

IF2211 – Strategi Algoritma
TUGAS BESAR 1
“Algoritma Greedy untuk Robocode Tank Royale”



Disusun Oleh:

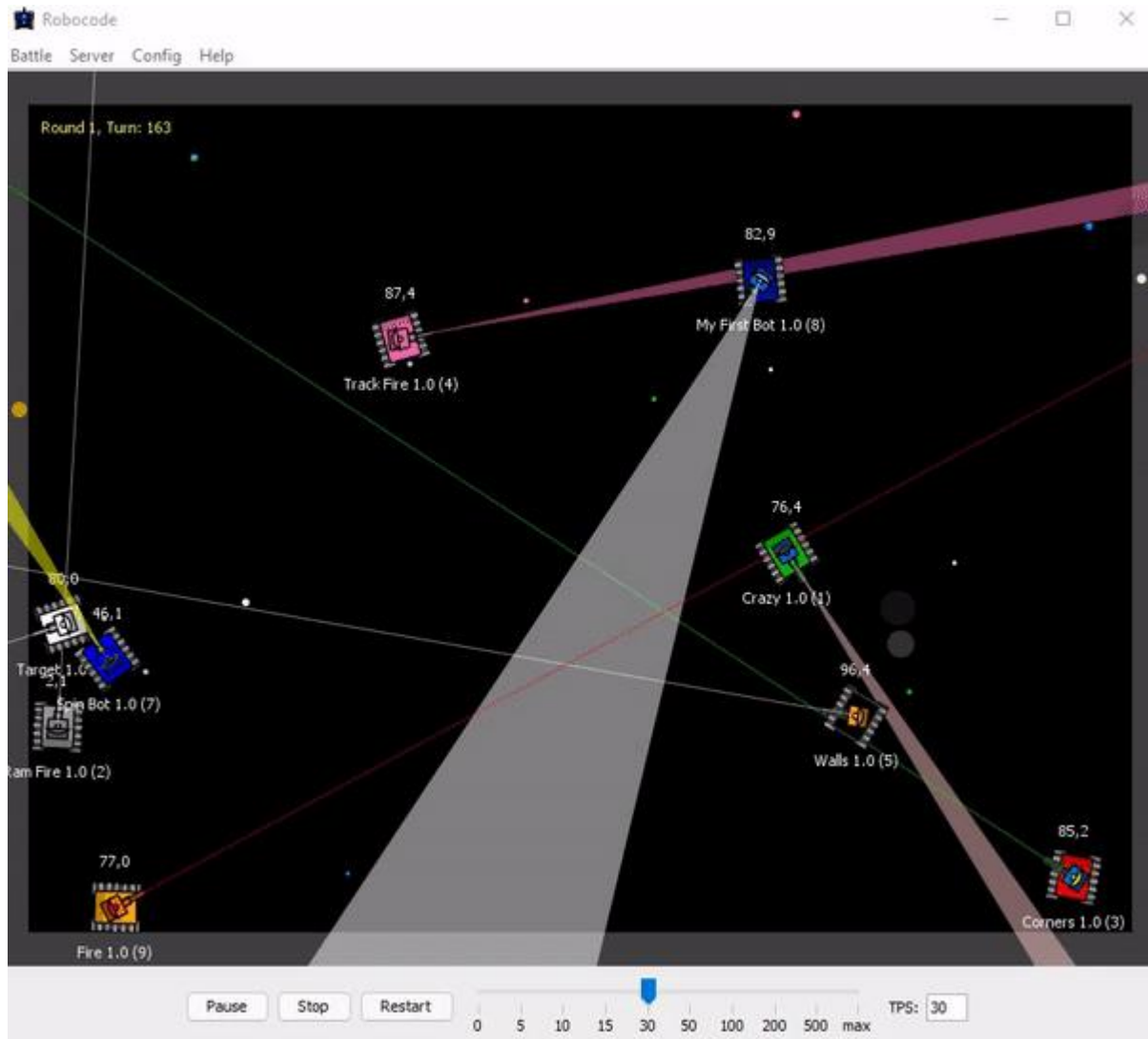
“PistonVarioBoreUp”

Fajar Kurniawan	13523027
Rafizan Muhammad Syawalazmi	13523034
Salman Hanif	13523056

Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung
Bandung 2025

BAB I

DESKRIPSI TUGAS



Gambar 1 Robocode Tank Royale

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

berisi instruksi tentang cara bot bergerak, mendeteksi bot lawan, menembakkan senjatanya, serta bagaimana bot bereaksi terhadap berbagai kejadian selama pertempuran.

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

Komponen-komponen dari permainan ini antara lain:

1. Rounds dan Turns

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

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

Pada setiap turn, sebuah bot dapat:

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

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

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

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

2. Batas Waktu Giliran

Penting untuk dicatat bahwa setiap bot memiliki batas waktu untuk setiap turn yang disebut turn timeout, biasanya antara 30-50 ms (dapat diatur sebagai aturan pertempuran). Ini berarti bahwa bot tidak bisa mengambil waktu sebanyak yang mereka inginkan untuk bergerak dan menyelesaikan turn saat ini.

Setiap kali turn baru dimulai, penghitung waktu ulang diatur ulang dan mulai berjalan. Jika batas waktu tercapai dan bot tidak mengirimkan pergerakannya untuk turn tersebut, maka tidak ada perintah yang dikirim ke server. Akibatnya, bot akan 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.

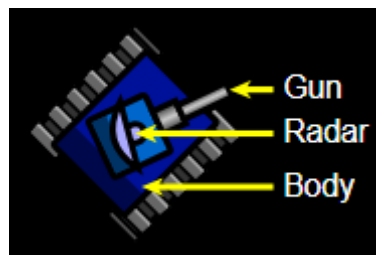
6. Tabrakan

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 pengereman dengan perlambatan maksimum 2 unit per giliran. Percepatan dan perlambatan maksimum tidak bergantung pada kecepatan bot saat itu.

9. Berbelok

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

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

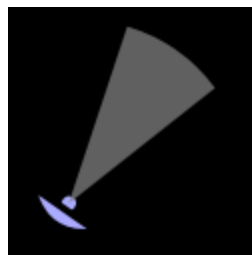
Bagian paling lambat adalah tubuh tank, yang dalam kondisi terbaik dapat berputar hingga 10 derajat per giliran. Namun, ini bergantung pada kecepatan bot saat ini. Semakin cepat bot bergerak, semakin lambat kemampuannya untuk berbelok.

