

# **TUGAS BESAR 1**

## **PEMANFAATAN ALGORITMA GREEDY DALAM PEMBUATAN BOT PADA ROBOCODE TANK ROYALE**

**IF-2211**

**STRATEGI ALGORITMA**



**HitnRun**

**13523099  
13523109  
13523109**

**Daniel Pedrosa Wu  
Steven Owen Liauw  
Haegen Quinston**

Dosen Pengampu:

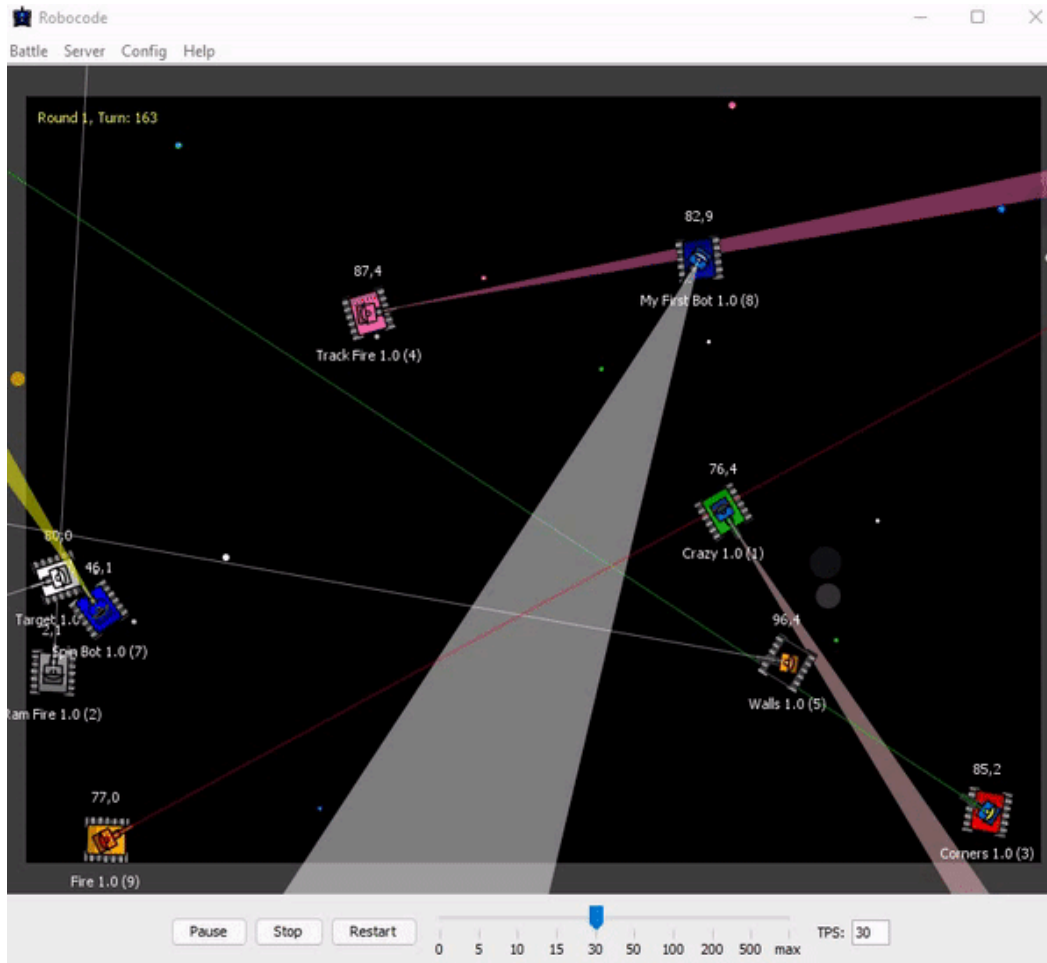
Dr. Nur Ulfa Maulidevi, S.T, M.Sc.  
Dr. Ir. Rinaldi Munir, M.T.  
Monterico Adrian, S.T, M.T.

**PROGRAM STUDI TEKNIK INFORMATIKA  
SEKOLAH TEKNIK ELEKTRO DAN INFORMATIKA  
INSTITUT TEKNOLOGI BANDUNG  
JL. GANESA 10, BANDUNG 40132  
2025**

# DAFTAR ISI

<b>HitnRun.....</b>	<b>1</b>
<b>DAFTAR ISI.....</b>	<b>2</b>
<b>BAB 1 DESKRIPSI TUGAS.....</b>	<b>3</b>
<b>BAB 2 LANDASAN TEORI.....</b>	<b>7</b>
2.1 Algoritma Greedy.....	7
2.2 Cara Kerja Program.....	8
<b>BAB 3 APLIKASI.....</b>	<b>9</b>
3.1 Mapping Persoalan Robocode Tank Royale ke Algoritma Greedy.....	9
3.2 Eksplorasi Alternatif Solusi Greedy.....	10
3.2.1 Pemilihan berdasarkan konsentrasi least risky target.....	10
3.2.2 Pemilihan berdasarkan furthest point.....	10
3.2.3 Pemilihan berdasarkan ramming.....	10
3.2.4 Pemilihan berdasarkan switch wall navigation.....	11
3.3 Analisis Efisiensi & Efektivitas Alternatif Solusi.....	11
3.3.1 Least Risky Target.....	11
3.3.2 Furthest Point.....	11
3.3.3 Ramming.....	11
3.3.4 Switch Wall Navigation.....	12
3.4 Strategi Greedy yang Dipilih.....	12
<b>BAB 4 IMPLEMENTASI &amp; PENGUJIAN.....</b>	<b>12</b>
4.1 Trishula.....	12
4.1.1 Pseudocode.....	12
4.1.2 Penjelasan Fungsi & Procedure.....	15
4.2 N1993R.....	17
4.2.1 Pseudocode.....	17
4.2.2 Penjelasan Fungsi & Procedure.....	23
4.3 Hit n Run Bot.....	25
4.3.1 Pseudocode.....	25
4.3.2 Penjelasan Fungsi & Procedure.....	27
4.4 Forza Ferrari.....	28
4.4.1 Pseudocode.....	28
4.4.2 Penjelasan Fungsi & Procedure.....	30
4.5 Pengujian.....	31
<b>BAB 5 KESIMPULAN.....</b>	<b>33</b>
<b>LAMPIRAN.....</b>	<b>34</b>
<b>DAFTAR PUSTAKA.....</b>	<b>34</b>

## BAB 1 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. Robocode Tank Royale adalah evolusi/versi dari Robocode, 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, menggunakan **strategi greedy** dalam pembuatannya.

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 lawan, 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.

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

Game engine yang akan digunakan pada tugas besar ini tidak mengikuti aturan default mengenai komponen Round & Turns.

## 2. Batas Waktu Giliran

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

Setiap kali turn baru dimulai, penghitung waktu ulang diatur ulang dan mulai berjalan. Jika batas waktu tercapai dan bot tidak mengirimkan pergerakannya untuk turn tersebut, maka tidak ada perintah yang dikirim ke server. Akibatnya, bot akan melewatkan turn tersebut. Jika bot melewatkan turn, ia tidak akan bisa menyesuaikan gerakannya atau menembakkan senjatanya karena server tidak menerima perintah tepat waktu sebelum turn berikutnya dimulai.

## 3. Energi

Semua bot memulai permainan dengan jumlah energi awal sebanyak 100 poin energi.

- Bot akan kehilangan energi jika ditembak atau ditabrak oleh bot lawan.
- Bot juga akan kehilangan energi jika menembakkan meriamnya.
- Bot akan mendapatkan energi jika peluru dari meriamnya mengenai lawan. 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 lawan.

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 lawan.

#### 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.

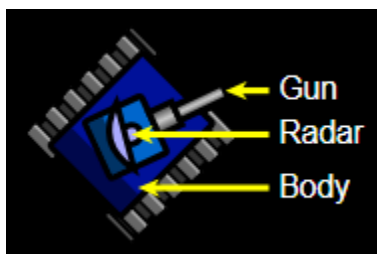
#### 6. Tabrakan

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 lawan 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 lawan 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 tergantung 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.

