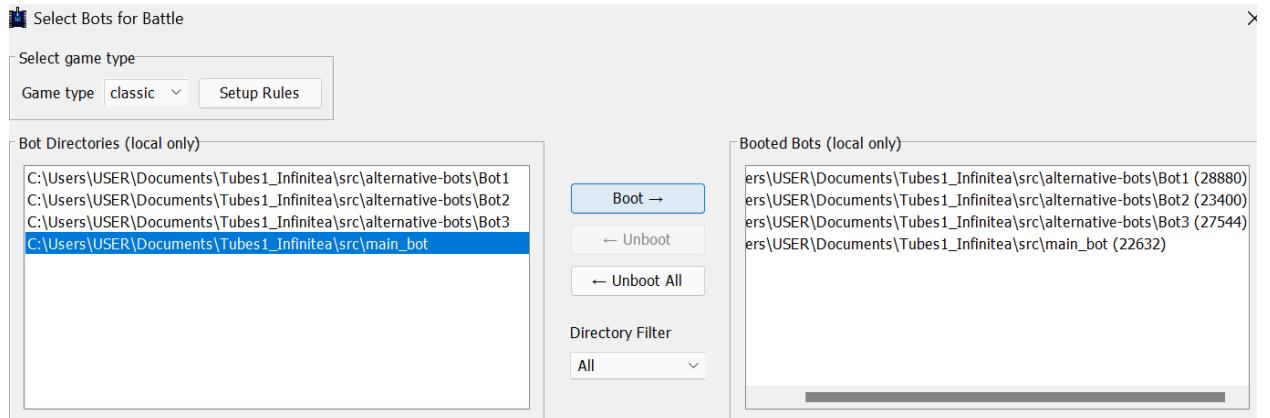
c. Configure directory

Pada config, tambahkan bot root directory Tubes1_Infinitea/src untuk mengakses bot utama dan Tubes1_Infinitea/src/alternative-bots untuk mengakses bot alternatif



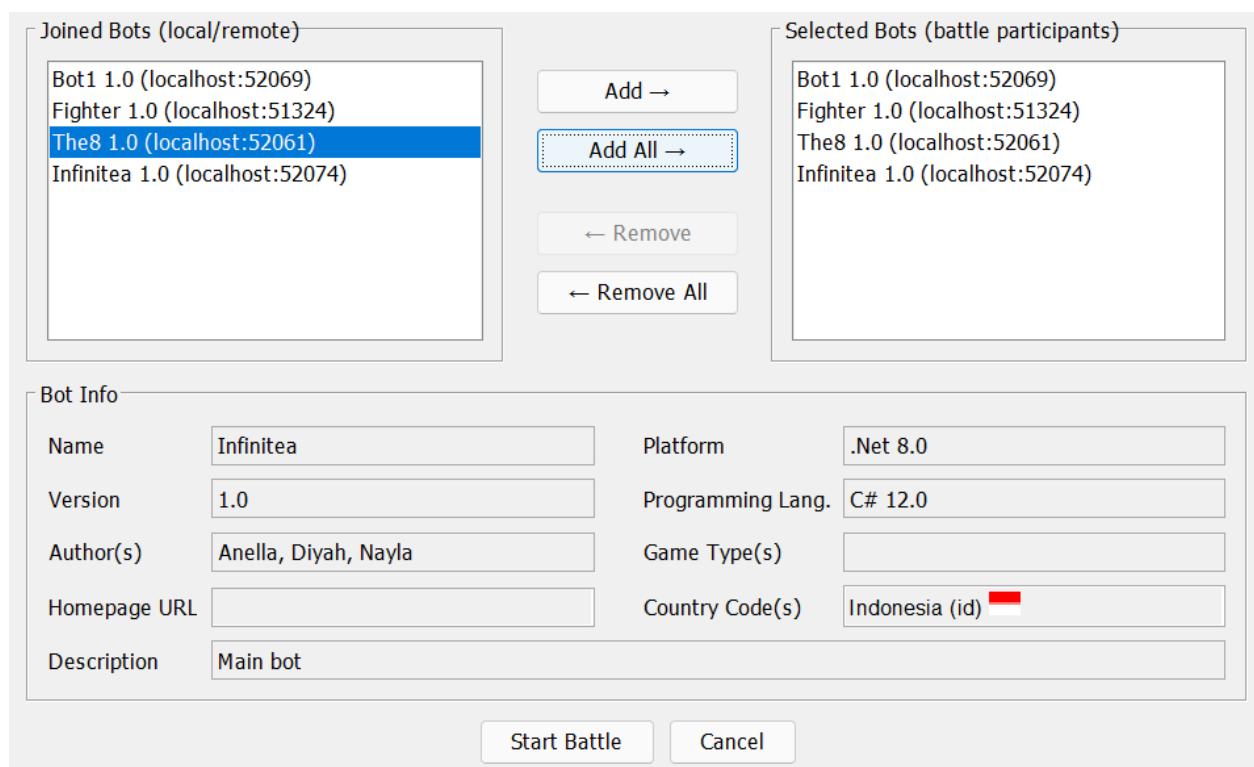
d. Booting Bot

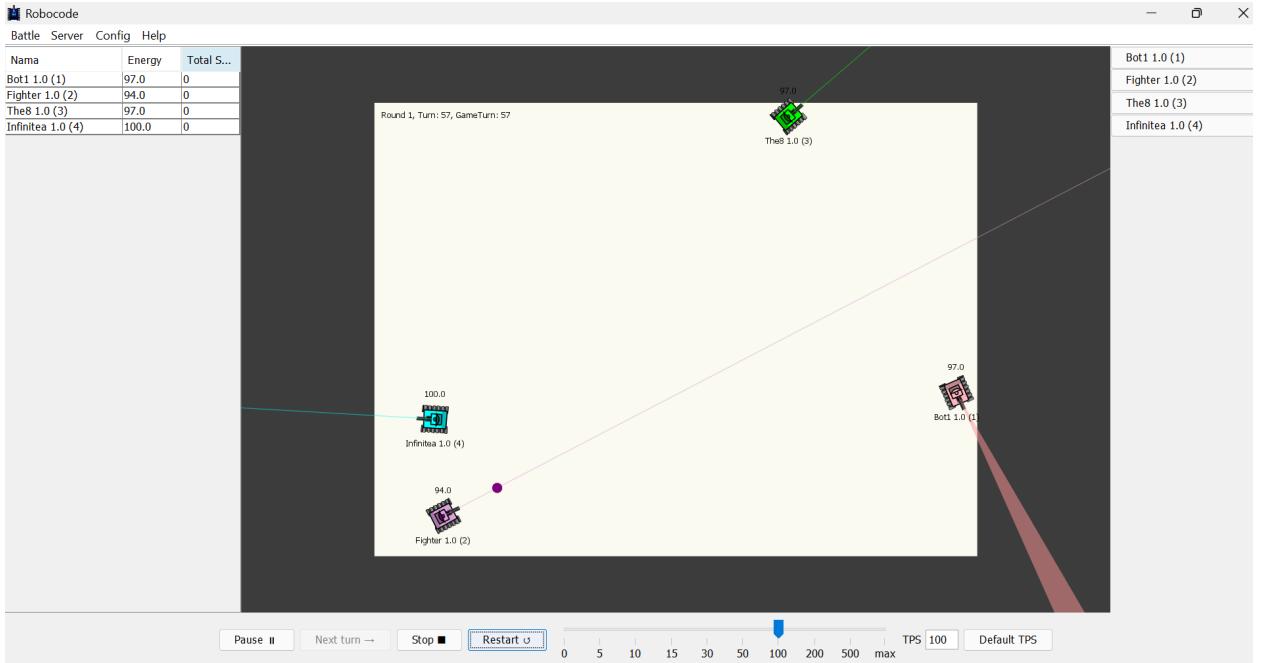
Setelah bot selesai dibuat, lakukan *booting* pada bot yang ingin dipilih supaya dapat ditambahkan ke dalam permainan.



e. *Start Battle*

Setelah menambahkan bot yang ingin dimainkan, mulai permainan dengan klik *start battle*





2.4. Alur Pengembangan Algoritma Greedy pada Program

Pada implementasi ini, kami memiliki struktur folder seperti berikut. Bot-bot yang dibuat dikelompokkan bot menjadi 2 jenis folder yaitu *main_bot* dan *alternative-bots*. Folder *main_bot* terdiri dari satu class yaitu bot utama, sedangkan folder *alternative-bots* terdiri dari beberapa class bot alternatif.