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

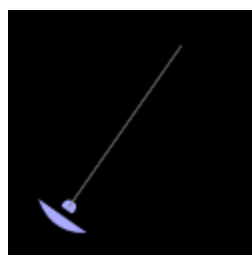
10. Pemindaian

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

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



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



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

11. Skor

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

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

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

BAB II

LANDASAN TEORI

2.1 ALGORITMA GREEDY

Algoritma *Greedy* adalah pendekatan dalam pemrograman yang memecahkan persoalan optimasi dengan cara yang tampaknya rakus. Pendekatan ini berfokus pada pengambilan keputusan sekarang dengan harapan bahwa setiap langkah akan membawa kita lebih dekat ke solusi akhir yang optimal.

Algoritma *greedy* adalah algoritma yang memecahkan persoalan secara langkah per langkah (*step by step*) sedemikian sehingga, pada setiap langkah: mengambil pilihan yang terbaik yang dapat diperoleh pada saat itu tanpa memperhatikan konsekuensi ke depan (prinsip *take what you can get now!*) dan berharap bahwa dengan memilih optimum lokal pada setiap langkah akan berakhir dengan optimum global.

Kegunaan utama dari algoritma greedy adalah untuk menemukan solusi optimal dalam persoalan optimasi dengan cepat. Pendekatan ini sangat berguna dalam banyak kasus di mana kita perlu memaksimalkan atau meminimalkan sesuatu dengan cara yang efisien. Contoh penerapannya termasuk perencanaan jadwal, pengkodean data, manajemen sumber daya, dan banyak lagi

Elemen-elemen algoritma greedy:

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

Dengan menggunakan elemen-elemen di atas, maka 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 CARA KERJA PROGRAM SECARA UMUM

Robocode adalah sebuah permainan pemrograman di mana pemain dapat mengembangkan robot dalam bahasa Java untuk bertarung dalam arena simulasi. Setiap robot harus memiliki strategi yang optimal agar dapat bertahan dan mengalahkan lawan. Salah satu pendekatan dalam mengembangkan strategi adalah dengan menggunakan **algoritma Greedy**, di mana robot akan selalu mengambil keputusan terbaik pada setiap langkah berdasarkan parameter yang tersedia.

Menjalankan Bot di Robocode :

- Instalasi Robocode
 1. Download starter Pack yang tersedia pada link berikut:
<https://github.com/Ariel-HS/tubes1-if2211-starter-pack>
 Download asset dibawah dari release terbaru.
 2. Download jar “robocode-tankroyale-gui-0.30.0.jar”, yang merupakan game engine.
 3. Download dan Ekstrak “TemplateBot.zip” sebagai template bagi bot C#.
 4. (opsional) Anda dapat pula mendownload dan ekstrak source code yang berisi: Sample Bots yang disediakan Robocode. Source Code game engine yang dapat di build ulang menggunakan gradle apabila dibutuhkan.
 5. Jalankan file .jar aplikasi GUI


```
java -jar robocode-tankroyale-gui-0.30.0.jar
```
 6. Ikuti Instruksi di program untuk menjalankan permainan
- Membuat dan Menambahkan Bot
 1. Menggunakan Folder *Template Bot* (ditulis dengan Bahasa C#) yang sudah tersedia yang berisi file-file berikut
 - *.json
 - *.cs
 - *.csproj
 - *.sh
 2. Algoritma dari perilaku bot diterapkan pada file dengan format *.cs. Pada contoh berikut adalah implementasi algoritma tersedia pada *template bot*

```
using System.Drawing;
using Robocode.TankRoyale.BotApi;
using Robocode.TankRoyale.BotApi.Events;
```

```

public class BotTemplate : Bot
{
    // The main method starts our bot
    static void Main(string[] args)
    {
        new BotTemplate().Start();
    }
    // Constructor, which loads the bot config file
    BotTemplate() : base(BotInfo.FromFile("BotTemplate.json")) { }
    // Called when a new round is started -> initialize and do some movement
    public override void Run()
    {
        // Repeat while the bot is running
        while (IsRunning)
        {
            Forward(100);
            TurnGunRight(360);
            Back(100);
            TurnGunRight(360);
        }
    }
    // We saw another bot -> fire!
    public override void OnScannedBot(ScannedBotEvent evt)
    {
        Fire(1);
    }
    // We were hit by a bullet -> turn perpendicular to the bullet
    public override void OnHitByBullet(HitByBulletEvent evt)
    {
        // Calculate the bearing to the direction of the bullet
        var bearing = CalcBearing(evt.Bullet.Direction);
        // Turn 90 degrees to the bullet direction based on the bearing
        TurnLeft(90 - bearing);
    }
}

```

```
}  
}
```

3. Baca dokumentasi API bot dari robocode.sourceforge.io/docs/robocode untuk melakukan implementasi dari fungsi-fungsinya lebih lanjut

Cara Kerja Robocode :

Robocode bekerja berdasarkan konsep *event-driven programming*, di mana setiap bot merespons berbagai event dalam simulasi. Beberapa event penting dalam Robocode meliputi:

1. `onScannedRobot(ScannedRobotEvent e)`: kondisi yang dipanggil saat bot mendeteksi musuh. Berisi serangkaian implementasi fungsi API bot.
2. `onHitByBullet(HitByBulletEvent e)`: kondisi yang dipanggil saat bot terkena tembakan. Berisi serangkaian implementasi fungsi API bot.
3. `onHitWall(HitWallEvent e)`: kondisi yang dipanggil saat bot menabrak dinding. Berisi serangkaian implementasi fungsi API bot.

Setiap bot memiliki metode `run()`, yang berisi loop utama untuk mengontrol pergerakan dan tindakan bot dalam pertempuran.

Implementasi Algoritma Greedy pada Bot :

Algoritma Greedy diterapkan dalam strategi bot untuk selalu memilih keputusan yang memberikan keuntungan maksimal pada setiap langkah. Strategi yang digunakan meliputi:

- Menembak musuh dengan jarak terdekat.
- Memilih kekuatan tembakan berdasarkan jarak dan energi musuh.
- Menghindari tembakan musuh dengan pergerakan yang cepat dan acak.