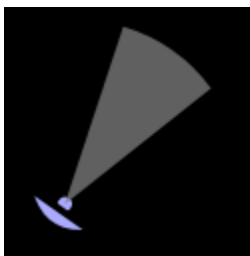
Bagian paling lambat adalah tubuh tank, yang dalam kondisi terbaik dapat berputar hingga 10 derajat per giliran. Namun, ini bergantung pada kecepatan bot saat ini. Semakin cepat bot bergerak, semakin lambat kemampuannya untuk berbelok.

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

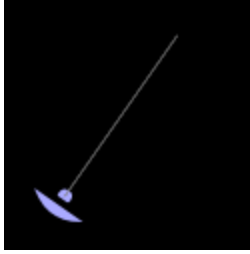
#### 10. Pemindaian

Aspek penting dalam Robocode adalah memindai bot lawan menggunakan radar. Radar dapat mendeteksi bot dalam jangkauan hingga 1200 piksel. Lawan 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 lawan 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 lawan.



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

Skor akhir bot adalah akumulasi dari 6 komponen diatas. Bot yang dianggap menang pertempuran adalah bot dengan akumulasi skor tertinggi.

## BAB 2 LANDASAN TEORI

### 2.1 Algoritma Greedy

Algoritma Greedy adalah pendekatan heuristik yang digunakan untuk menyelesaikan masalah optimasi, yang dapat dikategorikan ke dalam masalah maksimasi atau minimasi. Algoritma ini membuat serangkaian keputusan terbaik secara lokal pada setiap langkah dengan harapan bahwa keputusan tersebut akan menghasilkan solusi optimal secara keseluruhan. Pendekatan ini hanya efektif jika masalah memiliki sifat *greedy choice* (solusi optimal dibangun dari keputusan lokal terbaik) dan *optimal substructure* (solusi optimal dari suatu masalah mengandung solusi optimal dari sub masalahnya).

Sebuah algoritma greedy yang baik memiliki enam komponen utama:

1. **Himpunan Kandidat** – Kumpulan semua elemen yang dapat dipilih untuk membentuk solusi.

2. **Himpunan Solusi** – Subset dari himpunan kandidat yang membentuk solusi yang valid.
3. **Fungsi Seleksi** – Aturan yang menentukan elemen mana yang paling menjanjikan untuk ditambahkan ke dalam himpunan solusi pada setiap langkah.
4. **Fungsi Kelayakan** – Kriteria yang memastikan bahwa elemen yang dipilih tetap mempertahankan validitas solusi.
5. **Fungsi Objektif** – Fungsi yang mengevaluasi kualitas solusi dalam hal optimasi (minimasi atau maksimasi).
6. **Fungsi Solusi** – Kondisi yang menentukan kapan solusi yang lengkap telah terbentuk.

Meskipun algoritma greedy sering digunakan karena kesederhanaannya dan efisiensinya, tidak semua masalah dapat diselesaikan dengan pendekatan ini. Untuk beberapa masalah, metode lain seperti pemrograman dinamis atau backtracking lebih diperlukan untuk mendapatkan solusi optimal.

## 2.2 Cara Kerja Program

Dalam Robocode Tank Royale, bot beroperasi secara otomatis dengan terus-menerus memantau lingkungan sekitar, mengambil keputusan secara cepat, dan menjalankan aksi-aksi yang dinamis secara real-time. Salah satu metode yang efektif untuk mencapai perilaku ini adalah dengan menggunakan algoritma greedy. Dalam konteks ini, algoritma greedy berarti bot selalu memilih tindakan yang memberikan keuntungan langsung paling besar tanpa terlalu memikirkan dampak jangka panjangnya.

Bot mengumpulkan berbagai informasi penting dari sensor radar, seperti posisi lawan, kecepatan, arah, dan jarak antar objek di medan pertempuran. Berdasarkan data tersebut, bot segera menentukan aksi terbaik pada setiap kesempatan. Misalnya, saat radar mendeteksi lawan, algoritma greedy akan mengevaluasi posisi lawan dan memilih target yang paling dekat atau paling rentan, dengan tujuan memaksimalkan peluang tembakan yang sukses. Selanjutnya, bot menghitung sudut dan kekuatan tembakan secara optimal, berdasarkan prediksi pergerakan lawan, tingkat energi, dan jarak target.

Selain aspek ofensif, algoritma greedy juga berperan dalam strategi pertahanan bot. Saat mengevaluasi posisinya di medan tempur—seperti ketika mendekati tembok atau hambatan—bot akan langsung memilih tembok terdekat sebagai perlindungan sementara atau menyesuaikan arah geraknya untuk meminimalkan risiko terkena serangan. Dengan terus mengevaluasi data dari radar dan input lingkungan lainnya, bot dapat segera menyesuaikan arah gerak, rotasi radar, dan posisi senjata untuk mengoptimalkan kemampuan menyerang dan bertahan sekaligus.

Program Robocode Tank Royale dijalankan dengan mengeksekusi file .jar yang telah dikompilasi. Pengguna kemudian mengarahkan aplikasi ke direktori tempat file-file bot disimpan. Bot-bot ini akan otomatis dimuat ke dalam lingkungan pertandingan dan langsung dijalankan. Masing-masing bot kemudian masuk ke dalam loop utama, di mana mereka terus-menerus memperbarui tindakan, mulai dari pemindaian radar, pengaturan arah senjata, hingga pergerakan sesuai strategi greedy yang telah diterapkan. Proses iteratif ini memungkinkan bot merespons perubahan situasi medan tempur dengan cepat dan efisien, sehingga meningkatkan peluang untuk menang dengan selalu memilih tindakan yang paling optimal di setiap keputusan lokal.



## **BAB 3 APLIKASI**

### **3.1 Mapping Persoalan Robocode Tank Royale ke Algoritma Greedy**

Himpunan Kandidat :

- Aksi Pergerakan :
  1. Maju dengan jarak tertentu.
  2. Mundur dengan jarak tertentu.
  3. Berputar ke kiri dengan sudut tertentu.
  4. Berputar ke kanan dengan sudut tertentu.
- Aksi Penembakan :
  1. Menembak dengan fire power tinggi untuk jarak dekat.
  2. Menembak dengan fire power sedang untuk jarak menengah.
  3. Menembak dengan fire power rendah untuk jarak jauh.
- Aksi Radar dan Turret :
  1. Berputar ke kiri dengan sudut tertentu.
  2. Berputar ke kanan dengan sudut tertentu.

Himpunan Solusi :

- Aksi yang dipilih pada setiap turn.

Fungsi Seleksi:

- Mencari bot yang memiliki jarak rata-rata paling jauh dengan bot lain.
- Mencari titik yang memiliki jarak rata-rata paling jauh dengan bot lain untuk meminimalisir risiko kematian.
- Mencari set gerakan dinding terbaik untuk mengecoh tembakan lawan.
- Mengejar bot lawan yang ditemukan pertama kali.

Fungsi Solusi :

- Memeriksa apakah gerakan body berada pada jarak yang aman dari dinding.
- Memeriksa apakah gerakan body berada pada daerah yang aman dari lawan.
- Memeriksa apakah gerakan body sudah/akan menabrak lawan.
- Memeriksa apakah gerakan body menuju lawan yang terisolasi.

- Memeriksa apakah gerakan turret dan radar sudah sesuai dengan arah bot lawan.

Fungsi Kelayakan :

- Memeriksa apakah gerakan tidak menabrak tembok.
- Memeriksa apakah jarak lawan layak ditembak dengan kekuatan tertentu.
- Memeriksa apakah gerakan tidak menuju daerah yang banyak bot lawan.
- Memeriksa apakah tembakan turret tidak melebihi batas per turn.
- Memeriksa apakah arah body sudah sesuai untuk mengejar bot lawan.