Pengembangan bot dilakukan dengan cara membuat folder baru yang memiliki nama sesuai dengan nama class dari bot yang ingin dikembangkan. Setiap folder bot berisi file .cmd, .cs, .csproj, .json, dan .sh. Kemudian, implementasi algoritma greedy dikembangkan pada file .cs (C#) sesuai dengan event pada permainan, seperti onScannedBot, onHitBot, onHitWall, dan lain-lain. Berikut ilustrasi struktur folder dan contoh implementasi strategi greedy per event:

```

    ▼ TUBES1_INFINITEA
        > docs
        ▼ src
            ▼ alternative-bots
                > Bot1
                > Bot2
                > Bot3
            > main_bot
            README.md

    ▼ Bot1
        > bin
        > obj
        Bot1.cmd
        Bot1.cs
        Bot1.csproj
        Bot1.json
        Bot1.sh

```

```

0 references | Codeium: Refactor | Explain | Generate Documentation | X
public override void OnHitBot(HitBotEvent e)
{
    var distance = DistanceTo(e.X,e.Y);

    if (Energy < 30){                                // jika menabrak
        Back(50);                                    // bot akan meng
        TurnRight(90);
        Forward(100);
    } else {
        TurnToFaceTarget(e.X,e.Y);                  // jika energinya
        Fire(4);                                     // dan menembak
    }
}

0 references | Codeium: Refactor | Explain | Generate Documentation | X
public override void OnHitWall(HitWallEvent e)
{
    Back(50);                                      // jika menabrak
    TurnRight(180);
}

0 references | Codeium: Refactor | Explain | Generate Documentation | X
public override void OnHitByBullet(HitByBulletEvent evt)
{
    TurnRight(90);                                 // jika terkena l
    Forward(100);                                 // dengan belok l
}

```

BAB III

APLIKASI STRATEGI GREEDY

3.1. Alternatif Solusi *Greedy*

3.1.1. Alternatif Solusi 1

3.1.1.1. Proses *Mapping* Alternatif Solusi 1

- a. Himpunan kandidat : Algoritma alternatif solusi ini berfokus pada pemilihan aksi berdasarkan kondisi bot saat itu. Algoritma ini memiliki beberapa pilihan aksi utama, yaitu menembak dengan kekuatan yang bervariasi tergantung jarak dan energi yang dimiliki bot, hal ini bertujuan untuk menghemat energi yang tersisa saat itu dan memanfaatkan kesempatan yang ada untuk menembak musuh dengan melihat jarak antara bot dengan bot musuh.
- b. Himpunan solusi : Langkah-langkah yang digunakan dalam algoritma ini dalam memilih aksi berdasarkan kondisi tertentu yaitu jika bertabrakan dengan bot musuh, algoritma ini akan menyesuaikan strategi antara menyerang atau mundur tergantung pada energi yang dimiliki. Saat mendekati musuh, bot akan menyesuaikan jarak dan energi untuk menentukan kekuatan tembakan yang optimal.
- c. Fungsi seleksi : Selama proses pemilihan aksi, algoritma ini akan mempertimbangkan jarak ke musuh, semakin dekat jaraknya maka akan semakin besar kekuatan tembakan. Algoritma ini juga akan mempertimbangkan energi yang dimiliki pada saat itu, jika energinya masih tinggi maka bot akan lebih agresif untuk menembak dan mendekati musuh, tetapi jika energinya sudah rendah maka bot akan mencari cara aman untuk menghemat energinya.
- d. Fungsi kelayakan : Algoritma ini akan memastikan bahwa setiap aksi yang dipilih masih memungkinkan untuk dilakukan yaitu dengan tidak menembak jika energi tidak mencukupi dan memprioritaskan untuk menghindar saat berada dalam kondisi terancam bahaya.
- e. Fungsi objektif : Algoritma ini bertujuan untuk memaksimalkan skor dengan menembak musuh sebanyak mungkin serta menghindari serangan untuk bertahan hidup lebih lama. Dengan strategi ini, bot dapat menyerang secara efisien dan bertahan dalam pertempuran lebih lama tanpa perhitungan yang terlalu kompleks.

3.1.1.2. Eksplorasi Alternatif Solusi 1

Alternatif solusi 1 memiliki strategi greedy yang berfokus pada aspek penembakan saat bot melakukan scan dan menabrak bot lain. Strategi greedy pada saat scanner bot mengenai bot musuh adalah bot akan maju dan menembak berdasarkan energi yang ia punya saat ini dan jarak antara dirinya dengan bot musuh. Jika jarak dia dengan bot musuh lebih dari 100, maka ia akan maju mendekati musuh dan menembak dengan kekuatan fire 1. Jika jaraknya lebih dari 50, ia akan mengecek terlebih dahulu energi yang dia punya saat ini. Jika energi saat ini masih lebih dari 50, maka ia akan menembak dengan kekuatan fire 3. Jika energinya kurang dari 50, maka ia akan menembak dengan kekuatan fire 2. Jika jaraknya dekat yaitu kurang dari 50, ia akan melakukan hal yang sama yaitu dengan mengecek energinya saat ini dan menembak musuh dengan kekuatan fire 4 jika energinya masih di atas 50 dan akan menembak dengan kekuatan fire 3 jika energinya sudah kurang dari 50.

Strategi greedy yang dipakai selanjutnya berfokus pada saat bot menabrak bot musuh. Sama seperti aspek sebelumnya, aspek ini juga mempertimbangkan berdasarkan energi yang ia punya saat ini. Jika energinya masih di atas 30, maka dia akan mengarahkan botnya ke bot musuh dan menembak dengan kekuatan fire 4. Jika energinya telah kurang dari 30, maka ia akan mundur sebanyak 50 dan mengubah arah dengan membelokkan badan bot ke arah kanan sebanyak 90 derajat. Setelah itu, bot akan maju sesuai dengan algoritma pada saat bot sedang jalan.

Untuk pola jalan pada solusi alternatif ini tidak memiliki strategi greedy khusus karena pola jalan yang diimplementasikan hanya melakukan jalan maju sejauh 200 dan berbelok ke arah kanan sebanyak 45 derajat.

3.1.1.3. Analisis Efisiensi dan Efektivitas Alternatif Solusi 1

- Analisis Efisiensi Solusi

Algoritma pada strategi greedy ini memiliki kompleksitas waktu $O(n)$ karena sebagian besar aksi bot, seperti scanning dan pergerakan, dilakukan dalam iterasi linear terhadap jumlah musuh dan kondisi yang dihadapi. Selain itu, pemrosesan logika dalam setiap kondisi tidak memerlukan pencarian berulang yang kompleks, sehingga efisiensinya tetap dalam batas $O(n)$.

- Analisis Efektivitas Solusi

Strategi ini cukup efektif dalam menyerang musuh karena bot secara aktif bergerak dan menembak berdasarkan jarak dan energi yang dimiliki. Namun, karena bot menggunakan pola gerakan tetap (maju dan berbelok), terkadang pola ini dapat diprediksi oleh musuh yang lebih pintar. Meski demikian, strategi ini cukup baik untuk bertahan dalam jangka waktu lebih lama karena bot terus bergerak, mengurangi kemungkinan terkena tembakan dari lawan secara langsung.

3.1.2. Alternatif Solusi 2

3.1.2.1. Proses *Mapping* Alternatif Solusi 2

- a. Himpunan kandidat: algoritma ini mempertimbangkan kekuatan tembak berdasarkan posisi relatifnya terhadap musuh, yaitu dengan mempertimbangkan jarak dan sudut.
- b. Himpunan solusi: langkah-langkah yang diambil oleh algoritma ini dalam menentukan kekuatan tembak terhadap lawan, seperti menembak dengan kekuatan tinggi saat sudut dan jarak dengan lawan kecil, serta sebaliknya jika sudut dan jaraknya besar.
- c. Fungsi seleksi: fungsi untuk menentukan kekuatan tembak terhadap lawan berdasarkan posisi relatifnya dengan musuh. Ketika posisi memenuhi persyaratan sudut dan jarak yang telah ditentukan maka kekuatan peluru yang akan ditembakkan akan besar dan berlaku sebaliknya.
- d. Fungsi kelayakan: memperoleh sebanyak mungkin skor *bullet* dan *ram damage* dengan tetap memastikan bahwa setiap aksi yang dipilih masih memungkinkan untuk dilakukan yaitu dengan menetapkan kekuatan maksimal menembak adalah energi - 0.1.
- e. Fungsi objektif: memperoleh sebanyak mungkin skor dengan cara menembak dan menabrak sebanyak mungkin bot musuh lalu mendapatkan energi kembali dengan banyaknya *damage* yang diterima musuh.

3.1.2.2. Eksplorasi Alternatif Solusi 2

Alternatif Solusi 2 memiliki strategi yang fokus pada bullet damage dan ram damage. Algoritma ini bertujuan untuk mengumpulkan poin sebanyak

mungkin dengan memberikan damage ke bot musuh. Dalam implementasinya, algoritma ini selalu menggunakan kekuatan tembakan antara 2-5 sehingga dapat memaksimalkan damage ke musuh.

Strategi greedy pada alternatif solusi ini terletak pada proses pemindaian dan penyerangan musuh. Greedy yang digunakan adalah *greedy by angle and distance*. Saat mendeteksi musuh, algoritma ini mempertimbangkan *current position* atau posisi bot saat ini terhadap musuh. Jika jarak dan sudut relatif kecil, algoritma ini akan melakukan penyerangan dengan kekuatan besar. Jika jaraknya masih tidak terlalu jauh (dalam implementasinya diberikan batasan jarak 500), algoritma ini akan menembak dengan kekuatan sedang lalu menghampiri musuhnya untuk mendapatkan *ram damage*. Jika posisi bot saat ini tidak memenuhi dua kasus sebelumnya, algoritma ini hanya akan melakukan penyerangan dengan kekuatan rendah.