BAB III

APLIKASI STRATEGI GREEDY

3.1 Alternatif Greedy

3.1.1. Greedy by Distance

Greedy by jarak adalah strategi *greedy* yang mengutamakan menjaga jarak “aman” terhadap musuh yang ditarget. Jarak yang dianggap aman atau optimal di sini adalah jarak yang cukup dekat hingga memungkinkan bot pemain menembakkan pelurunya dengan cukup akurat tanpa meleset, tetapi jaraknya juga harus cukup jauh agar tidak mudah di-*ram* atau ditabrak/diseruduk oleh bot musuh.

a) Mapping Elemen *Greedy*

- i. Himpunan Kandidat: Semua aksi yang tersedia (pergerakan, penembakan, arah, kekuatan peluru, dsb.)
- ii. Himpunan Solusi: Pergerakan yang menjaga jarak aman dari musuh sambil menembakkan peluru.
- iii. Fungsi Solusi: Memeriksa apakah jarak atau pergerakan ke lawan memenuhi kriteria jarak aman, jika bertabrakan dengan musuh segera mundur sambil belok agar mengurangi kemungkinan terperjokkan ke dinding.
- iv. Fungsi Seleksi: Memilih pergerakan (maju/mundur) dengan jarak tertentu agar tetap berada di jarak aman.
- v. Fungsi Kelayakan: Memeriksa apakah jarak dengan musuh membuat pemain perlu maju atau mundur seberapa jauh dan peluru jenis apa yang ditembakkan.
- vi. Fungsi Obyektif: Jarak yang dijaga antara bot pemain dengan musuh adalah jarak aman.

b) Analisis Efisiensi Solusi

Pada strategi *greedy* ini, jarak yang dianggap optimal/aman adalah suatu konstan sehingga perbandingan yang dilakukan untuk mengetahui apakah jarak saat ini antara bot pemain dengan bot musuh sudah dianggap aman akan selalu konstan.

Begitu pula perbandingan untuk menentukan jarak yang harus ditempuh oleh bot pemain. Perbandingan yang dilakukan selalu konstan. Algoritma untuk menentukan sudut pergerakan dan sudut penembakan adalah konstan. Algoritma untuk menentukan kekuatan peluru yang ditembakkan juga selalu konstan

Secara umum, kompleksitas dari seluruh algoritma pada strategi greedy ini adalah $O(1)$ karena semuanya pada dasarnya hanyalah perbandingan konstan.

c) Analisis Efektivitas Solusi

Strategi ini efektif apabila:

1. Bot musuh diam di suatu tempat
2. Bot musuh tidak banyak bergerak
3. Bot musuh banyak bergerak tapi kecepatannya lambat
4. Bot musuh melakukan *ramming* tapi tidak terlalu agresif dan tidak berusaha memojokkan bot pemain ke dinding

Strategi ini tidak efektif jika:

1. Bot musuh sangat agresif dalam *me-ram* bot pemain ke dinding
2. Bot musuh memiliki algoritma penembakan yang akurat

3.1.2. Greedy by Survival with Dodge and Ram

Greedy by dodge adalah strategi algoritma greedy yang memfokuskan untuk bertahan hidup dan meninggikan point survival, tetapi memiliki pola perilaku yang berbeda ketika menyisakan satu musuh nantinya. *Dodge* pada strategi ini dilakukan dengan melakukan *dancing*, yaitu gerakan pola acak yang sulit ditebak dan bergerak cepat sehingga pergerakannya dapat menghindari berbagai peluru lawan. Pola perilaku yang berbeda saat musuh tersisa satu adalah mengunci satu musuh sebagai sasaran, kemudian fokus menyerangnya dengan *damage ram* dan *fire*.

a) Mapping Elemen *Greedy*

- i. Himpunan Kandidat: Aksi yang tersedia dalam API game robocode, di antaranya aksi maju, mundur, berbelok, dan penembakan.
- ii. Himpunan Solusi: Pergerakan acak berpola dengan cepat yang menghindari peluru datang sambil menembak dengan strategi *firetactics*. Pergerakan pola perilaku berbeda dengan ram dan fire ketika musuh tersisa satu.
- iii. Fungsi Solusi: Memeriksa ada berapa musuh yang tersisa. Jika jumlah musuh masih banyak fokus survival, melakukan dodge dengan mempertahankan energi. Ketika jumlah musuh sedikit fokus mengeksekusi dengan ram.
- iv. Fungsi Seleksi: Menentukan pola perilaku dari bot tergantung jumlah musuh yang tersisa.

- v. Fungsi Kelayakan: Memeriksa apakah jumlah musuh tersisa dengan energy yang dimiliki saat ini sudah siap untuk mengganti pola perilaku menjadi ram.
- vi. Fungsi Obyektif: Memeriksa apakah pola perilaku ini dengan keadaan musuh saat ini sudah dapat membuat bot bertahan hidup *survival* hingga akhir.

b. Analisis Efisiensi Solusi

Strategi greedy ini awal mulanya langsung memilih pola perilaku pertama, yaitu melakukan pola pergerakan acak (*dancing*) dengan strategi penembakan *firetactics* sambil melakukan *scanning* ketika melakukan pergerakan. Dalam pola perilaku ini, terus-menerus dilakukan scan jumlah musuh tersisa, ketika jumlah musuh tersisa sedikit maka pola perilaku diubah menjadi ram untuk menyelesaikan pertandingan.

Begitu pula perbandingan untuk menentukan jumlah musuh yang tersedia sebelum mengambil Keputusan pola perilaku yang akan dilakukan bot, hanya merupakan perbandingan konstan antara dua nilai. Algoritma *FireTactics* yang diperlukan dalam penentuan tembakan juga konstan. Kompleksitas waktu dari algoritma penentuan perilaku ini adalah $O(1)$.

c. Analisis Efektivitas Solusi

Strategi ini efektif apabila:

1. *Match* dilakukan dengan jumlah bot dalam rentang 4—8 dalam satu permainan.
2. Bot berhasil bertahan hidup dan hanya menyisakan sedikit bot musuh.
3. Bot tidak terlalu banyak dan menyisakan ruang untuk pola perilaku *dancing bot*.

Strategi ini tidak efektif jika:

1. Bot musuh melakukan lock dan ram ketika bot masih dalam pola *dancing*.
2. Bot musuh maju dan berhasil mengunci pergerakan *dancing bot*.