Fungsi Objektif:

- Mendapatkan damage output tertinggi.
- Mendapatkan peluang kill lawan tertinggi.
- Memiliki tingkat pengecohkan tertinggi terhadap peluru lawan.
- Bertahan hidup hingga ronde selesai.
- Mendapatkan posisi bot paling strategis pada arena.

## 3.2 Eksplorasi Alternatif Solusi Greedy

### 3.2.1 Pemilihan berdasarkan konsentrasi *least risky target*

Dalam strategi ini, bot menentukan target berdasarkan risiko serangan balik yang paling rendah, sambil melakukan serangan dengan agresif. Target yang dipilih adalah bot lawan yang paling terisolasi atau memiliki jarak rata-rata terbesar terhadap bot lain di arena. Tujuan utama dari strategi ini adalah mengurangi potensi serangan balasan atau pengepungan oleh lawan lain, sekaligus melancarkan serangan yang agresif agar bisa dengan cepat membunuh bot lawan.

### 3.2.2 Pemilihan berdasarkan *furthest point*

Dalam strategi ini, bot menentukan arah gerakan dengan memilih titik yang memiliki jarak rata-rata paling jauh dari semua bot lawan di arena. Lokasi yang dipilih adalah area dengan tingkat ancaman paling rendah, sehingga bot diprediksi mampu bertahan hidup lebih lama. Strategi ini memungkinkan bot secara dinamis berpindah posisi ke wilayah-wilayah yang relatif lebih aman, bergantung pada kondisi lingkungan yang terus berubah.

### 3.2.3 Pemilihan berdasarkan *ramming*

Dalam strategi ini, bot memilih aksi yang menghasilkan tingkat kerusakan tertinggi dalam waktu sesingkat mungkin terhadap lawan. Kerusakan yang dimaksud dapat berupa serangan tembakan peluru (bullet) dan benturan langsung (ramming). Prioritas utama strategi ini adalah dengan cepat menemukan target yang paling optimal untuk diserang, kemudian melakukan aksi ofensif dengan agresif. Fokus strategi ini terletak pada perolehan poin maksimum, terlepas dari durasi bertahan hidup bot di arena.

### 3.2.4 Pemilihan berdasarkan *switch wall navigation*

Dalam strategi ini, bot memilih pola gerak teraman dengan memanfaatkan tembok sebagai elemen pertahanan. Bot secara otomatis mencari tembok terdekat dan bergerak menyusuri tembok untuk menjaga jarak aman dari serangan musuh sambil menembak. Ketika ada bot lawan yang memasuki jarak tertentu yang dianggap sebagai zona serangan, bot akan segera beralih ke taktik ramming, yaitu menyerang langsung lawan dengan menabraknya. Setelah eksekusi ramming atau ketika jarak aman kembali tercapai, bot dengan cepat kembali ke pola gerak teraman dengan kembali bergerak menyusuri tembok sambil menembak. Dengan demikian, strategi ini secara dinamis menggabungkan elemen pertahanan dan serangan, memastikan bot tetap berada dalam posisi strategis yang optimal untuk bertahan sambil memanfaatkan peluang untuk mendapatkan bonus poin ram.

## 3.3 Analisis Efisiensi & Efektivitas Alternatif Solusi

Berikut ini merupakan analisis terhadap efisiensi dan efektivitas dari setiap alternatif strategi greedy yang telah dirumuskan sebelumnya:

### 3.3.1 Least Risky Target

Strategi ini cukup efisien karena keputusan target dibuat berdasarkan evaluasi sederhana mengenai jarak rata-rata antar-bot musuh, yang secara komputasional tidak terlalu kompleks. Proses identifikasi target dengan tingkat isolasi tertinggi tergolong ringan dan dapat dilakukan secara cepat di setiap siklus keputusan bot.

Strategi ini cukup efektif dalam meningkatkan ketahanan hidup bot. Dengan memilih target yang terisolasi, bot mengurangi risiko terjebak dalam kepungan atau serangan dari beberapa musuh sekaligus. Akan tetapi, strategi ini juga memiliki risiko tersendiri, yakni bot yang diincar dapat melakukan serangan balik karena tidak menghadapi ancaman lain selain dari bot pengguna strategi ini.

### 3.3.2 Furthest Point

Walaupun memerlukan perhitungan jarak terhadap semua posisi bot musuh, strategi ini masih tergolong efisien. Komputasi jarak rata-rata dilakukan secara berkala dan tidak terlalu kompleks, sehingga penggunaan sumber daya pemrosesan tetap efektif dan terkelola dengan baik.

Strategi ini sangat efektif dalam meningkatkan peluang bertahan hidup, terutama pada situasi pertempuran di mana sebagian besar bot berperilaku agresif. Kecenderungan bot untuk bergerak menjauh dari keramaian membuatnya dapat bertahan hidup lebih lama serta memungkinkan bot berpindah secara dinamis ke lokasi yang lebih aman. Namun, sisi kelemahannya adalah potensi untuk mencetak skor dari tembakan atau ramming relatif rendah dibandingkan bot yang agresif, karena sifat strategi ini yang lebih defensif.

### 3.3.3 Ramming

Strategi ini sangat efektif untuk menghasilkan skor yang tinggi dalam waktu singkat berkat sifatnya yang agresif dan ofensif. Potensi poin yang bisa diraih sangat tinggi, namun strategi ini memiliki risiko tinggi terhadap kerusakan serius yang mungkin diterima bot akibat aksi agresif yang dilakukannya. Kelemahan utama adalah risiko besar bot terekspos serangan balik ataupun kehilangan energi secara drastis. Oleh karena itu, strategi ini masuk dalam kategori strategi dengan karakteristik **high risk, high reward**.

### 3.3.4 Switch Wall Navigation

Strategi ini cukup efektif dalam menjaga ketahanan hidup bot sekaligus memiliki potensi untuk menghasilkan skor yang cukup tinggi, bergantung pada kondisi medan tempur. Dengan bergerak secara acak di sekitar dinding, bot mampu mengurangi peluang terkena tembakan musuh. Di sisi lain, ketika musuh berada dalam jangkauan dekat, bot dapat beralih menjadi agresif untuk mencetak poin tambahan. Fleksibilitas ini menjadi kekuatan utama strategi ini dalam situasi yang sangat dinamis.

## 3.4 Strategi Greedy yang Dipilih

Dari empat strategi greedy yang telah dianalisis, *Furthest Point* terbukti paling optimal untuk dijadikan strategi utama. Alasannya jelas: sistem skor memberikan +50 poin setiap kali bot bertahan hidup lebih lama dari bot lain yang mati, sedangkan menembak musuh hanya memberi maksimal 15 poin per tembakan. Selain itu, terdapat bonus bagi pemenang ronde berupa  $+10 \cdot n$  poin ( $n$  jumlah bot). Perbedaan ini menunjukkan bahwa bertahan hidup cukup menguntungkan secara jangka panjang.

Strategi agresif seperti *Least Risky Target* dan *Ramming* terlalu bergantung pada keberhasilan serangan yang tidak pasti dan berisiko tinggi karena mudah terkena serangan balik. Akibatnya, skor dari strategi ini cenderung tidak stabil. Sebaliknya, *Furthest Point* fokus menjauh dari zona konflik dan mencari posisi paling aman, membuat bot bisa bertahan lebih lama dan konsisten mengumpulkan poin. Strategi ini juga fleksibel karena bot terus bergerak menyesuaikan kondisi, jadi tak mudah ditebak lawan—kelebihan yang jarang dimiliki strategi defensif lainnya.

Walau strategi *Switch Wall Navigation* juga defensif, *Furthest Point* tetap unggul karena secara aktif menghindari bahaya, bukan hanya bertahan di dekat dinding. Dengan mempertimbangkan sistem skor, karakter strategi, dan risiko, *Furthest Point* adalah pilihan paling ideal untuk strategi utama dalam game ini.