Selain penyerangan saat mendeteksi musuh, algoritma ini juga melakukan penyerangan saat tertabrak atau menabrak musuh. Saat hal itu terjadi, algoritma ini melakukan penyerangan dengan kekuatan besar guna memanfaatkan situasi yang memungkinkan musuh berada pada posisi yang sangat dekat.

Selain fokus pada penyerangan, algoritma ini juga melakukan *management* energi dengan selalu memastikan bahwa setiap tindakan penyerangannya tidak melebihi kapasitas energi yang dimiliki. Hal ini dilakukan dengan menentukan minimum antara *fire power* yang hendak dikeluarkan dengan *current energy* dikurang 0.1. Hal ini memastikan bahwa minimalnya terdapat energi 0.1 yang tersisa setelah melakukan penyerangan. Selain pembatasan kekuatan penyerangan, algoritma ini juga memperoleh energi dengan penyerangan, karena bot akan menerima energi sebanding dengan banyaknya *damage* yang diterima musuh.

3.1.2.3. Analisis Efisiensi dan Efektivitas Alternatif Solusi 2

- Analisis Efisiensi Solusi

Algoritma pada strategi greedy ini memiliki kompleksitas waktu maksimal $O(n)$. Hal ini terjadi karena semua operasi dilakukan dalam waktu konstan tanpa adanya iterasi bersarang. Pergerakan utama bot menggunakan loop `run()` yang terus berjalan selama pertandingan. Respon terhadap event seperti `OnScannedBot`, `OnHitWall`, dan `OnHitByBullet` juga bekerja dalam kompleksitas $O(1)$ karena hanya berisi perintah sederhana seperti pergerakan, rotasi, dan tembakan tanpa operasi rekursif atau iteratif

tambahan. Sehingga, maksimal kompleksitas waktu adalah $O(n)$ yaitu saat perintah terjadi bersamaan.

- Analisis Efektivitas Solusi

Strategi alternatif ini cukup efektif dalam melakukan serangan karena menyesuaikan kekuatan tembakan berdasarkan jarak dan sudut terhadap musuh, serta memanfaatkan tabrakan untuk memberikan serangan jarak dekat yang lebih kuat. Selain itu, algoritma ini juga efektif karena bot dapat aktif menyerang dan memperoleh energi di waktu yang bersamaan. Namun, dalam hal penghindaran, algoritma ini kurang efektif karena pola gerakannya yang cenderung dapat diprediksi oleh lawan. Oleh karena itu, dapat disimpulkan bahwa algoritma ini efektif dalam penyerangan dan pengumpulan poin tetapi memiliki kekurangan dalam aspek *survival* atau bertahan hidup.

3.1.3. Alternatif Solusi 3

3.1.3.1. Proses *Mapping* Alternatif Solusi 3

- a. Himpunan kandidat : pada alternatif solusi ini, algoritma yang digunakan berfokus pada pola jalur yang dilalui bot. Pada alternatif ini digunakan pola jalur berbentuk angka delapan, hal ini bertujuan untuk memperluas jangkauan radar sehingga dapat mendekripsi lebih banyak musuh. Pola ini juga bertujuan untuk menghindari serangan musuh dengan menggunakan gerakan yang tidak terduga.
- b. Himpunan solusi : langkah-langkah yang diambil oleh bot ini untuk memilih musuh adalah dengan melakukan *scanning* radar 45 derajat setiap kali bergerak dengan delapan kali gerakan pada arah yang sama dan diulangi lagi dengan arah yang berbeda sehingga membentuk pola menyerupai angka delapan.
- c. Fungsi seleksi : selama proses *scanning bot*, ketika terdeteksi ada musuh maka bot akan menyerang dengan mendekati bot musuh apabila jaraknya masih jauh dan masih memiliki energi yang cukup besar. Hal tersebut dilakukan untuk memperbesar peluang peluru terkena musuh.
- d. Fungsi kelayakan : bot akan menentukan seberapa besar power peluru yang akan ditembakkan ke musuh berdasarkan fungsi seleksi yang telah diimplementasikan sebelumnya.

- e. Fungsi objektif : memperoleh sebanyak mungkin skor dengan cara menembak sebanyak mungkin bot musuh dan menghindar sebisa mungkin dari serangan musuh supaya mendapatkan *survive* bonus dan *bullet* bonus.

3.1.3.2. Eksplorasi Alternatif Solusi 3

Alternatif solusi 3 memiliki strategi yang berfokus pada aspek pola jalur yang digunakan. Bot ini memiliki pola jalur menyerupai angka delapan, hal tersebut terinspirasi dari nomor kelompok kami delapan dan nama kelompok kami infinite. Bot ini menggunakan jalur pola delapan dengan tujuan memiliki peluang lebih besar ketika menghindar dari tembakan musuh karena pergerakan botnya tidak monoton maju atau mundur. Jalur delapan ini juga memungkinkan pemindaian arena yang lebih luas karena kepala bot akan menyisir seluruh sudut arena secara perlahan-lahan. Pola delapan dibentuk dari pergerakan delapan kali sudut 45 derajat dengan arah ke kanan untuk loop pertama dan arah ke kiri untuk loop kedua. Dengan demikian pola delapan akan terbentuk sebagai bentuk dua lingkaran yang menyerupai angka delapan.

Selain berfokus pada aspek jalur, bot ini juga mempertimbangkan ukuran peluru yang ditembakkan berdasarkan jarak dengan musuh. Ketika jarak terlalu jauh, maka bot akan menembakkan peluru yang kecil berukuran dua karena peluang terkena musuh juga kecil. Sebaliknya, jika musuh berada dekat dengan bot maka peluru yang ditembakkan akan besar berukuran tiga.

Strategi lain yang diterapkan dalam bot ini adalah ketika bot menabrak musuh maka bot akan mengukur posisi musuh dan menghadap ke wajah musuh lalu menembakkan peluru dengan kekuatan berukuran lima, hal tersebut dilakukan karena peluang peluru terkena musuh sangat besar dalam kondisi ini. Ketika bot terkena peluru dari musuh, bot akan mencoba untuk kabur dan mengubah arah geraknya. Apabila bot menabrak tembok, bot akan berputar balik 180 derajat dan mengubah arah geraknya supaya tidak mengarah ke tembok lagi.

3.1.3.3. Analisis Efisiensi dan Efektivitas Alternatif Solusi 3

- Analisis Efisiensi Solusi

Algoritma pada alternatif solusi ini memiliki kompleksitas waktu $O(n)$. Hal ini terjadi karena dalam algoritma ini dilakukan iterasi *looping* sebanyak n kali ketika bot berjalan dengan pola angka delapan. *Method* yang lain menggunakan kompleksitas maksimal $O(n)$ karena tidak ada

iterasi khusus dan hanya membandingkan kondisi saja. Dengan demikian, total waktu yang dibutuhkan dalam algoritma ini adalah sebesar $O(n)$.

- Analisis Efektifitas Solusi

Algoritma ini cukup efektif untuk menghindar dari serangan musuh karena bot selalu bergerak dan mengubah arah geraknya sehingga peluang peluru mengenai bot menjadi kecil. Akan tetapi, dalam konteks *scanning* musuh, bot ini kurang efisien caranya berjalan karena terkadang dia menyisir area yang berlawanan arah dengan posisi musuh yang ada. Namun apabila digunakan untuk melawan banyak musuh, pola gerak bot ini tergolong efisien karena dengan *scanning* radar ke segala arah maka peluang menyerang musuh juga semakin besar.

3.1.4. Alternatif Solusi 4

3.1.4.1. Proses *Mapping* Alternatif Solusi 4

- a. Himpunan kandidat : Algoritma ini mempertimbangkan kekuatan tembak berdasarkan jarak dan sudut dengan lawan. Algoritma ini juga mempertimbangkan kekuatan peluru ketika akan melakukan serangan balik saat ditabrak oleh bot musuh berdasarkan jumlah energi. Selain itu, algoritma ini juga memiliki pola jalan yang unik untuk menghindar dari serangan musuh serta memperbesar area *scanning* radar dalam rangka mencari musuh.
- b. Himpunan solusi : Langkah-langkah yang diambil oleh algoritma ini dalam menentukan kekuatan tembak terhadap lawan, seperti menembak dengan kekuatan tinggi saat sudut dan jarak dengan lawan kecil, serta sebaliknya jika sudut dan jaraknya besar. Pada kondisi menabrak dengan bot lain, algoritma ini akan menentukan kekuatan tembak terhadap lawan berdasarkan jumlah energi yang dimiliki, seperti menembak dengan kekuatan tinggi jika energi yang dimiliki masih tinggi, dan akan menghindar jika energi yang dimiliki sudah rendah.
- c. Fungsi seleksi : Menentukan kekuatan tembak terhadap lawan berdasarkan posisi relatifnya dengan musuh. Ketika posisi memenuhi persyaratan sudut dan jarak yang telah ditentukan maka kekuatan peluru yang akan ditembakkan akan besar dan berlaku sebaliknya. Ketika energi memenuhi batas minimum untuk serangan balik maka ketika menabrak musuh akan dilakukan serangan balik sesuai dengan yang telah ditentukan.