3.1.3. Greedy by Survival Ranking

Greedy by survival ranking adalah strategi algoritma greedy yang berfokus kepada poin yang didapatkan ketika berhasil bertahan hidup. Algoritma ini terinspirasi dari sifat interaksi antar atom bermuatan, di mana suatu atom akan terdorong untuk

menjauhi atom yang sesama jenis (untuk kasus ini, bot akan menjauhi bot yang masih kuat, dinding, dan tengah arena) dan juga terdorong untuk mendekati atom yang berlawanan jenis (bot yang sudah lemah). Dengan strategi ini, bot dapat bertahan sekaligus menghancurkan robot untuk mendapatkan peringkat yang memuaskan.

a) Mapping Elemen *Greedy*

- i. Himpunan Kandidat: Aksi yang tersedia dalam API game robocode, di antaranya aksi maju, mundur, berbelok, dan penembakan.
- ii. Himpunan Solusi: Pergerakan terkalkulasi dari posisi-posisi musuh dan lokasi bot.
- iii. Fungsi Solusi: Memeriksa apakah pergerakan adalah solusi yang tepat berdasarkan dari hitungan yang ada.
- iv. Fungsi Seleksi: Menghitung arah pergerakan yang harus dilakukan berdasarkan kondisi musuh, jarak musuh, dan lokasi bot sekarang menggunakan vektor 2 dimensi.
- v. Fungsi Kelayakan: Memeriksa apakah pergerakan akan mengakibatkan tabrakan atau tidak
- vi. Fungsi Obyektif: Poin akhir yang didapatkan maksimum

b) Analisis Efisiensi Solusi

Strategi greedy ini dimulai dengan melakukan *scanning* memutar dan memasukkan informasi posisi semua bot. Kemudian proses *scanning* akan mengulangi dari mode mutar sama fokus kepada target, sehingga proses *scanning* konstan.

Algoritma penembakan dan pengekeran hanya pengecekan secara konstan. Algoritma pergerakan membutuhkan iterasi pergerakan terakhir untuk semua bot sehingga membuat kompleksitas $O(n)$.

Secara umum, kompleksitas dari seluruh algoritma pada strategi greedy ini adalah $O(n)$ karena membutuhkan iterasi seluruh bot.

c) Analisis Efektivitas Solusi

Strategi ini efektif apabila:

1. Bot musuh agresif
2. Bot musuh memiliki algoritma menyerang robot paling dekat
3. Bot musuh melakukan *ramming*
4. Masih banyak bot yang tersisa

Strategi ini tidak efektif jika:

1. Bot musuh diam di suatu tempat
2. Bot musuh memiliki algoritma penembakan yang akurat dari jauh
3. Bot musuh memiliki algoritma menarget keras
4. Satu lawan satu dengan bot yang masih kuat.

3.1.4. Greedy by Quick Damage

Greedy by quick damage adalah strategi algoritma greedy yang berfokus melakukan damage ke musuh dengan cepat tanpa terlalu membahayakan diri sendiri. Algoritma ini akan menembak musuh paling dekatnya dan berjalan terus dari sisi arena ke sisi yang lain, seolah-olah bot berpantul pada dinding.

a) Mapping Elemen *Greedy*

- i. Himpunan Kandidat: Aksi yang tersedia dalam API game robocode, di antaranya aksi maju, mundur, berbelok, dan penembakan.
- ii. Himpunan Solusi: Pergerakan lurus linear dari satu sisi ke sisi lainnya dan penembakan berdasarkan jarak terpendek.
- iii. Fungsi Solusi: Memeriksa apakah penentuan arah pergerakan sesuai untuk setiap sisi arena
- iv. Fungsi Seleksi: Menentukan arah pergerakan bot ketika berada di ujung arena dan menentukan target musuh yang paling dekat.
- v. Fungsi Kelayakan: Memeriksa apakah pergerakan adalah pergerakan yang valid dan apakah bot mempunyai target.
- vi. Fungsi Obyektif: Poin akhir yang didapatkan maksimum

b) Analisis Efisiensi Solusi

Strategi greedy ini dimulai dengan melakukan *scanning* memutar dan memasukkan informasi posisi semua bot. Kemudian proses *scanning* akan mengulangi dari mode mutar sama fokus kepada target, sehingga proses *scanning* konstan.

Algoritma penembakan dan pengekeran hanya pengecekan secara konstan. Algoritma pergerakan adalah dengan bergerak secara garis lurus hingga ujung arena, sehingga algoritma konstan. Kompleksitas akhir dari strategi ini adalah $O(1)$.

c) Analisis Efektivitas Solusi

Strategi ini efektif apabila:

1. Kumpulan bot musuh di tengah

2. Bot musuh memiliki algoritma penembakan tanpa prediksi

Strategi ini tidak efektif jika:

1. Bot musuh melakukan *ramming*
2. Bot musuh memiliki algoritma penembakan yang dapat memprediksi gerakan.
3. Bot musuh selalu menjaga jaraknya

3.2 Strategi Greedy yang Diimplementasikan

Strategi algoritma yang diimplementasikan pada kasus ini adalah “*greedy by survival ranking*”. Strategi algoritma ini berfokus untuk bertahan hidup sampai akhir pertandingan sembari melakukan langkah efektif di setiap turn-nya. Algoritma yang terinspirasi dari sifat atom ini akan selalu mencari tempat kosong yang jauh dari rata-rata letak musuh sehingga menjauhkan bot dari bahaya serangan *ram* dan mengurangi risiko untuk tertembak. Mengincar bot yang memiliki energi rendah untuk dieksekusi juga meningkatkan poin yang didapat oleh bot ini secara signifikan.

Secara keseluruhan, algoritma greedy dengan heuristik inilah yang dirasa akan menghasilkan performa terbaik dalam pertempuran. Greedy survival ranking ini mengungguli survival with dodge yang mengandalkan gerakan random berpola untuk menghindari serangan musuh, langkah survival dengan sebisa mungkin menjaga jarak dari musuh dengan kalkulasi ala atomic lebih efektif karena risiko diserang musuh menurun drastis.

Pola serangan dari jarak jauh yang melakukan kalkulasi dalam penembakan peluru yang dimiliki strategi algoritma ini dirasa akan menghasilkan manajemen energi yang baik karena serangan dirancang dalam ukuran *bullet* yang lebih ringan dan kencang serta berpotensi mengenai musuh dengan kecepatannya. Keterampilan bot dalam posisi 1 v 1.