## BAB 4 IMPLEMENTASI & PENGUJIAN

### 4.1 Trishula

#### 4.1.1 Pseudocode

```
BEGIN Trishula BOT

Deklarasi Variabel Global:
  enemies : Dictionary<int, Enemy>
  target : Enemy
  curPos, prevPos, nextPos : PointD
  MIN_POWER, MAX_POWER, BASE_DISTANCE : double
  stopAndGoCounter, currentInterval : integer
  MinMoveInterval, MaxMoveInterval, MinStopInterval, MaxStopInterval :
integer
  isMoving : boolean
  rand : Random

MAIN:
```

Buat instance bot Trishula dan panggil method Start()

CONSTRUCTOR Trishula():

Inisialisasi parameter bot dari file konfigurasi "Trishula.json"

enemies = dictionary kosong

target = Enemy tidak aktif

curPos, nextPos, prevPos = posisi saat ini

MIN\_POWER = 0.1

MAX\_POWER = 3.0

BASE\_DISTANCE = 150.0

stopAndGoCounter = 0

currentInterval = 0

MinMoveInterval = 30

MaxMoveInterval = 180

MinStopInterval = 5

MaxStopInterval = 10

isMoving = true

rand = Random()

FUNCTION Run():

SET semua warna komponen bot (BodyColor, TurretColor, RadarColor, BulletColor, ScanColor, TracksColor, GunColor)

SET AdjustGunForBodyTurn dan AdjustRadarForGunTurn = true

WHILE IsRunning DO:

CALL SetTurnRadarLeft(60)

CALL FindNewTarget()

curPos = posisi bot sekarang

IF TurnNumber > 9 DAN target aktif THEN:

CALL Move(battlefield)

ENDIF

CALL Go()

END WHILE

FUNCTION Move(RectangleD battlefield):

IF (musuh tersisa == 1 DAN jarak ke target > 200) THEN:

IF stopAndGoCounter >= currentInterval THEN:

TOGGLE isMoving

stopAndGoCounter = 0

currentInterval = interval acak berdasarkan kondisi isMoving

ENDIF

IF isMoving THEN:

Bergerak menuju nextPos atau GeneratePoint baru jika perlu

ELSE:

CALL SetForward(0)

ENDIF

stopAndGoCounter++

ELSE:

Bergerak menuju nextPos atau GeneratePoint baru jika perlu

ENDIF

```

FUNCTION GeneratePoint(double distanceToTarget, RectangleD battlefield):
  FOR i = 0 sampai 360 DO:
    test = titik acak berdasarkan jarak dan sudut
    IF battlefield mengandung titik test DAN risiko test < risiko terkecil
DAN jalur aman THEN:
      nextPos = test
    ENDIF
  ENDFOR
  prevPos = curPos

FUNCTION RiskFunction(PointD dest):
  Hitung risiko titik berdasarkan jarak, energi musuh, dan sudut
  RETURN risiko total

FUNCTION FindNewTarget():
  minDistance = ∞
  FOR EACH enemy IN enemies DO:
    IF enemy aktif DAN jarak enemy < minDistance THEN:
      minDistance = jarak enemy
      target = enemy
    ENDIF
  ENDFOR

FUNCTION OnScannedBot(event e):
  Tambahkan atau perbarui enemy dalam dictionary enemies
  CALL FindNewTarget()
  IF target.id == e.ScannedBotId THEN:
    Hitung firePower berdasarkan jarak dan energi
    CALL LinearTargeting(firePower, angleTolerance)
  ENDIF

FUNCTION LinearTargeting(double bulletPower, double angleTolerance):
  Hitung posisi prediksi musuh
  SET sudut senjata ke posisi prediksi
  IF GunHeat == 0 DAN GunTurnRemaining <= 5 THEN:
    CALL SetFire(bulletPower)
  ENDIF

FUNCTION HeadOnTargeting(double bulletPower, double angleTolerance):
  SET sudut senjata langsung ke posisi musuh
  IF GunHeat == 0 DAN GunTurnRemaining <= 5 THEN:
    CALL SetFire(bulletPower)
  ENDIF

FUNCTION OnBotDeath(event e):
  SET enemy menjadi tidak aktif
  CALL FindNewTarget()

```

```

FUNCTION OnHitBot(event e):
    target = enemy yang ditabrak
    CALL HeadOnTargeting(3, 5)
    CALL GeneratePoint( $\infty$ , battlefield)

FUNCTION OnHitByBullet(event e):
    IF jarak ke target < 100 ATAU kekuatan peluru > 1.5 THEN:
        CALL HeadOnTargeting(3, 5)
        CALL GeneratePoint( $\infty$ , battlefield)
    ENDIF

END Trishula BOT

```

#### 4.1.2 Penjelasan Fungsi & Procedure

##### 1. Run()

- Fungsi utama yang terus-menerus berjalan selama pertandingan berlangsung.
- Mengatur warna seluruh komponen bot serta parameter AdjustGunForBodyTurn dan AdjustRadarForGunTurn agar pergerakan senjata dan radar independen.
- Melakukan pemindaian radar secara kontinu dengan sudut terbatas (60 derajat) untuk efisiensi.
- Memilih target terbaik dengan memanggil FindNewTarget().
- Jika sudah memasuki putaran tertentu dan target aktif ditemukan, bot memanggil fungsi Move() untuk bergerak secara strategis.
- Melaksanakan seluruh perintah yang telah diatur menggunakan Go().

##### 2. Move(RectangleD battlefield)

- Fungsi ini mengatur logika pergerakan bot, menggunakan teknik stop-and-go terutama saat pertandingan menyisakan hanya satu musuh (1v1).
- Menghitung interval berhenti dan bergerak secara acak untuk mengurangi prediktabilitas gerakan bot.
- Jika sedang bergerak, bot akan menuju posisi tujuan (nextPos) atau memanggil GeneratePoint() untuk mencari posisi baru yang lebih aman.
- Memastikan bot selalu berada dalam batas aman dari medan pertempuran yang ditentukan.

##### 3. GeneratePoint(double distanceToTarget, RectangleD battlefield)

- Menghasilkan titik tujuan pergerakan baru secara acak berdasarkan jarak ke musuh dan pertimbangan keamanan posisi.
- Memilih titik dengan risiko terkecil berdasarkan perhitungan dari RiskFunction().

- Memastikan jalur menuju titik tujuan baru tidak terhalang oleh bot musuh lain dan tetap berada dalam batas arena.

#### **4. RiskFunction(PointD dest)**

- Menghitung risiko dari sebuah titik berdasarkan jarak terhadap bot musuh, rasio energi antara musuh dan bot sendiri, serta sudut antar posisi.
- Mengembalikan nilai risiko yang digunakan untuk menentukan titik tujuan terbaik dengan risiko terkecil.

#### **5. FindNewTarget()**

- Mengidentifikasi dan memilih musuh terdekat dari bot.
- Mengatur musuh tersebut sebagai target utama untuk serangan.

#### **6. OnScannedBot(ScannedBotEvent e)**

- Fungsi yang dipanggil saat radar bot mendeteksi keberadaan bot musuh.
- Memperbarui data musuh dalam dictionary atau menambahkan musuh baru jika sebelumnya belum pernah terdeteksi.
- Memanggil FindNewTarget() untuk memperbarui target utama bot.
- Jika musuh yang terdeteksi adalah target utama, bot menghitung kekuatan tembakan (firePower) yang optimal, kemudian melakukan serangan dengan memanggil metode targeting (misalnya LinearTargeting()).

#### **7. LinearTargeting(double bulletPower, double angleTolerance)**

- Fungsi targeting prediktif yang menghitung posisi masa depan musuh berdasarkan kecepatan dan arahnya saat ini.
- Mengarahkan senjata ke posisi yang telah diprediksi dan menembak jika posisi senjata tepat (GunHeat = 0 dan GunTurnRemaining  $\leq$  5 derajat).