- d. Fungsi kelayakan : Apakah algoritma alternatif ini dapat menyesuaikan serangan berdasarkan posisi dan kondisi yang telah ditentukan seperti jarak dan sudut ketika akan menembak musuh, serta sisa energi ketika akan menyerang balik.
- e. Fungsi objektif : Mendapatkan skor sebanyak mungkin dengan cara paling efisien dengan mempertimbangkan posisi dan kondisi ketika akan menyerang musuh.

3.1.4.2. Eksplorasi Alternatif Solusi 4

Alternatif solusi ini memadukan tiga alternatif sebelumnya, yaitu perpaduan antara strategi *fire by distance and energy*, *fire by distance and angle*, serta *greedy by adaptability*. Sehingga, algoritma ini menghasilkan strategi greedy yang dapat berfokus pada aspek *bullet damage*, *ram damage*, dan *survival point*. Ide utama dalam alternatif solusi ini adalah membangun algoritma yang pintar secara keseluruhan atau *all-rounder built*. Dalam implementasinya, beberapa poin penting yang digunakan adalah sebagai berikut:

1. Selalu menggunakan pola pergerakan yang unik yaitu pola jalan 8 untuk menghindari serangan musuh serta memperluas area *scanning radar* dalam rangka pencarian musuh.
2. Jika mendeksi musuh, algoritma ini akan mempertimbangkan kekuatan menembak berdasarkan jarak dan sudut dengan lawan. Jika jarak dan sudutnya relatif kecil, ia akan menembak dengan kekuatan tinggi. Jika sudutnya kecil dan jaraknya tidak terlalu besar, ia akan menembak dengan kekuatan sedang lalu menghampiri musuhnya untuk mendapatkan *ram damage*. Jika tidak masuk ke dua kasus sebelumnya, ia akan menembak dengan kekuatan kecil.
3. Pada saat menabrak/tertabrak musuh, algoritma ini akan mempertimbangkan jumlah energi saat ini untuk menentukan kekuatan serangan balik terhadap lawan.
4. Jika tertembak oleh *bullet* musuh, algoritma ini akan memutar badan bot 90 derajat ke kanan lalu maju untuk menghindar dari serangan susulan oleh musuh.
5. Jika menabrak tembok, algoritma ini akan memutar badan bot 180 derajat lalu maju untuk menghindari tembok.

3.1.4.3. Analisis Efisiensi dan Efektivitas Alternatif Solusi 4

- Analisis Efisiensi Solusi

Algoritma pada strategi greedy ini memiliki kompleksitas waktu $O(n)$ karena aksi yang dilakukan dalam algoritma ini seperti saat melakukan *scanning* dan pergerakan dilakukan secara iteratif terhadap sudut musuh yang terdeteksi. Setiap keputusan dalam menembak, menghindar, dan bergerak diproses dalam satu iterasi, sehingga efisiensinya tetap linier terhadap jumlah interaksi dengan musuh. Dengan demikian, strategi ini tetap efisien dalam menghadapi berbagai skenario pertempuran dengan kompleksitas waktu $O(n)$.

- Analisis Efektivitas Solusi

Algoritma ini cukup efektif untuk menghindari serangan musuh karena bot akan selalu bergerak dengan pola angka 8, sehingga sulit menjadi target tetap oleh musuh dan memperluas area untuk mendeteksi musuh. Selain itu, algoritma ini juga akan membuat bot menghindar saat terkena tembakan atau saat energi yang dimilikinya rendah, hal ini dapat meningkatkan peluang untuk bertahan. Dalam mendeteksi musuh, algoritma ini juga cukup efektif karena menyesuaikan serangan berdasarkan sudut dan jarak musuh sehingga tembakan yang dilakukan dapat lebih akurat dan efisien dalam penggunaan energi.

3.2. Strategi *Greedy* Solusi Utama

Strategi greedy yang dipilih sebagai solusi utama adalah alternatif solusi empat. Alternatif empat dipilih sebagai solusi utama karena strategi yang digunakan pada alternatif solusi empat telah memadukan semua unsur yang diperlukan yaitu strategi jalan, menembak, dan serangan balik sehingga secara performa alternatif solusi empat telah menangani setiap kemungkinan kasus. Strategi yang digunakan pada alternatif solusi empat juga sebenarnya diambil dari perpaduan ketiga alternatif solusi yang lain dan diambil aspek paling menonjol dari masing-masing alternatif solusi yang kemudian dipadukan menjadi satu. Seiring dengan berjalannya *trial and error*, strategi yang diambil untuk solusi utama juga telah dimodifikasi untuk meningkatkan performa algoritma supaya dapat menghasilkan skor semaksimal mungkin.

BAB IV

IMPLEMENTASI DAN PENGUJIAN

4.1.Implementasi Solusi

4.1.1. Implementasi Bot Utama

```
Run ()  
while IsRunning do  
    {membentuk pola 8, loop dua kali dengan membuat lingkaran yang arahnya berbeda}  
        i traversal [0..8]  
            SetTurnRight (45)  
            Forward (70)  
  
        i traversal [0..8]  
            SetTurnLeft (45)  
            Forward (70)  
  
OnScannedBot(ScannedBotEvent e)  
    {jika jarak dan sudut dengan musuh relatif kecil, tembak dengan kuat. jika jaraknya  
     tidak terlalu jauh, tembak lalu dekati musuh untuk mendapat ram damage. Selain  
     kasus 2 itu, tembak dengan kekuatan kecil saja}  
        bearing ← BearingTo(e.X, e.Y)  
        distance ← DistanceTo(e.X, e.Y)  
        if absolute(distance) < 100 and absolute(bearing) < 10 then  
            TurnLeft(bearing)  
            Fire(min(4, Energy-0.1))  
        else if (absolute(distance) <=500) then  
            TurnLeft(bearing)  
            Fire(min(2.5, Energy-0.1))  
        else  
            TurnLeft(bearing)  
            Fire(min(2, Energy-0.1))  
  
OnHitBot(HitBotEvent e)  
    {bot akan menembak dengan kekuatan tinggi jika energinya masih tinggi, tetapi akan  
     menghindar jika energinya rendah}  
    distance ← DistanceTo(e.X, e.Y)  
    if Energy < 30 then  
        TurnRight(90)  
        Forward(100)  
    else  
        TurnToFaceTarget(e.X, e.Y)  
        Fire(5)  
  
OnHitWall(HitWallEvent e)  
    {putar balik lalu maju}  
    TurnRight(180)  
    Forward(50)  
  
OnHitByBullet(HitByBulletEvent evt)  
    {bot akan menghindar dengan mengganti arah jalannya ke arah kanan sebesar 90 derajat}  
    TurnRight(90)  
    Forward(100)  
  
TurnToFaceTarget(x,y)  
bearing ← BearingTo(x,y)
```

```

if bearing >= 0 then
    turnDirection <- 1
else
    turnDirection <- -1
TurnLeft(bearing)

```

4.1.2. Implementasi Bot Alternatif 1

```

Run()
while IsRunning do
    Forward(200)
    TurnRight(45)

OnScannedBot(SacnnedBotEvent e)
    {bot akan menembak dengan kekuatan kecil jika jaraknya jauh dan energinya rendah, bot
     akan menembak dengan kekuatan besar jika jaraknya dekat dan energinya tinggi}

    Distance ← DistanceTo(e.X, e.Y)
    if distance > 100 then
        Fire(1)
        Forward(100)
    else if distance > 50 then
        Forward(50)
        if Energy > 50 then
            Fire(3)
        else
            Fire(2)
    else
        if Energy > 50 then
            Fire(4)
        else
            Fire(3)

OnHitBot(HitBotEvent e)
    {bot akan menembak dengan kekuatan tinggi jika energinya masih tinggi, tetapi akan
     menghindar jika energinya rendah}

    Distance ← DistanceTo(e.X, e.Y)
    if Energy < 30 then
        Back(50)
        TurnRight(90)
        Forward(100)
    else
        TurnToFaceTarget(e.X, e.Y)
        Fire(4)

OnHitWall(HitWallEvent e)
    {bot akan mundur dan putar balik}
    Back(50)
    TurnRight(180)

OnHitByBullet(HitByBulletEvent evt)
    {bot akan menghindar dengan mengganti arah jalannya ke arah kanan sebesar 90 derajat}

    TurnRight(90)
    Forward(100)

```

```

TurnToFaceTarget(x,y)
bearing ← BearingTo(x,y)
if bearing >= 0 then
    turnDirection ← 1
else
    turnDirection ← -1
TurnLeft(bearing)