BOT	Survival Point Expected	Bullet Point Expected	Ram Point Expected	Total
Greedy by Distance	3	4	1	8
Greedy by Survival with Dodge & Ram	4	2	2	8

Greedy by Survival Ranking	4	4	2	10
Greedy by Quick Damage	3	4	2	9

Dengan hasil analisis dengan poin ekspektasi ini, dipilih Greedy by Survival Ranking dipilih sebagai bot utama.

BAB IV

IMPLEMENTASI DAN PENGUJIAN

4.1 Implementasi Greedy Dalam Pseudocode

➤ Objek Abstrak yang didefinisikan dan digunakan Bot

```
CLASS BotIntel:
  ATTRIBUTE botId
  ATTRIBUTE botHistory

  CONSTRUCTOR BotIntel(id):
    SET botId to id
    SET botHistory to new BotHistory()

CLASS BotHistory:
  ATTRIBUTE history (array of BotHistoryEntry, size 5)
  ATTRIBUTE length

  CONSTRUCTOR BotHistory():
    INITIALIZE history as an array of 5 BotHistoryEntry
    SET length to 0

  METHOD AddEntry(event, time):
    CREATE newEntry as BotHistoryEntry(time, event.Energy, event.X
      event.Y, event.Direction, event.Speed)
    IF length < history size:
      SET history[length] to newEntry
      INCREMENT length
    ELSE:
      SHIFT history left (overwrite oldest entry)
      SET last element of history to newEntry

  METHOD ResetHistory():
    SET length to 0

  METHOD GetMostRecentEntry():
    RETURN history[length - 1]

CLASS BotHistoryEntry:
  ATTRIBUTE Time
  ATTRIBUTE Energy
  ATTRIBUTE Location (Point)
  ATTRIBUTE Direction
  ATTRIBUTE Speed

  CONSTRUCTOR BotHistoryEntry(time, energy, x, y, direction, speed):
    SET Time to time
    SET Energy to energy
    SET Location to new Point(x, y)
```

```
SET Direction to direction
SET Speed to speed

CLASS Point:
  ATTRIBUTE X
  ATTRIBUTE Y

  CONSTRUCTOR Point(x, y):
    SET X to x
    SET Y to y
```

Greedy by Distance

```
WHILE bot is running:
  FIRE minimal power (0.1)
  TURN radar left by very large amount (1000_00000)
  IF first run OR no enemy detected yet:
    TURN left by large amount (10_000 * turnDirection)
  ELSE:
    TURN left by calculated bearing to enemy

  SET distance = distance to enemy
  IF distance > 160:
    MOVE forward by distance/3

  EXECUTE moves
  TURN radar right by very large amount (1000_00000)

ON_SCANNED_BOT(event):
  STORE enemy direction as dir
  STORE enemy X position as eX
  STORE enemy Y position as eY

  IF first scan:
    SET first to false

  DETERMINE turnDirection based on bot's direction and enemy direction:
    IF bot's direction is between 90 and 270:
      IF enemy direction < bot direction:
        turnDirection = -1
      ELSE:
        turnDirection = 1
    ELSE:
      [Complex logic based on quadrant relationships]

  CALL TurnToFaceTargetWithLeadPrediction with enemy position, speed,
  and direction

  CALCULATE distance to enemy
  IF distance < 115:
    FIRE with power 3
```

ELSE IF distance < 145:

FIRE with power 2

ELSE:

FIRE with power 1

IF distance <= 125:

MOVE forward by (distance - 125)

ELSE:

MOVE forward by distance/3

EXECUTE moves

TURN left by (4 * turnDirection)

ON_HIT_BOT(event):

TURN to face the bot that was hit

ADJUST firing power based on remaining energy:

IF energy > 16: **FIRE** with power 3

ELSE IF energy > 10: **FIRE** with power 2

ELSE IF energy > 4: **FIRE** with power 1

ELSE IF energy > 2: **FIRE** with power 0.5

ELSE IF energy > 0.4: **FIRE** with power 0.1

TURN left by 150

MOVE backward by 200

EXECUTE moves

TurnToFaceTarget(x, y):

CALCULATE bearing to target

IF bearing >= 0:

SET turnDirection to 1

ELSE:

SET turnDirection to -1

TURN left by bearing

TurnToFaceTargetWithLeadPrediction(x, y, enemyVelocity, enemyHeading):

CALCULATE distance to enemy

SET bulletSpeed to 8

CALCULATE timeToReachTarget = distance / bulletSpeed

PREDICT enemy's future position:

futureX = x + sin(enemyHeading in radians) * enemyVelocity * timeToReachTarget

timeToReachTarget

futureY = y + cos(enemyHeading in radians) * enemyVelocity * timeToReachTarget

timeToReachTarget

CALCULATE bearing to predicted position

IF bearing >= 0:

SET turnDirection to 1

ELSE:

SET turnDirection to -1

TURN left by bearing

Greedy by Survival with Dodge and Ram

```
WHILE bot is running:
    SET enemiesRemaining to current enemy count

    IF enemiesRemaining >= 3:
        SET isforward to true
        TURN right by 200
        TURN gun left by (360 * 200)
        MOVE forward by 200
        CALL Dancing()

    ELSE IF enemiesRemaining > 2:
        SET isDancing to false
        TURN right by 360
        CALL Dancing()

    ELSE IF targetId = 0:
        DISABLE gun-body adjustment
        DISABLE radar-body adjustment
        TURN left by 360

    ELSE:
        ENABLE radar-body adjustment
        ENABLE radar-gun adjustment
        CALL FocusScan()
        CALL Go()
        DISABLE radar-body adjustment
        DISABLE radar-gun adjustment
        TURN left by 360

ON_SCANNED_BOT(event):
    IF enemy ID is not in botIntels:
        CREATE new BotIntel entry for enemy ID
        ADD scan data to bot history with TurnNumber

    CALCULATE distance to enemy

    IF enemiesRemaining >= 4:
        FIRE with power 2
    ELSE IF enemiesRemaining <= 3 AND bot is not dancing:
        TURN to face enemy
        IF distance > 200:
            CALL FireTactics with distance
            MOVE forward by 150
        ELSE:
            CALL FireTactics with distance
```

```
MOVE backward by 50
ELSE IF enemiesRemaining <= 2:
    SET targetId to enemy ID
    TURN to face enemy
    TURN gun to calculated bearing for enemy
    CALL FireTactics with distance
    MOVE forward by 1000

ON_HIT_WALL(event):
    IF moving forward:
        MOVE backward by 250
    ELSE:
        MOVE forward by 250

DANCING():
    SET isDancing to true
    REPEAT danceCount times:
        SET isforward to true
        TURN right by 100
        MOVE forward by 100
        SET isforward to false
        TURN right by 100
        MOVE backward by 100

FIRE_TACTICS(distance):
    IF distance <= 150:
        FIRE with power 3
    ELSE IF distance <= 500:
        FIRE with power 1

TURN_TO_FACE_TARGET(x, y):
    CALCULATE bearing to target
    IF bearing >= 0:
        SET turnDirection to 1
    ELSE:
        SET turnDirection to -1
    TURN left by bearing

FOCUS_SCAN():
    IF targetId exists in botIntels AND bot has history entries:
        SET BodyColor to Red
        GET last known enemy position from bot history
        CALCULATE direction to enemy

    IF radar bearing to direction is negative:
        TURN radar right by (-calculated radar bearing + 22.5) modulo 360
    ELSE:
        TURN radar right by (-calculated radar bearing - 22.5) modulo 360
```