#### **8. HeadOnTargeting(double bulletPower, double angleTolerance)**

- Fungsi targeting sederhana dengan mengarahkan senjata langsung ke posisi musuh tanpa prediksi.
- Digunakan dalam kondisi khusus, seperti saat musuh berhenti atau pada jarak yang sangat dekat.
- Melakukan penembakan langsung begitu posisi senjata siap.

#### **9. OnBotDeath(BotDeathEvent e)**

- Fungsi yang aktif saat bot musuh mati.
- Memperbarui status musuh yang mati menjadi tidak aktif dalam dictionary.



- Mengubah target ke musuh terdekat lainnya yang masih aktif dengan memanggil FindNewTarget().

#### **10. OnHitBot(HitBotEvent e)**

- Fungsi yang aktif ketika bot menabrak bot musuh.  
Bot langsung mengarahkan senjata dan radar ke musuh yang tertabrak.
- Melakukan tembakan kuat dengan metode targeting langsung (head-on).
- Bot kemudian segera memilih posisi baru yang lebih aman dengan memanggil GeneratePoint().

#### **11. OnHitByBullet(HitByBulletEvent e)**

- Fungsi ini aktif saat bot terkena peluru lawan.
- Jika jarak ke musuh sangat dekat ( $< 100$  unit) atau peluru lawan memiliki kekuatan besar ( $> 1.5$ ), bot segera melakukan balasan agresif dengan targeting langsung (head-on).
- Bot juga akan memanggil GeneratePoint() untuk segera pindah ke posisi yang lebih aman.

#### **12. Class Enemy**

Kelas ini digunakan untuk menyimpan data lengkap dari bot musuh yang telah dipindai oleh radar bot.

##### **Atribut:**

- active : boolean  
Menyatakan apakah bot musuh masih aktif (hidup) atau sudah mati.
- id : integer  
Identifier unik dari bot musuh.
- location, prevLocation : PointD  
Menyimpan posisi saat ini dan posisi sebelumnya dari bot musuh.
- energy, prevEnergy : double  
Energi saat ini dan energi sebelumnya dari bot musuh.
- speed, prevSpeed : double  
Kecepatan gerak saat ini dan kecepatan sebelumnya dari bot musuh.
- direction, prevDirection : double  
Arah gerak saat ini dan arah gerak sebelumnya dari bot musuh.
- scanTime, prevScanTime : integer  
Waktu terakhir kali bot musuh terdeteksi oleh radar dan waktu sebelumnya.

##### **Constructor:**

- Menginisialisasi semua atribut dari bot musuh saat pertama kali terdeteksi.

### 13. Class PointD

Kelas ini digunakan untuk merepresentasikan koordinat posisi dalam bentuk dua dimensi (X,Y) dengan presisi tipe data double.

#### Atribut:

- X : double  
Posisi horizontal di arena.
- Y : double  
Posisi vertikal di arena.

#### Constructor:

- PointD(double X, double Y)  
Membuat titik koordinat baru dengan posisi X dan Y yang diberikan.

#### Method:

- DistanceTo(double x, double y)  
Menghitung jarak Euclidean dari posisi saat ini ke koordinat lain.

### 14. Class RectangleD

Kelas ini digunakan untuk merepresentasikan area persegi panjang pada arena dengan presisi tipe data double, digunakan untuk membatasi area pergerakan bot.

#### Atribut:

- x, y : double  
Posisi sudut kiri atas persegi panjang.
- width, height : double  
Lebar dan tinggi area persegi panjang.

#### Constructor:

- RectangleD(double x, double y, double width, double height)  
Menginisialisasi persegi panjang dengan posisi awal serta ukuran yang diberikan.

#### Method:

- contains(double x, double y)  
Memeriksa apakah suatu titik dengan koordinat (x,y) berada di dalam area persegi panjang atau tidak.

## 4.2 N1993R

### 4.2.1 Pseudocode

BEGIN N1993R BOT

Deklarasi : *{variable global}*  
 isAligning, Hostile : boolean  
 SafeDistance, firePower : double  
 currentWall : string  
 lastScannedTime, scanTimeout : long

MAIN:

Buat instance bot N1993R dan panggil method Start()

CONSTRUCTOR N1993R():

Inisialisasi parameter bot dari file konfigurasi "N1993R.json"

isAligning, Hostile = false  
 safeDistance = 35  
 lastScannedTime = 0  
 scanTimeout = 8

FUNCTION Run():

Deklarasi: *{Tidak ada variable Lokal}*

Algoritma:

Set BodyColor, GunColor, RadarColor, TurretColor, BulletColor,  
 ScanColor = Black

lastScannedTime = Environment.TickCount *{waktu sekarang}*

CALL SetTurnRadarLeft(360)

WHILE IsRunning DO:

IF isAligning == false THEN

CALL AlignWithWall()

ENDIF

CALL MoveAlongWall()

CALL Go() *{Eksekusi perintah melalui API Robocode}*

END WHILE

FUNCTION AlignWithWall():

Deklarasi:

distanceToLeft, distanceToRight, distanceToBottom, distanceToTop,  
 currentHeading : double

distances : dictionary<string, number>

```

Algoritma:
IF (Environment.TickCount - lastScannedTime > scanTimeout) THEN
    CALL SetTurnRadarLeft(360)
ENDIF
distanceToLeft = X
distanceToRight = ArenaWidth - X
distanceToBottom = Y
distanceToTop = ArenaHeight - Y
    Buat dictionary distances dengan { "Left": distanceToLeft, "Right":
distanceToRight,
                                     "Top": distanceToTop, "Bottom":
distanceToBottom }
    currentWall = key pada distances dengan nilai minimum
    currentHeading = Direction
    SWITCH (currentWall):
        CASE "Right":
            IF currentHeading <= 180 THEN
                CALL TurnRight(currentHeading)
            ELSE
                CALL TurnLeft(360 - currentHeading)
            ENDIF
            IF distanceToRight > SafeDistance THEN
                CALL Forward(distanceToRight - SafeDistance)
            ENDIF
            CALL TurnRight(90)
        CASE "Left":
            IF currentHeading <= 180 THEN
                CALL TurnLeft(180 - currentHeading)
            ELSE
                CALL TurnRight(currentHeading - 180)
            ENDIF
            IF distanceToLeft > SafeDistance THEN
                CALL Forward(distanceToLeft - SafeDistance)
            ENDIF
            CALL TurnRight(90)
        CASE "Top":
            IF currentHeading <= 90 THEN
                CALL TurnLeft(90 - currentHeading)
            ELSE IF currentHeading >= 270 THEN
                CALL TurnLeft(360 - currentHeading + 90)
            ELSE
                CALL TurnRight(currentHeading - 90)
            ENDIF
            IF distanceToTop > SafeDistance THEN
                CALL Forward(distanceToTop - SafeDistance)
            ENDIF
            CALL TurnRight(90)
        CASE "Bottom":
            IF currentHeading >= 270 THEN

```

```

        CALL TurnRight(currentHeading - 270)
    ELSE IF currentHeading <= 90 THEN
        CALL TurnRight(currentHeading + 90)
    ELSE
        CALL TurnLeft(270 - currentHeading)
    ENDIF
    IF distanceToBottom > SafeDistance THEN
        CALL Forward(distanceToBottom - SafeDistance)
    ENDIF
    CALL TurnRight(90)
END SWITCH
isAligning = true

FUNCTION MoveAlongWall():
    Deklarasi:
        alignHeading, distanceToTop, distanceToBottom, distanceToLeft,
distanceToRight, totalWeight, randomValue, deltaDistance, move : double

    Algoritma:
    IF (Environment.TickCount - lastScannedTime > scanTimeout) THEN
        CALL SetTurnRadarLeft(360)
    ENDIF
    alignHeading = Direction
    IF (currentWall == "Right" OR currentWall == "Left") THEN:
        distanceToTop = ArenaHeight - Y
        distanceToBottom = Y
        Pilih secara acak antara UP atau DOWN berdasarkan total
(distanceToTop + distanceToBottom)
        IF UP THEN:
            IF distanceToTop > SafeDistance THEN:
                move = random number between 0 and (distanceToTop - SafeDistance)
                IF alignHeading != 90 THEN
                    CALL Forward(-move)
                ELSE
                    CALL Forward(move)
                ENDIF
            ELSE:
                move = random number between 0 and (distanceToBottom -
SafeDistance)
                IF alignHeading == 90 THEN
                    CALL Forward(-move)
                ELSE
                    CALL Forward(move)
                ENDIF
            ENDIF
        ELSE {DOWN}:
            IF distanceToBottom > SafeDistance THEN:
                move = random number between 0 and (distanceToBottom -
SafeDistance)