```

4.1.3. Implementasi Bot Alternatif 2

```

Run ()
while IsRunning do
    {berjalan lurus dan melingkar kiri}
    Forward(450);
    SetTurnLeft(10_000);

OnScannedBot(ScannedBotEvent e)
    {jika jarak dan sudut dengan musuh relatif kecil, tembak dengan kuat. jika
     jaraknya tidak terlalu jauh, tembak lalu dekati musuh untuk mendapat ram damage.
     Selain kasus 2 itu, tembak dengan kekuatan kecil saja}
    bearing ← BearingTo(e.X, e.Y)
    distance ← DistanceTo(e.X, e.Y)
    if absolute(distance) < 100 and absolute(bearing) < 10 then
        TurnLeft(bearing)
        Fire(min(4, Energy-0.1))
    else if (absolute(distance) <=500) then
        TurnLeft(bearing)
        Fire(min(2.5, Energy-0.1))
    else
        TurnLeft(bearing)
        Fire(min(2, Energy-0.1))

OnHitByBullet (HitByBulletEvent evt)
    {kalau terkena peluru, bot akan menghindar dengan mundur lalu belok kanan lalu maju
     kembali}
    Forward(-5)
    TurnRight(10)
    Run()

OnHitBot (HitBotEvent e)
    {jika menabrak/ditabrak dan energi mencukupi, bot akan menembak dengan kekuatan
     tinggi}
    {kalau ditabrak bot lain, bot akan menghindar dengan belok kiri 10 derajat}
    bearing ← BearingTo(e.X, e.Y)
    if (absolute(bearing) <= 10) then
        TurnLeft(bearing)
        Fire(min(3, Energy-0.1))
        Run()
    if (e.IsRammed) then
        TurnLeft(10)

OnHitWall (HitWallEvent evt)
    {kalau menabrak tembok, mundur lalu putar arah kiri 10 derajat}
    Forward(-30)
    TurnLeft(10)
    Run()

```

4.1.4. Implementasi Bot Alternatif 3

```
Run ()
while IsRunning do
    {membentuk pola 8, loop dua kali dengan membuat lingkaran yang arahnya berbeda}
    i traversal [0...8]
        SetTurnRight (45)
        Forward (70)

    i traversal [0...8]
        SetTurnLeft (45)
        Forward (70)

OnScannedBot(ScannedBotEvent e)
    {kalau musuh jauh dan masih punya banyak energi, bot akan mendekat}
    TurnToFaceTarget()
    Distance ← DistanceTo(e.X, e.Y)
    if distance > 100 and Energy > 80 then
        SetFire (2)
        Forward (50)
    else
        setFire (3)

OnHitByBullet (HitByBulletEvent evt)
    {kalau terkena peluru, akan berusaha menghindar dengan mundur}
    Back (100)
    TurnLeft (120)
    Forward (100)
    TurnRight (60)

OnHitBot (HitBotEvent e)
    {kalau ditabrak bot lain, bot akan mengarah ke musuh dan menembak dengan kekuatan besar}
    {karena peluang peluru mengenai musuh akan besar}
    TurnToFaceTarget (e.X, e.Y)
    Fire (5)
    Run ()

OnHitWall (HitWallEvent evt)
    {kalau menabrak tembok, putar balik dan belok untuk menghindari menabrak lagi}
    TurnLeft (180)
    Forward (100)
    TurnRight (100)

TurnFaceToTarget (double x, double y)
    {mendeteksi musuh dan menghadap ke musuh dengan memperhitungkan sudut dan arah}
    bearing ← BearingTo (x, y)
    if (bearing >= 0) then
        turnDirection = 1
    else
        turnDirection = -1
    TurnLeft (bearing)
```

4.2.Penjelasan Program Bot Utama

4.2.1 Struktur Data Bot Utama

Program ini menggunakan paradigma *object-oriented programming* dengan menggunakan bahasa C#. Dalam program ini terdapat satu kelas bernama main_bot yang merupakan sebuah kelas yang diwarisi oleh kelas Bot yang berasal dari library Robocode.TankRoyale.BotApi. Karena diturunkan dari kelas bot maka segala sifat dan perilaku yang dimiliki oleh bot pada API tersebut diwariskan kepada main_bot. Struktur data yang digunakan dalam program ini meliputi variabel numerik seperti turnDirection yang menyimpan arah rotasi bot, serta tipe data berbasis objek seperti Color untuk menentukan warna komponen bot.

4.2.2 Fungsi dan Prosedur Bot Utama

- Run

Fungsi Run bertanggung jawab dalam mengatur pergerakan bot selama permainan berlangsung. Fungsi ini mengimplementasi pola gerakan angka 8 yang dilakukan melalui iterasi dengan memanfaatkan perintah SetTurnRight, SetTurnLeft, dan Forward. Setiap loop akan melakukan iterasi sebanyak delapan kali dan setiap gerakan akan merubah sudut sebesar 45 derajat. Ketika dua loop telah dilakukan maka akan terbentuk pola menyerupai angka delapan.

- OnScannedBot

Fungsi OnScannedBot bertanggung jawab dalam menentukan aksi bot ketika mendeteksi musuh. Strategi yang digunakan adalah dengan menentukan kekuatan tembak peluru berdasarkan jarak dan sudut musuh. Jika musuh berada dalam jangkauan dan sudutnya sempit maka bot akan menembak dengan daya yang besar karena peluru mengenai musuh juga besar. Variabel *bearing* bertujuan untuk menghitung sudut sedangkan variabel *distance* bertujuan untuk mengukur jarak terhadap musuh. Jika kondisi memenuhi syarat, maka bot akan menembak dengan kekuatan fire 4, jika sudut terlalu lebar maka bot akan menembak dengan kekuatan fire 2.5, dan selain kondisi tersebut bot akan menembak dengan kekuatan fire 2.

- OnHitBot

Fungsi OnHitBot bertanggung jawab untuk menangani interaksi dengan musuh yang bertabrakan dengan bot. Strategi greedy yang digunakan dalam fungsi ini mempertimbangkan energi yang dimiliki oleh bot. Bot akan menghadap ke arah musuh dengan memanggil fungsi TurnToFaceTarget dan menembak dengan kekuatan tinggi yaitu 5 jika energinya mencukupi, tetapi bot akan menghindar

dengan cara berbelok ke arah kanan sebanyak 90 derajat kemudian maju sejauh 100 dengan memanfaatkan perintah TurnRight dan Forward jika energi yang dimilikinya rendah, yaitu kurang dari 30.

- OnHitWall

Fungsi OnHitWall akan mengubah arah gerak bot etika menabrak tembok dengan cara melakukan putar balik. Setelah putar balik bot akan melanjutkan jalannya dengan maju 50 dan lanjut melakukan Run.

- OnHitByBullet

Fungsi OnHitByBullet memungkinkan bot untuk menghindar ketika terkena tembakan dari musuh. Bot akan menghindar dengan cara berbelok ke arah kanan sebesar 90 derajat kemudian maju sejauh 100. Fungsi ini memanfaatkan perintah TurnRight dan Forward.

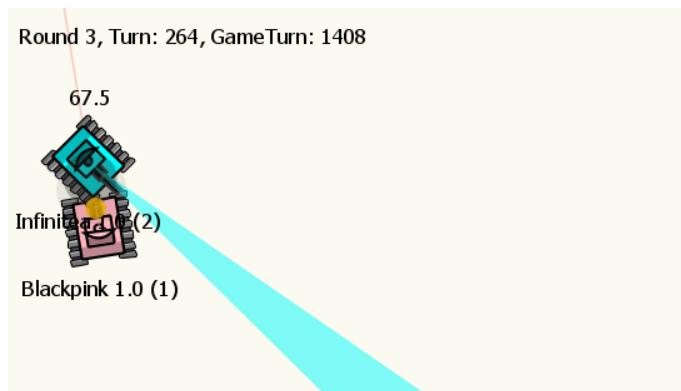
- TurnToFaceTarget

Fungsi TurnToFaceTarget digunakan untuk memastikan bahwa bot selalu mengarahkan senjatanya ke musuh sebelum menembak. Hal ini dilakukan untuk memperbesar peluang peluru mengenai musuh dengan akurat supaya energi yang dikeluarkan tidak terbuang sia-sia. Fungsi ini diimplementasikan dengan menghitung sudut relatif terhadap musuh dan mengubah arah sesuai dengan sudut tersebut.