Greedy by Survival Ranking

```
WHILE bot is running:
  // MOVEMENT
  Set preferred movement to <0,0>
  For every bot in bot intels:
    Calculate the bot influence to preferred movement
    Adjust preferred movement based on the influence
  Adjust preferred movement based on arena walls and middle
  Set bot speed based on how hard the bot needs to turn
  Turn bot's body to preferred movement

  // SHOOTING
  Set aim direction to target
  Turn gun into aim direction
  If gun is near aim direction and aim direction is not outdated:
    If distance from bot to enemy is close:
      Shoot with power 3
    Else
      Shoot with power 1

  // SCAN
  If scan mode = radar:
    Repeat
      Turn radar continuously
    Until radar time is done or have found target
    Set scan mode to focus
  Else If scan mode = focus:
    Repeat
      Turn radar back and forth around the target
    Until focus time is done and enemy is more than 1
    Set scan mode to radar

  If the target has not been seen for a while:
    Reset target
    Set scan mode to radar

ON_SCANNED_BOT(Event):
  Update bot intel and add more informations
  If newest entry's distance is close than the last:
    Set target to newest entry's id
```

Greedy by Quick Damage


```
Set preferred movement to any random angle
WHILE bot is running:
  // MOVEMENT
  Set bot speed based on how hard the bot needs to turn
  Turn bot's body to preferred movement

  // SHOOTING
  Set aim direction to target
  Turn gun into aim direction
  If gun is near aim direction and aim direction is not outdated:
    If distance from bot to enemy is close:
      Shoot with power 3
    Else
      Shoot with power 1

  // SCAN
  If scan mode = radar:
    Repeat
      Turn radar continuously
    Until radar time is done or have found target
    Set scan mode to focus
  Else If scan mode = focus:
    Repeat
      Turn radar back and forth around the target
    Until focus time is done and enemy is more than 1
    Set scan mode to radar

  If the target has not been seen for a while:
    Reset target
    Set scan mode to radar

ON_HIT_WALL (Event):
  If bot is near the left side or the right side of the arena:
    Mirror the x of the preferred direction
  If bot is near the top side or the down side of the arena:
    Mirror the y of the preferred direction

ON_SCANNED_BOT(Event):
  Update bot intel and add more informations
  If newest entry's distance is close than the last:
    Set target to newest entry's id
```

4.2 Penjelasan Struktur Algoritma *Greedy by Survival Rankings*

1) Penjelasan struktur data yang ada pada algoritma ini:

a. BotIntel.

Sebuah tipe data untuk menyimpan informasi tentang bot lawan. Memiliki atribut bot id dan bot history yang menyimpan histori posisi lawan.

b. BotHistory.

Struktur data yang digunakan untuk menyimpan histori pergerakan bot lawan.

Terdiri dari atribut array of BotHistoryEntry dan panjang array yang sudah terisi.

c. BotHistoryEntry.

Struktur data untuk menyimpan satu entri informasi bot lawan dari hasil *scanning*.

Terdiri dari atribut time, energy, location, direction, dan speed.

d. Point.

Struktur data untuk menyimpan koordinat posisi. Terddiri dari atribut x dan y.

2) Penjelasan fungsi dan prosedur yang ada pada algoritma ini:

a. Run()

Fungsi utama pada bot, dijalankan ketika permainan dimulai. Dimulai dengan mengubah warna bot dan melakukan looping memanggil prosedur HandleMovement(), HandleShooting(), dan HandleScan() hingga permainan selesai.

b. HandleMovement()

Fungsi mengatur pergerakan bot. Menghitung arah pergerakan dengan memanggil fungsi CalculateAvoidDirection() dan bergerak ke arah yang sudah ditentukan dengan memanggil prosedur MoveToDirection()

c. CalculateAvoidDirection()

Menghitung dan mengembalikan arah yang optimal untuk menghindari lawan dan tembok arena. Jika terdapat musuh dengan energi rendah, justru bot akan ingin mendekatinya.

d. MoveToDirection()

Prosedur menggerakkan bot ke arah yang diinginkan. Membelokkan bot ke arah yang sesuai serta mengatur kecepatan bot tergantung dengan besarnya belokan.

e. HandleShooting()

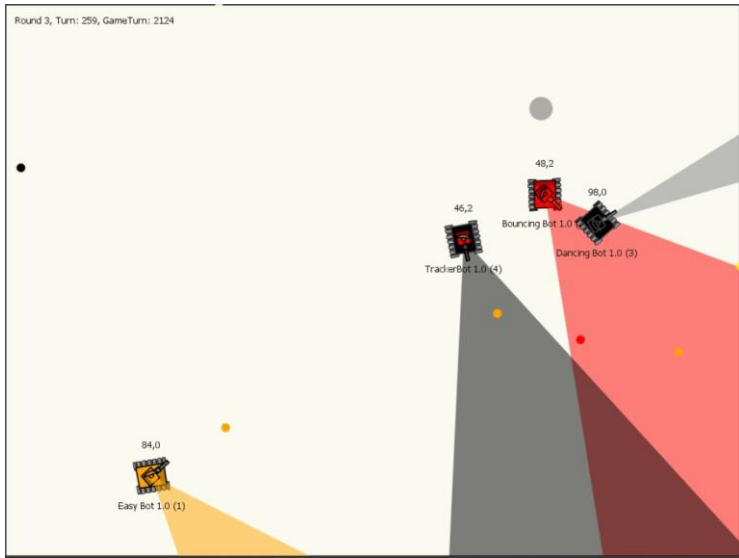
Prosedur mengatur penembakan bot. Memutar tembakan bot hingga sudah arah tembakan sudah tepat. Menembak peluru bertenaga 3 jika musuh cukup dekat dan bertenaga 1 jika musuh jauh.