```

```

        IF alignHeading != 270 THEN
            CALL Forward(-move)
        ELSE
            CALL Forward(move)
        ENDIF
    ELSE:
        move = random number between 0 and (distanceToTop - SafeDistance)
        IF alignHeading == 270 THEN
            CALL Forward(-move)
        ELSE
            CALL Forward(move)
        ENDIF
    ENDIF
ELSE IF (currentWall == "Top" OR currentWall == "Bottom") THEN:
    distanceToLeft = X
    distanceToRight = ArenaWidth - X
    Pilih secara acak antara LEFT atau RIGHT berdasarkan total
    (distanceToLeft + distanceToRight)
    IF LEFT THEN:
        IF distanceToLeft > SafeDistance THEN:
            move = random number between 0 and (distanceToLeft -
SafeDistance)
            IF alignHeading != 180 THEN
                CALL Forward(-move)
            ELSE
                CALL Forward(move)
            ENDIF
        ELSE:
            move = random number between 0 and (distanceToRight -
SafeDistance)
            IF alignHeading == 180 THEN
                CALL Forward(-move)
            ELSE
                CALL Forward(move)
            ENDIF
        ENDIF
    ELSE {RIGHT}:
        IF distanceToRight > SafeDistance THEN:
            move = random number between 0 and (distanceToRight -
SafeDistance)
            IF alignHeading != 0 THEN
                CALL Forward(-move)
            ELSE
                CALL Forward(move)
            ENDIF
        ELSE:
            move = random number between 0 and (distanceToLeft -
SafeDistance)
            IF alignHeading == 0 THEN

```

```

        CALL Forward(-move)
    ELSE
        CALL Forward(move)
    ENDIF
ENDIF
ENDIF

FUNCTION OnScannedBot(event e):
    Deklarasi:
        distance, bulletSpeed, enemyVX, enemyVY, dx, dy, a, b, c, discriminant,
        t, t1, t2, enemyXPredicted, enemyYPredicted : double

    Algoritma:
        lastScannedTime = Environment.TickCount {waktu saat ini}
        distance = DistanceTo(e.X, e.Y)
        nextX = e.X + e.Speed * cos(e.Direction * PI / 180)
        nextY = e.Y + e.Speed * sin(e.Direction * PI / 180)
        radarLockAngle = DirectionTo(nextX, nextY)
        radarTurn = CalcRadarBearing(radarLockAngle)
        predictedAngle = AngleProjection(e)
        gunTurn = CalcGunBearing(predictedAngle)
        Turn = BearingTo(e.X, e.Y)
        CALL SetTurnRadarLeft(radarTurn)
        CALL SetTurnGunLeft(gunTurn)
        IF gunTurn < 5 THEN
            CALL SetFire(firePower)
        ENDIF
        IF distance > 200 THEN:
            IF Hostile == true THEN
                isAligning = false
            ENDIF
            Hostile = false
        ELSE IF distance <= 200 THEN:
            CALL SetTurnLeft(Turn)
            CALL Forward(min(distance/2, 150))
            Hostile = true
        ENDIF
        CALL Rescan()
        CALL ClearEvents()

    FUNCTION AngleProjection(event e):
        Deklarasi:
            distance, bulletSpeed, enemyVX, enemyVY, dx, dy, a, b, c, discriminant,
            t, t1, t2, enemyXPredicted, enemyYPredicted : double

        Algoritma:
            distance = DistanceTo(e.X, e.Y)
            IF distance < 10 THEN:
                firePower = (Energy > 3) ? 3 : Energy
            
```

```

ELSE:
  IF (Energy < 10 OR distance > 200) THEN:
    firePower = 1
  ELSE IF (distance > 150) THEN:
    firePower = 2
  ELSE:
    firePower = 3
  ENDIF
ENDIF
bulletSpeed = CalcBulletSpeed(firePower)
enemyVX = e.Speed * cos(e.Direction * PI / 180)
enemyVY = e.Speed * sin(e.Direction * PI / 180)
dx = e.X - X
dy = e.Y - Y
a = enemyVX^2 + enemyVY^2 - bulletSpeed^2
b = 2 * (dx * enemyVX + dy * enemyVY)
c = dx^2 + dy^2
discriminant = b^2 - 4 * a * c
IF (a ≠ 0 AND discriminant >= 0) THEN:
  t1 = (-b + sqrt(discriminant)) / (2 * a)
  t2 = (-b - sqrt(discriminant)) / (2 * a)
  t = pilih nilai positif antara t1 dan t2, atau 0 jika tidak ada yang
positif
ELSE:
  t = distance / bulletSpeed
ENDIF
enemyXPredicted = e.X + enemyVX * t
enemyYPredicted = e.Y + enemyVY * t
RETURN DirectionTo(enemyXPredicted, enemyYPredicted)

FUNCTION OnHitWall(event e):
  Deklarasi: { *Tidak ada variabel lokal* }

  Algoritma:
  isAligning = false

FUNCTION OnHitBot(event e):
  Deklarasi:
  get_radar_angle, get_gun_angle : double
  Algoritma:
  get_radar_angle = RadarBearingTo(e.X, e.Y)
  get_gun_angle = GunBearingTo(e.X, e.Y)
  CALL SetTurnRadarLeft(get_radar_angle)
  CALL SetTurnGunLeft(get_gun_angle)
  CALL SetFire(3)
  CALL Forward(50)

END N1993R BOT

```



## 4.2.2 Penjelasan Fungsi & Procedure

### 1. Run()

Fungsi utama yang berjalan terus-menerus selama pertandingan.

- Mengatur warna bot (opsional) agar tampilan konsisten.
- Menginisialisasi variabel status seperti `isAligning` dan `lastScannedTime`.
- Memulai pemindaian penuh (`SetTurnRadarLeft(360)`).
- Di dalam loop utama, bot akan:
  - Memanggil fungsi `AlignWithWall()` jika belum dalam keadaan penyelarasan (`isAligning = false`).
  - Memanggil fungsi `MoveAlongWall()` untuk bergerak sepanjang tembok.
  - Memanggil fungsi `Go()` yang mengandung perintah tambahan melalui API Robocode

### 2. AlignWithWall()

Fungsi untuk menyelaraskan posisi bot dengan dinding terdekat.

- Mengecek apakah waktu sejak pemindaian terakhir melebihi batas (`scanTimeout`); jika ya, bot melakukan putaran radar ulang.
- Menghitung jarak ke masing-masing dinding (kiri, kanan, atas, bawah) menggunakan koordinat bot (X, Y) dan ukuran arena.
- Membuat dictionary untuk menentukan dinding mana yang paling dekat.
- Berdasarkan dinding terdekat dan arah bot saat ini (`currentHeading`), bot akan menyesuaikan arah (menggunakan `TurnRight` atau `TurnLeft`) sehingga menghadap ke dinding tersebut.
- Setelah menghadap dinding, bot maju (`Forward`) dengan jarak yang dihitung agar tidak terlalu dekat dengan dinding, kemudian berputar 90° untuk melanjutkan pergerakan sejajar dinding.
- Akhirnya, flag `isAligning` diset ke `true` untuk menandakan bahwa bot sudah selaras dengan dinding.