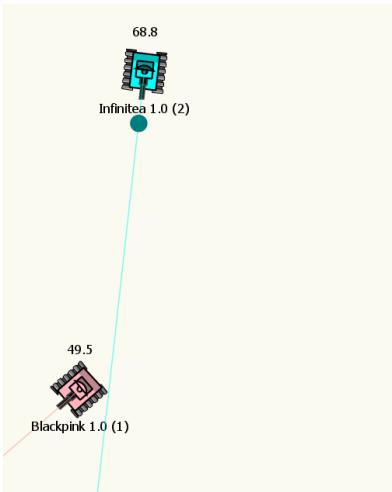
4.3. Hasil Pengujian Bot Utama

4.3.1 Bot Utama Melawan Alternatif Bot 1

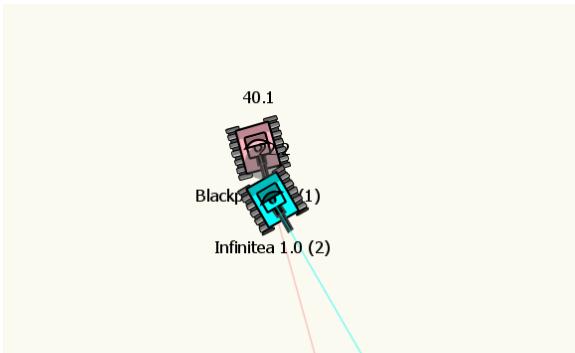
Kondisi saat kedua bot saling bertabrakan dan saling menembak



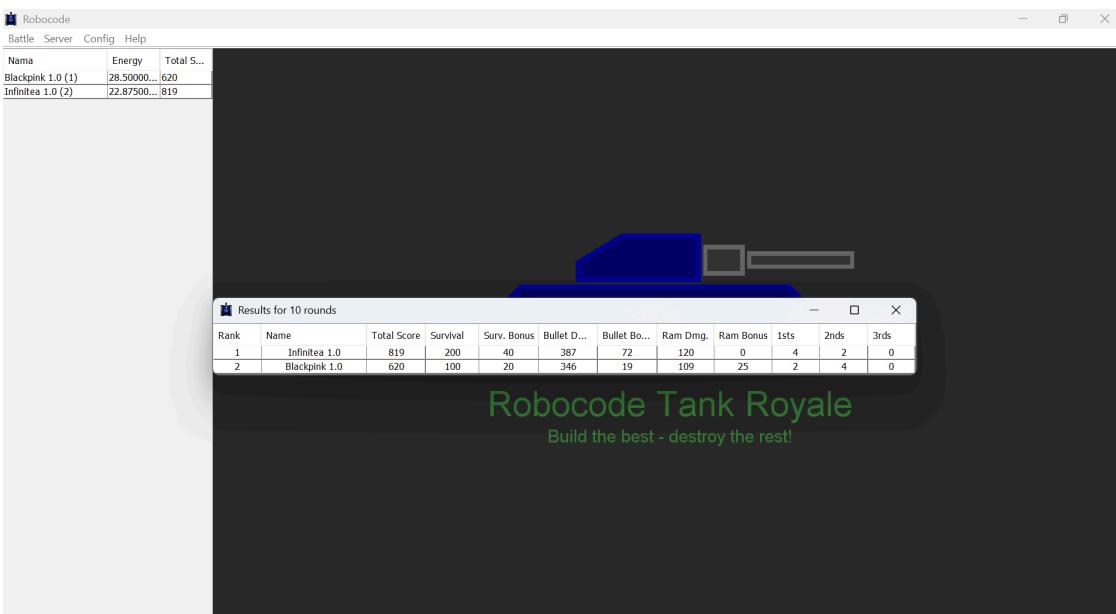
Kondisi saat bot utama mendeteksi alternatif bot 1 dan menembaknya



Kondisi saat bot utama putar balik karena energi rendah



Hasil akhir pertarungan setelah sepuluh *round* :

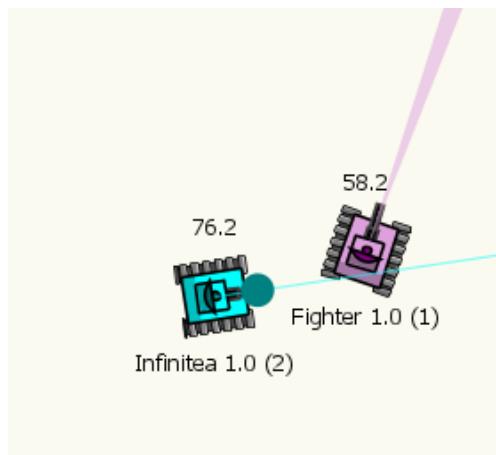


4.3.1.1 Analisis Hasil Pengujian

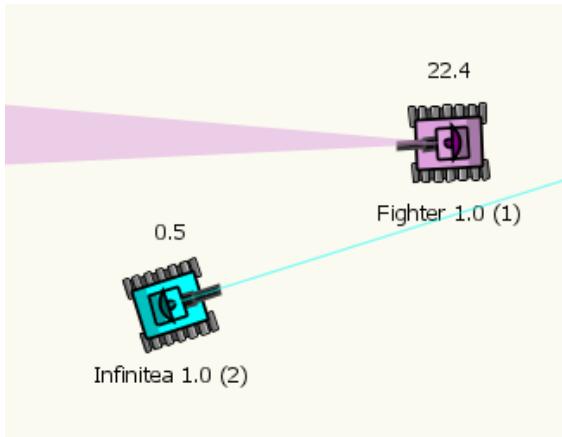
Berdasarkan hasil pengujian bot utama melawan alternatif bot 1 sebanyak sepuluh ronde, bot utama (Infinitea) berhasil memenangkan permainan dengan menang sebanyak empat kali dan kalah sebanyak dua kali. Dilihat dari keseluruhan skor, bot utama lebih unggul hampir 200 poin dibandingkan dengan alternatif bot 1 (Blackpink). Dilihat dari aspek *survival score* dan *survival bonus*, bot utama lebih unggul dua kali lipat dibandingkan dengan alternatif bot 1, ini berarti bot utama mampu bertahan lebih lama dalam suatu ronde dibandingkan alternatif bot 1. Pada aspek *bullet damage*, bot utama tetap lebih unggul dibandingkan alternatif bot 1 yang berarti bot utama memberikan *damage* yang lebih besar melalui tembakan peluru yang menunjukkan keakuratan dan efektivitasnya dalam menyerang. Bot utama juga mengungguli skor pada aspek *bullet damage bonus* yang berarti bot utama lebih sering menembakkan alternatif bot 1 hingga mati sehingga memperoleh bonus yang jauh lebih besar dibandingkan alternatif bot 1. Pada aspek *ram damage*, bot utama tetap lebih unggul dalam menabrakkan dirinya terhadap alternatif bot 1. Tetapi dalam aspek *ram bonus*, alternatif bot 1 lebih unggul 25 poin yang berarti alternatif bot 1 lebih sering menghabisi musuh dengan menabrak sehingga mendapatkan lebih banyak bonus dari strategi ini dibandingkan bot utama.

4.3.2 Bot Utama Melawan Alternatif Bot 2

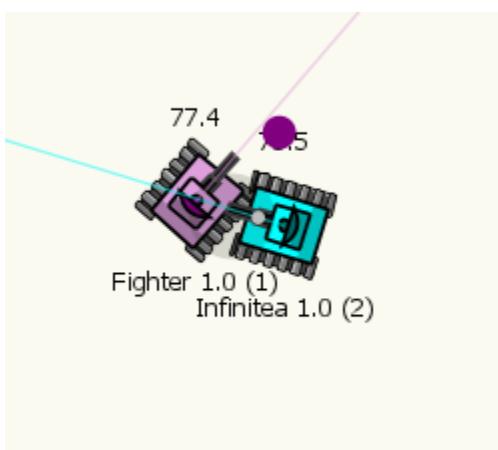
Kondisi ketika menembak bot lain



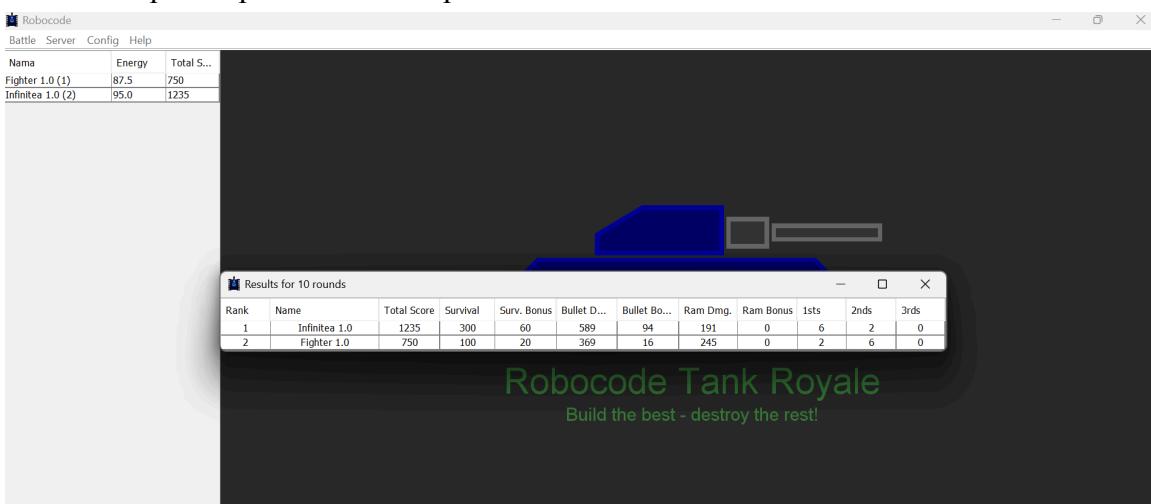
Kondisi ketika melakukan *scanning* musuh dengan radar



Kondisi ketika melakukan *ram* setelah ditabrak musuh



Hasil akhir pertempuran setelah sepuluh *round*



4.3.2.1 Analisis Hasil Pengujian Melawan Alternatif Bot 2

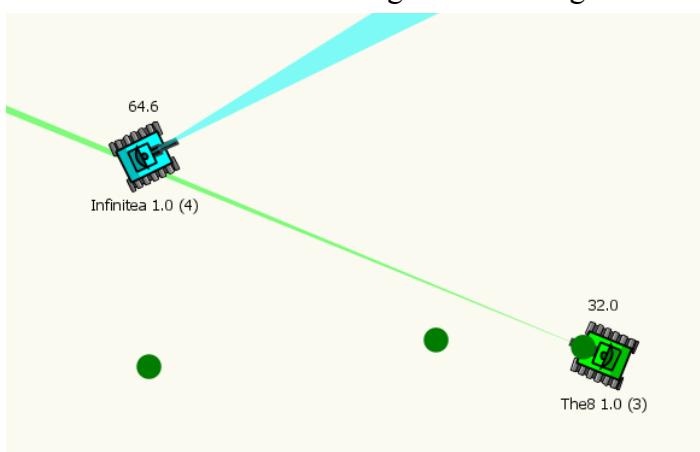
Berdasarkan hasil pengujian bot utama dengan bot alternatif 2, bot utama memperoleh hasil menang 6 kali dan kalah 2 kali. Hal ini menunjukkan frekuensi

kemenangannya adalah 75% dari total kejadian. Jika dilihat dari aspek *survival poin*, bot utama lebih unggul dalam bertahan hidup dibandingkan bot alternatif 2. Hal ini menunjukkan efektifitas dari strategi greedy by adaptability yang dimiliki oleh bot utama, yaitu dengan menggunakan pola pergerakan yang unik.