f. HandleScan()

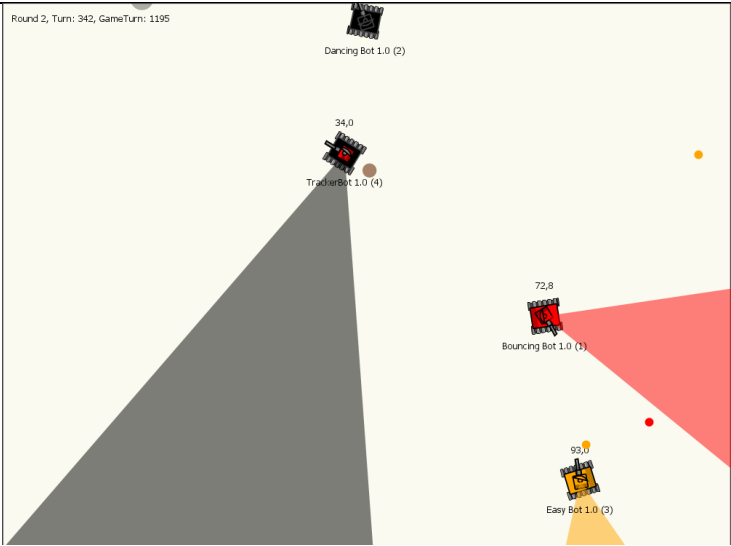
Prosedur mengatur pemindaian *scanner* untuk mendeteksi lawan. Memiliki 2 mode, yaitu: radar dan focus. Radar akan membuat *scanner* berrotasi hingga menemukan target, sedangkan focus akan membuat *scanner* berfokus *scanning* targetnya saja. Kedua mode ini akan saling bergantian saat di permainan.

4.3 Pengujian Perbandingan antar Bot dalam Pertandingan

Dalam pengujian ini, kami melakukan pertandingan 1 v 1 v 1 v 1

No	Pertandingan	Analisis																																																												
1	<div><div>Results for 5 rounds</div><table><thead><tr><th>Rank</th><th>Name</th><th>Total Score</th><th>Survival</th><th>Surv. Bonus</th><th>Bullet Dmg.</th><th>Bullet Bonus</th><th>Ram Dmg.</th><th>Ram Bonus</th><th>1sts</th><th>2nds</th><th>3rds</th></tr></thead><tbody><tr><td>1</td><td>Easy Bot 1.0</td><td>1050</td><td>450</td><td>60</td><td>376</td><td>54</td><td>109</td><td>0</td><td>2</td><td>2</td><td>1</td></tr><tr><td>2</td><td>Dancing Bot 1.0</td><td>781</td><td>400</td><td>30</td><td>292</td><td>15</td><td>22</td><td>22</td><td>1</td><td>2</td><td>1</td></tr><tr><td>3</td><td>Bouncing Bot 1.0</td><td>739</td><td>200</td><td>30</td><td>352</td><td>8</td><td>149</td><td>0</td><td>1</td><td>1</td><td>1</td></tr><tr><td>4</td><td>TrackerBot 1.0</td><td>409</td><td>150</td><td>0</td><td>238</td><td>10</td><td>10</td><td>0</td><td>1</td><td>0</td><td>2</td></tr></tbody></table><div><div>Round 3, Turn: 259, GameTurn: 2124</div></div></div>	Rank	Name	Total Score	Survival	Surv. Bonus	Bullet Dmg.	Bullet Bonus	Ram Dmg.	Ram Bonus	1sts	2nds	3rds	1	Easy Bot 1.0	1050	450	60	376	54	109	0	2	2	1	2	Dancing Bot 1.0	781	400	30	292	15	22	22	1	2	1	3	Bouncing Bot 1.0	739	200	30	352	8	149	0	1	1	1	4	TrackerBot 1.0	409	150	0	238	10	10	0	1	0	2	EasyBot focus menghindari bot lain sehingga kerusakan yang diterima minimal sambil menembak bot lain sehingga lebih bisa survive. Karena ia bisa survive sampai akhir juga sehingga secara overall dia menghasilkan damage lebih banyak kepada ketiga bot lainnya. TrackerBot kalah dibandingkan bot lainnya karena kurang agresif jika dibandingkan bot lain yang lebih sering menembak dengan
Rank	Name	Total Score	Survival	Surv. Bonus	Bullet Dmg.	Bullet Bonus	Ram Dmg.	Ram Bonus	1sts	2nds	3rds																																																			
1	Easy Bot 1.0	1050	450	60	376	54	109	0	2	2	1																																																			
2	Dancing Bot 1.0	781	400	30	292	15	22	22	1	2	1																																																			
3	Bouncing Bot 1.0	739	200	30	352	8	149	0	1	1	1																																																			
4	TrackerBot 1.0	409	150	0	238	10	10	0	1	0	2																																																			
2	<div><div>Results for 5 rounds</div><table><thead><tr><th>Rank</th><th>Name</th><th>Total Score</th><th>Survival</th><th>Surv. Bonus</th><th>Bullet Dmg.</th><th>Bullet Bonus</th><th>Ram Dmg.</th><th>Ram Bonus</th><th>1sts</th><th>2nds</th><th>3rds</th></tr></thead><tbody><tr><td>1</td><td>Easy Bot 1.0</td><td>1290</td><td>700</td><td>120</td><td>344</td><td>36</td><td>72</td><td>17</td><td>3</td><td>2</td><td>0</td></tr><tr><td>2</td><td>Bouncing Bot 1.0</td><td>1157</td><td>350</td><td>0</td><td>528</td><td>32</td><td>233</td><td>14</td><td>1</td><td>3</td><td>1</td></tr><tr><td>3</td><td>Dancing Bot 1.0</td><td>857</td><td>400</td><td>30</td><td>368</td><td>22</td><td>36</td><td>0</td><td>1</td><td>0</td><td>4</td></tr><tr><td>4</td><td>TrackerBot 1.0</td><td>175</td><td>50</td><td>0</td><td>90</td><td>0</td><td>35</td><td>0</td><td>0</td><td>0</td><td>0</td></tr></tbody></table></div>	Rank	Name	Total Score	Survival	Surv. Bonus	Bullet Dmg.	Bullet Bonus	Ram Dmg.	Ram Bonus	1sts	2nds	3rds	1	Easy Bot 1.0	1290	700	120	344	36	72	17	3	2	0	2	Bouncing Bot 1.0	1157	350	0	528	32	233	14	1	3	1	3	Dancing Bot 1.0	857	400	30	368	22	36	0	1	0	4	4	TrackerBot 1.0	175	50	0	90	0	35	0	0	0	0	
Rank	Name	Total Score	Survival	Surv. Bonus	Bullet Dmg.	Bullet Bonus	Ram Dmg.	Ram Bonus	1sts	2nds	3rds																																																			
1	Easy Bot 1.0	1290	700	120	344	36	72	17	3	2	0																																																			
2	Bouncing Bot 1.0	1157	350	0	528	32	233	14	1	3	1																																																			
3	Dancing Bot 1.0	857	400	30	368	22	36	0	1	0	4																																																			
4	TrackerBot 1.0	175	50	0	90	0	35	0	0	0	0																																																			