### 3. MoveAlongWall()

Fungsi yang mengatur pergerakan bot di sepanjang dinding.

- Sama seperti pada `AlignWithWall`, bot memastikan radar tetap aktif dengan melakukan putaran radar ulang jika diperlukan.
- Bot mengambil arah saat ini (`alignHeading`) dan kemudian, berdasarkan sisi dinding (`currentWall`) yang diikuti, bot menentukan secara acak apakah akan bergerak ke atas/bawah (untuk dinding kiri/kanan) atau ke kiri/kanan (untuk dinding atas/bawah).

- Dalam tiap kasus, dihitung jarak yang tersedia dari dinding dan kemudian dipilih jarak maju atau mundur secara acak agar pergerakan tidak selalu monoton sambil menjaga jarak aman (tidak terlalu dekat dengan dinding).

#### **4. OnScannedBot(ScannedBotEvent e)**

Fungsi yang dipanggil saat bot mendeteksi bot lawan melalui radar.

- Meng-update waktu pemindaian terakhir (lastScannedTime).
- Menghitung jarak ke bot lawan serta memprediksi posisi lawan di masa depan (menggunakan perhitungan dengan kecepatan dan arah lawan).
- Menghitung sudut untuk mengunci radar (radarTurn), mengatur sudut tembakan (gunTurn), dan menentukan arah pergerakan (Turn) berdasarkan posisi lawan.
- Jika perbedaan sudut tembakan kecil (kurang dari 5 derajat), bot akan menembak dengan kekuatan yang telah disesuaikan (firePower).
- Jika jarak ke musuh lebih besar dari 200, bot menonaktifkan status agresif (Hostile) dan memastikan untuk keluar dari mode penyelarasan.
- Jika jarak mendekati ( $\leq 200$ ), bot melakukan manuver seperti berputar (SetTurnLeft) dan maju (Forward) untuk melakukan pendekatan atau ramming, serta mengaktifkan flag Hostile.
- Terakhir, bot memanggil Rescan() dan ClearEvents() untuk memperbarui data dan membersihkan event-event yang belum ditangani.

#### **5. AngleProjection(ScannedBotEvent e)**

Fungsi untuk memprediksi sudut tembakan yang optimal terhadap bot lawan.

- Menghitung jarak ke bot lawan dan menentukan firePower berdasarkan jarak dan energi bot sendiri.
- Menghitung kecepatan peluru menggunakan fungsi CalcBulletSpeed(firePower).
- Menghitung komponen kecepatan bot lawan (enemyVX, enemyVY) berdasarkan kecepatan dan arah lawan.
- Menggunakan perhitungan persamaan kuadrat (dengan koefisien a, b, c) untuk mencari waktu (t) di mana peluru akan mengenai musuh.
- Jika solusi waktu (t) ditemukan dan positif, digunakan nilai tersebut; jika tidak, t dihitung sebagai perbandingan antara jarak dan kecepatan peluru.
- Menghitung posisi prediksi bot lawan (enemyXPredicted, enemyYPredicted) dengan menggunakan waktu t, lalu mengembalikan sudut (DirectionTo) dari posisi bot sendiri ke posisi prediksi tersebut.

#### **6. OnHitWall(HitWallEvent e)**

Fungsi yang dipanggil saat bot menabrak dinding.

- Fungsi ini hanya mengatur flag `isAligning` ke `false`, sehingga bot akan segera menyesuaikan kembali posisinya melalui fungsi `AlignWithWall()` pada iterasi berikutnya.

## 7. OnHitBot(HitBotEvent e)

Fungsi yang dipanggil saat bot menabrak bot lain.

- Menghitung sudut radar dan tembakan (`get_radar_angle`, `get_gun_angle`) berdasarkan posisi tabrakan.
- Mengatur pergerakan radar dan gun untuk langsung mengunci posisi bot lawan.
- Bot segera menembak (`SetFire(3)`) dan maju dengan perintah `Forward(50)` untuk melakukan manuver (misalnya, ramming) terhadap bot lawan.

## 4.3 Hit n Run Bot

### 4.3.1 Pseudocode

```
BEGIN HitnRunBot BOT

Deklarasi Variabel Global:
  movingForward : boolean
  firePower : double

MAIN:
  Buat instance bot HitnRunBot dan panggil method Start()

CONSTRUCTOR HitnRunBot():
  Inisialisasi parameter bot dari file konfigurasi "HitnRunBot.json"
  movingForward = false
  firePower = 3

FUNCTION Run():
  SET semua warna komponen bot (BodyColor, TurretColor, RadarColor,
  BulletColor, ScanColor, TracksColor, GunColor)
  SET RadarTurnRate = 20, GunTurnRate = 20
  CALL SetTurnGunLeft(∞)
  CALL SetTurnRadarLeft(∞)
  WHILE IsRunning DO:
    CALL TurnRadarLeft(360)
  END WHILE

FUNCTION ReverseDirection():
  IF movingForward THEN:
    CALL Back(300)
    movingForward = false
  ELSE:
```

```

    CALL Forward(300)
    movingForward = true

FUNCTION OnScannedBot(event e):
    Hitung posisi prediksi musuh (nextX, nextY)
    radarLockAngle = arah ke posisi prediksi musuh
    CALL SetTurnRadarLeft(CalcRadarBearing(radarLockAngle))
    gunTurn = CalcGunBearing(AngleProjection(e))
    CALL SetTurnGunLeft(gunTurn)
    IF gunTurn < 10 THEN:
        CALL SetFire(firePower)
    ENDIF
    CALL SetTurnLeft(CalcBearing(radarLockAngle))
    CALL SetForward(min(jarak ke musuh/5, 50))
    CALL Rescan()

FUNCTION AngleProjection(event e):
    Hitung firePower berdasarkan energi dan jarak musuh
    Prediksi posisi musuh (enemyXPredicted, enemyYPredicted) dengan hitungan
    fisika
    RETURN arah tembakan ke posisi prediksi musuh

FUNCTION OnHitBot(event e):
    CALL SetTurnRadarLeft ke arah musuh
    CALL SetTurnGunLeft ke arah musuh
    CALL SetFire(min(Energy, 3))
    CALL SetTurnLeft ke arah musuh
    CALL SetForward(10)

FUNCTION OnHitWall(event e):
    CALL ReverseDirection()

END HitnRunBot BOT

```

#### 4.3.2 Penjelasan Fungsi & Procedure

##### 1. Run()

- Fungsi utama yang berjalan terus selama pertandingan berlangsung.
- Mengatur warna bot dan laju rotasi radar serta senjata.
- Radar dan senjata terus menerus diputar untuk deteksi cepat dan aksi tembakan yang efektif.
- Melakukan pemindaian radar penuh secara kontinu (TurnRadarLeft(360)).

##### 2. ReverseDirection()

- Fungsi ini dipanggil ketika bot menabrak dinding (dari fungsi OnHitWall).
- Menggerakkan bot ke arah sebaliknya dengan jarak jauh (300 unit) agar bot segera keluar dari posisi yang menyebabkan tabrakan.

- Mengubah status arah gerak (movingForward) untuk menghindari tabrakan berulang di titik yang sama.

### 3. OnScannedBot(ScannedBotEvent e)

- Fungsi ini aktif saat radar mendeteksi posisi musuh.
- Menghitung prediksi posisi musuh di masa depan berdasarkan arah dan kecepatannya.
- Mengunci radar dan mengarahkan senjata ke arah posisi prediksi tersebut.
- Jika senjata sudah berada pada posisi akurat (kurang dari  $10^\circ$ ), bot langsung melakukan tembakan.
- Melakukan pergerakan pendek ke arah musuh untuk manuver menghindar sekaligus melakukan pendekatan yang aman.