Namun, pada aspek *bullet damage* dan *ram damage*, jika poinnya dijumlahkan, kedua bot akan menunjukkan hasil yang tidak jauh berbeda. Hal ini terjadi karena bot utama juga menggunakan strategi greedy “*fire by distance and angle*” yang dimiliki oleh bot alternatif 2.

4.3.3 Bot Utama Melawan Alternatif Bot 3

Kondisi ketika bot utama menghindari serangan



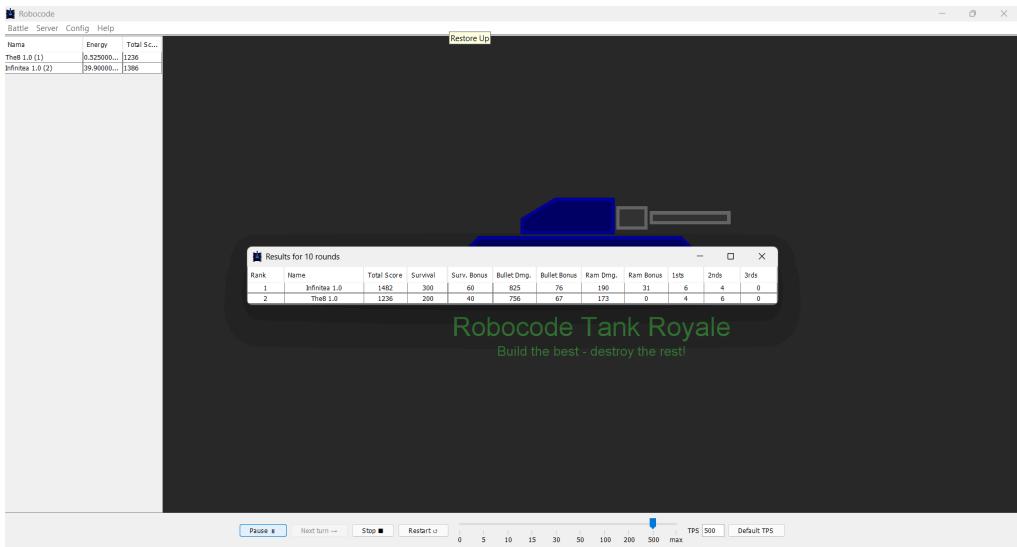
Kondisi ketika bot utama melakukan *ram* saat ditabrak bot lain



Kondisi ketika bot mencoba untuk mengejar bot lain setelah musuh terdeteksi oleh radar



Hasil akhir pertempuran setelah sepuluh *round*

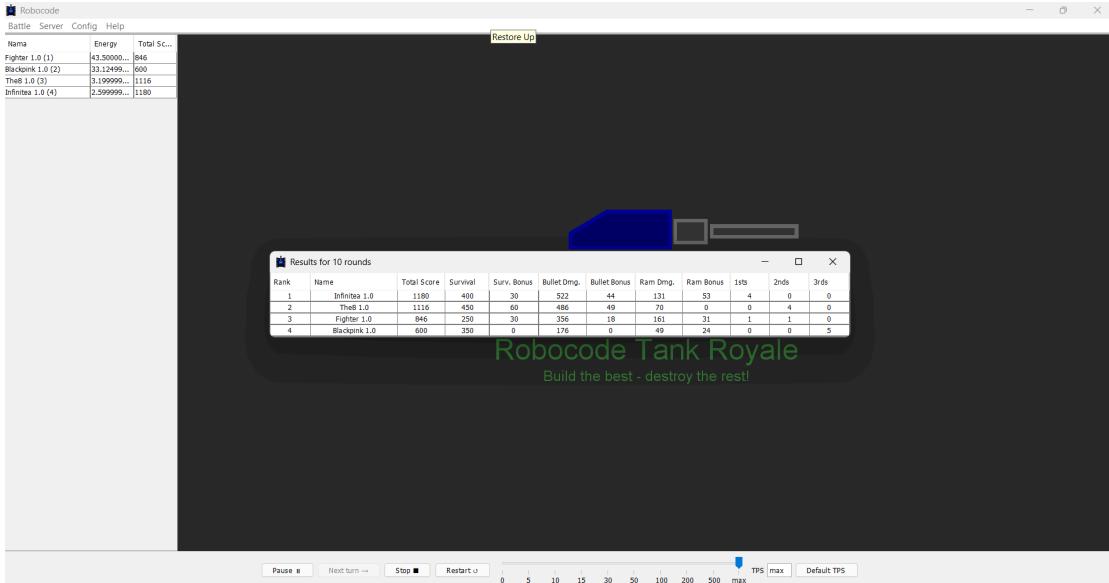


4.3.3.1 Analisis Hasil Pengujian Melawan Alternatif Bot 3

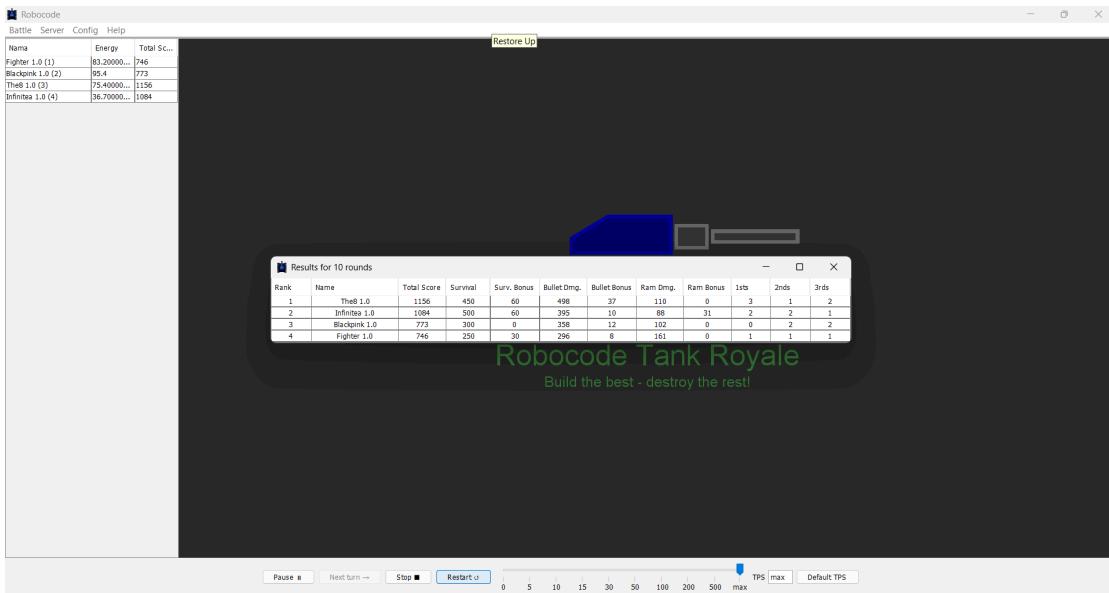
Berdasarkan hasil pengujian bot solusi utama melawan bot alternatif tiga, diperoleh hasil dalam sepuluh *round* bot utama menang sebanyak enam kali dan kalah sebanyak empat kali. Skor akhir yang dihasilkan cukup sengit dengan selisih yang tidak begitu jauh dikarenakan kedua bot memang memiliki performa yang baik namun bot utama lebih unggul dengan pendekatan *greedy* yang lebih baik. Skor paling banyak diperoleh dari *bullet damage* karena bot utama menggunakan strategi *greedy* dalam menentukan kekuatan peluru untuk memperoleh skor semaksimal mungkin. Selain dari *bullet damage*, skor juga banyak diperoleh dari *ram damage*. Hal tersebut terjadi karena pada bot utama digunakan strategi *greedy* ketika ditabrak oleh bot musuh maka akan melakukan serangan balik dengan menembak ke hadapan musuh. Selain itu, skor juga diperoleh dari *survival* karena bot utama dapat bertahan terakhir sebanyak enam kali dari dalam sepuluh *round*.