3



Round 2, Turn: 342, GameTurn: 1195

Dancing Bot 1.0 (2)

34,0

TrackerBot 1.0 (4)

72,8

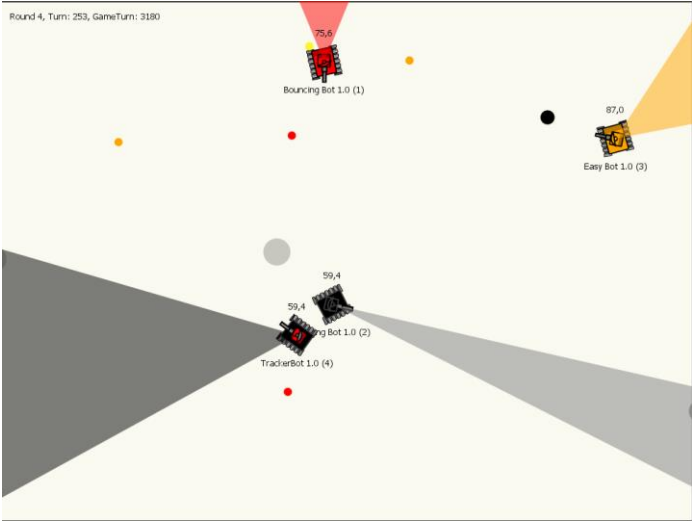
Bouncing Bot 1.0 (1)

95,0

Easy Bot 1.0 (3)

akurat dan sering melakukan ramming untuk memojokkan bot lainnya. DancingBot bergerak dengan sangat dinamis sehingga sulit diprediksi tapi tidak random karena arah pergerakannya juga menjamin kebertahanan hidupnya sambil sangat agresif menembakkan peluru ke banyak bot, kemudian ketika musuh tinggal sedikit ia focus menembak dan ramming memojokkan satu bot supaya bisa mendominasi. Bouncing Bot memantul ke sekeliling arena sehingga bisa mencakup banyak area untuk melakukan serangan sehingga damage yang dihasilkan cukup banyak.

3



Round 4, Turn: 253, GameTurn: 3180

75,6

Bouncing Bot 1.0 (1)

87,0

Easy Bot 1.0 (3)

59,4

59,4

TrackerBot 1.0 (2)

TrackerBot 1.0 (4)

Rank	Name	Total Score	Survival	Surv. Bonus	Bullet Dmg.	Bullet Bonus	Ram Dmg.	Ram Bonus	1sts	2nds	3rds
1	Bouncing Bot 1.0	1265	500	30	532	46	156	0	2	3	0
2	Easy Bot 1.0	1168	650	90	344	23	43	16	3	2	0
3	Dancing Bot 1.0	450	200	0	236	0	14	0	0	0	3
4	TrackerBot 1.0	245	100	0	122	3	19	0	0	0	2

BAB V

KESIMPULAN DAN SARAN

5.1 Kesimpulan

Pada pembuatan bot yang berjalan di sistem program real-time seperti Robocode Tank Royale ini, strategi algoritma greedy memang terbukti sebagai pendekatan yang efektif. Algoritma greedy yang berprinsip “*do the best move now*”, memungkinkan bot untuk mengambil Keputusan secara cepat berdasarkan informasi yang tersedia saat itu tanpa harus mempertimbangkan konsekuensi jangka Panjang. Pendekatan ini dipandang sesuai dengan karakteristik gim ini karena bot harus bereaksi dengan segera terhadap pergerakan dan serangan lawan serta peluang serangan.

Dengan menerapkan algoritma greedy yang telah dikembangkan, bot dapat mengeksekusi keputusan optimal di setiap *turn*-nya sehingga meningkatkan ketepatan pergerakan, efektivitas serangan, serta efisiensi energi untuk bertahan selama pertandingan berlangsung.

Namun, meskipun algoritma greedy memberikan Keputusan yang cepat dan efisien dalam kondisi real-time, strategi ini juga memiliki keterbatasan. Kelemahannya adalah kecenderungan untuk hanya mempertimbangkan keuntungan jangka pendek tanpa mengevaluasi dampak Keputusan terhadap strategi keseluruhan. Dalam beberapa situasi, penempatan awal bot di arena yang bersifat random berpengaruh besar terhadap poin yang didapat dan kemenangan bot.

5.2 Saran

Berbagai strategi greedy dalam pembuatan bot dapat terus dikembangkan untuk meningkatkan performa dalam *match*. Penggabungan dari 2 bot atau lebih algoritma yang memanfaatkan algoritma greedy, memungkinkan bot beralih strategi sesuai situasi. Misalnya, empat strategi greedy yang telah dibahas dapat dikombinasikan untuk meningkatkan efektivitas navigasi dan serangan.

LAMPIRAN DAN DAFTAR PUSTAKA

Repository Github :

https://github.com/salmaanhaniif/Tubes1_PistonVarioBoreUp.git

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 :

Munir, R. 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)

Munir, R. Algoritma Greedy Bagian 2. Diakses pada 24 Maret 2025 dari :

[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)

Munir, R. Algoritma Greedy Bagian 3. Diakses pada 24 Maret 2025 dari :

[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)

<https://robocode-dev.github.io/tank-royale/>