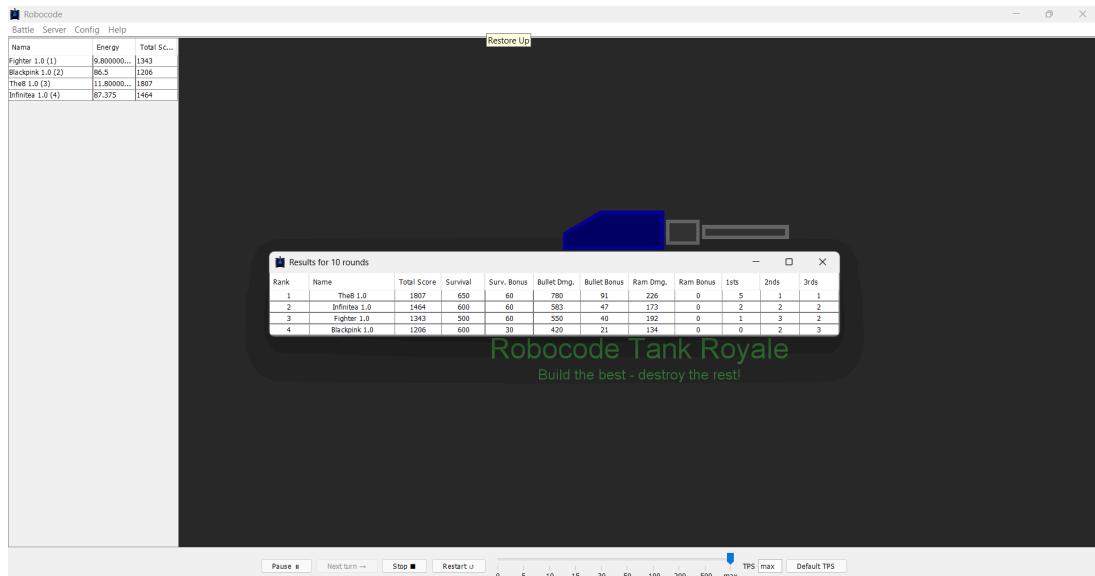
4.3.4 Bot Utama Melawan Seluruh Alternate Bot

Kondisi ketika bot utama menjadi pemenang pertempuran melawan bot alternatif

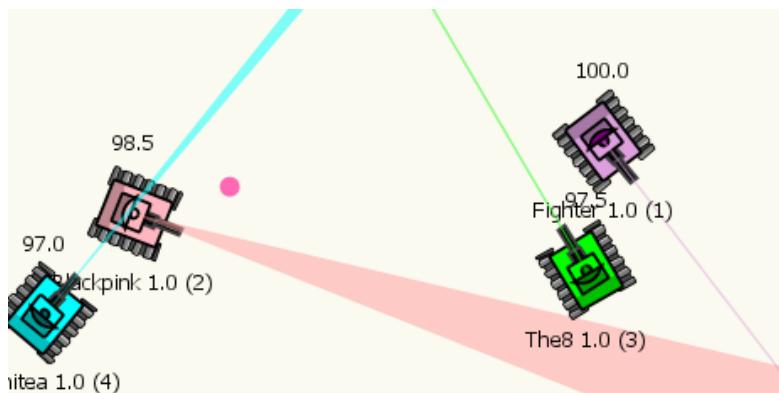


Kondisi ketika bot utama kalah dalam pertempuran melawan bot alternatif

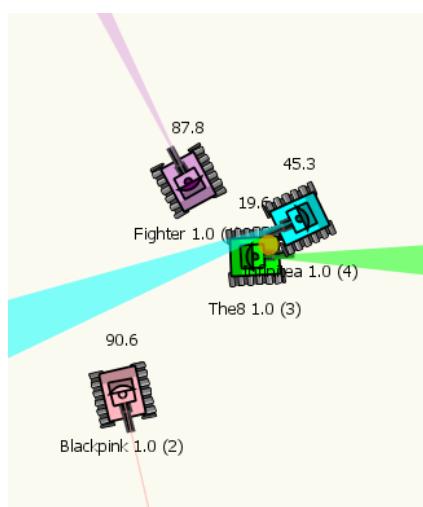




Kondisi ketika bot melakukan *scanning* terhadap bot lain



Kondisi ketika bot melakukan *ram* setelah ditabrak oleh bot lain



Kondisi ketika bot saling melakukan *ram*



4.3.4.1 Analisis Hasil Pengujian Melawan Seluruh Alternatif Bot

Berdasarkan hasil pengujian bot solusi utama melawan seluruh bot alternatif, diperoleh hasil dalam sepuluh *round* bot utama menang empat kali. Dihadapkan dengan musuh yang memiliki tipe serang beragam, performa bot utama masih tergolong baik karena mampu untuk bertahan dan melawan bot musuh. Skor paling banyak diperoleh dari *bullet damage* karena bot utama memang menerapkan strategi *greedy* dalam menembakkan peluru. Jika terdapat banyak musuh dalam satu arena, algoritma pada bot utama akan mengutamakan menembak bot musuh yang dekat dan sudutnya kecil dengan kekuatan lebih besar. Selain dari skor menembak, bot utama juga mendapatkan skor dari *ram damage* karena bot utama akan menyerang balik musuh yang menabraknya dengan melakukan *ram* ketika energi masih mencukupi. Bot utama memperoleh skor dari segala aspek karena memang algoritma pada bot utama menggabungkan beberapa strategi dari bot alternatif yang kemudian dimodifikasi untuk disesuaikan lagi.

Meskipun memiliki strategi *greedy* yang lebih baik di beberapa aspek, namun ada kalanya bot utama mengalami kekalahan ketika bertempur dengan bot alternatif. Hal tersebut sangat mungkin untuk terjadi mengingat beberapa faktor seperti faktor keberuntungan peletakan bot di dalam arena. Selain itu, karena bot utama merupakan perpaduan dari beberapa strategi yang ada di bot alternatif maka sangat masuk akal apabila bot utama melawan seluruh alternatif bot maka akan kalah karena dia diserang oleh musuh yang memiliki strategi yang dia punya juga. Akan tetapi, karena bot yang akan dilawan oleh bot utama adalah bot yang dibuat oleh orang lain, maka peluang bot utama infinitea untuk menang cukup besar sebab menggabungkan beberapa strategi yang bagus.

BAB V

KESIMPULAN DAN SARAN

5.1. Kesimpulan

Robocode adalah permainan pemrograman yang bertujuan untuk membuat kode bot dalam bentuk tank virtual untuk berkompetisi melawan bot lain di arena. Pada Tugas Besar 1 ini, tim pengembang bertugas membuat bot dengan menggunakan algoritma greedy, yaitu metode pemecahan masalah yang memilih pilihan terbaik di setiap langkahnya dengan harapan bahwa pilihan tersebut akan mengarah pada solusi optimal secara keseluruhan.

Pada implementasi ini, strategi greedy yang digunakan adalah “*greedy by adaptability*” dan “*fire by considering distance, angle and energy*”. Algoritma greedy ini mempertimbangkan kekuatan tembak berdasarkan *current position*-nya, yaitu dengan menghitung jarak dan sudut dengan posisi musuh. Algoritma ini juga mampu melakukan pengelolaan energi yang baik dengan cara mempertimbangkan kapasitas energi ketika akan melakukan serangan. Selain itu, algoritma ini juga memiliki pola pergerakan yang unik untuk menghindar dari serangan musuh serta memperbesar area scanning radar dalam rangka mencari musuh.

Berdasarkan hasil pengujian, dapat dibuktikan bahwa strategi greedy yang digunakan sudah cukup efektif dan efisien dalam mengalahkan musuh.

5.2. Saran

Berikut beberapa saran oleh tim pengembang yang berguna untuk memaksimalkan potensi pengembangan selanjutnya:

1. Mengembangkan algoritma yang dapat memprediksi gerak musuh agar dapat menembak peluru dengan lebih presisi.
2. Pertimbangkan peletakan bot pada map pada awal permainan, karena peletakan bot akan sangat berpengaruh terhadap efektivitas strategi yang digunakan.

LAMPIRAN

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

- Tautan *repository* GitHub : naylzhra/Tubes1_Infinitea
- Tautan video YouTube : <https://youtu.be/O0IeXVXoT6E?feature=shared>

DAFTAR PUSTAKA

- Munir, R. *Algoritma Greedy - Bagian 1*, 2025. [Daring]. Tersedia: [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). [Diakses: 24-Mar-2025].
- Munir, R. *Algoritma Greedy - Bagian 2*, 2025. [Daring]. Tersedia: [https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2024-2025/05-Algoritma-Greedy-\(2025\)-Bag2.pdf](https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2024-2025/05-Algoritma-Greedy-(2025)-Bag2.pdf). [Diakses: 24-Mar-2025].
- Munir, R. *Algoritma Greedy - Bagian 3*, 2025. [Daring]. Tersedia: [https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2024-2025/06-Algoritma-Greedy-\(2025\)-Bag3.pdf](https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2024-2025/06-Algoritma-Greedy-(2025)-Bag3.pdf). [Diakses: 24-Mar-2025].