### 4. AngleProjection(ScannedBotEvent e)

- Fungsi prediksi sudut tembakan optimal yang dihitung berdasarkan jarak, energi yang tersedia, dan prediksi posisi musuh.
- Menghitung kekuatan tembakan (firePower) berdasarkan kondisi energi dan jarak.
- Mengembalikan sudut tembakan optimal yang akan mengenai musuh berdasarkan prediksi gerakannya.

### 5. OnHitBot(HitBotEvent e)

- Fungsi yang dipanggil ketika bot bertabrakan dengan bot musuh.
- Langsung mengarahkan radar dan senjata ke posisi bot lawan.
- Menembak dengan kekuatan maksimum yang tersedia (hingga 3 energi).
- Melakukan manuver pendek ke arah depan (Forward(10)) untuk segera menjauh atau mengejar bot lawan.

### 6. OnHitWall(HitWallEvent e)

- Fungsi dipanggil ketika bot menabrak dinding.
- Memanggil fungsi ReverseDirection() untuk langsung mengubah arah gerak dan menjauh dari dinding.

## 4.4 Forza Ferrari

### 4.4.1 Pseudocode

```
BEGIN Forza_Ferrari BOT

Deklarasi Variabel Global:
    movingForward : boolean
    firePower : double
    enemies : Dictionary<int, Enemy>
    target : Enemy
    mode : enum {Scanning, Evaluating, Targeting}

MAIN:
```

Buat instance bot Forza\_Ferrari dan panggil method Start()

```
CONSTRUCTOR Forza_Ferrari():  
  Inisialisasi parameter bot dari file konfigurasi "Forza_Ferrari.json"  
  movingForward = false  
  firePower = 3  
  enemies = dictionary kosong  
  target = null  
  mode = Scanning
```

```
FUNCTION Run():  
  SET semua warna komponen bot (BodyColor, TurretColor, RadarColor,  
  BulletColor, ScanColor, TracksColor, GunColor)  
  WHILE IsRunning DO:  
    SWITCH mode:  
      CASE Scanning:  
        CALL SetTurnRadarLeft(360)  
        CALL WaitFor(RadarCompleteCondition)  
        mode = Evaluating  
      CASE Evaluating:  
        CALL Evaluate()  
        IF target ≠ null THEN:  
          mode = Targeting  
      CASE Targeting:  
        CALL SetTurnRadarLeft(360)  
    END SWITCH  
    CALL WaitFor(TurnCompleteCondition)  
  END WHILE
```

```
FUNCTION Evaluate():  
  bestCandidate : Enemy = null  
  maxAvgDistance : double = -∞  
  FOR EACH candidate IN enemies DO:  
    IF candidate aktif THEN:  
      hitung rata-rata jarak (avgDistance) terhadap semua musuh lain  
      IF avgDistance > maxAvgDistance THEN:  
        maxAvgDistance = avgDistance  
        bestCandidate = candidate  
    END FOR  
  target = bestCandidate
```

```
FUNCTION OnScannedBot(event e):  
  IF mode == Scanning THEN:  
    IF enemies tidak mengandung ID bot musuh THEN:  
      Tambahkan bot musuh baru ke enemies  
    ELSE:  
      Perbarui data bot musuh yang sudah ada  
  ELSE IF mode == Targeting THEN:  
    IF e.ScannedBotId == target.id THEN:
```

```

        Hitung posisi prediksi musuh (nextX, nextY)
        radarLockAngle = arah ke posisi prediksi musuh
        CALL SetTurnRadarLeft(CalcRadarBearing(radarLockAngle))
        gunTurn = CalcGunBearing(AngleProjection(e))
        CALL SetTurnGunLeft(gunTurn)
        IF gunTurn < 10 THEN:
            CALL SetFire(firePower)
        ENDIF
        CALL SetTurnLeft(CalcBearing(radarLockAngle))
        CALL SetForward(min(jarak ke musuh/4, 30))
        CALL Rescan()
    END IF

FUNCTION AngleProjection(event e):
    Hitung firePower berdasarkan energi dan jarak musuh
    Prediksi posisi musuh (enemyXPredicted, enemyYPredicted) dengan hitungan
    fisika
    RETURN arah tembakan ke posisi prediksi musuh

FUNCTION OnHitBot(event e):
    CALL SetTurnRadarLeft ke arah musuh
    CALL SetTurnGunLeft ke arah musuh
    CALL SetFire(min(Energy, 3))
    CALL SetTurnLeft ke arah musuh
    CALL SetForward(10)

FUNCTION OnHitWall(event e):
    IF movingForward THEN:
        CALL Back(300)
        movingForward = false
    ELSE:
        CALL Forward(300)
        movingForward = true

FUNCTION OnBotDeath(event e):
    Tandai musuh sebagai tidak aktif
    IF musuh yang mati adalah target THEN:
        target = null
        mode = Scanning

END Forza_Ferrari BOT

```

#### 4.4.2 Penjelasan Fungsi & Procedure

##### 1. Run()

- Fungsi utama yang terus berjalan selama pertandingan.

- Mengatur warna komponen bot untuk identitas visual yang jelas.
- Menginisialisasi dictionary musuh (enemies) dan mode awal bot ke Scanning.
- Dalam loop utama bot:
  - Mode Scanning: melakukan pemindaian radar penuh ( $360^\circ$ ), lalu berpindah ke mode Evaluating.
  - Mode Evaluating: memanggil fungsi Evaluate() untuk memilih target terbaik (bot dengan jarak rata-rata terjauh dari musuh lain). Jika target ditemukan, bot berpindah ke mode Targeting.
  - Mode Targeting: kembali melakukan pemindaian penuh radar secara kontinu untuk terus memperbarui informasi target.

## **2. Evaluate()**

- Fungsi ini mengevaluasi musuh-musuh yang telah terdeteksi.
- Menghitung jarak rata-rata tiap musuh terhadap bot-bot musuh lain.
- Memilih musuh yang paling terisolasi (jarak rata-rata terjauh dengan musuh lain) sebagai target utama.
- Mengupdate variabel target dengan kandidat terbaik yang ditemukan.

## **3. OnScannedBot(ScannedBotEvent e)**

- Fungsi yang aktif ketika radar mendeteksi bot lawan.
- Pada mode Scanning, fungsi ini menambahkan atau memperbarui informasi bot lawan ke dictionary enemies.
- Pada mode Targeting, jika bot yang dipindai adalah target:
  - Memperkirakan posisi musuh pada waktu mendatang menggunakan prediksi kecepatan dan arah musuh.
  - Mengatur arah radar (radar lock), arah tembakan (gun turn), serta arah pergerakan bot (turn).
  - Melakukan tembakan jika posisi senjata sudah dalam sudut yang tepat (kurang dari  $10^\circ$ ).
  - Bergerak maju secara terkontrol ke arah musuh dan melakukan rescanning.

## **4. AngleProjection(ScannedBotEvent e)**

- Fungsi untuk menghitung sudut tembakan optimal berdasarkan prediksi posisi bot musuh.
- Menyesuaikan kekuatan tembakan (firePower) berdasarkan energi yang tersedia dan jarak ke musuh.



- Menghitung kecepatan peluru dan pergerakan musuh untuk menentukan posisi prediksi secara akurat.
- Mengembalikan sudut ideal untuk menembak ke arah posisi prediksi musuh.

#### **5. OnHitBot(HitBotEvent e)**

- Dipanggil ketika bot bertabrakan dengan bot lawan.
- Langsung mengarahkan radar dan senjata ke posisi bot lawan tersebut.
- Melakukan tembakan dengan kekuatan maksimum yang tersedia (hingga 3 poin energi).
- Melakukan gerakan maju pendek (Forward(10)) untuk manuver tambahan.

#### **6. OnHitWall(HitWallEvent e)**

- Fungsi yang dipanggil saat bot bertabrakan dengan dinding arena.
- Memerintahkan bot untuk mundur (Back) atau maju (Forward) sejauh 300 unit tergantung pada arah gerak sebelumnya.
- Mengubah arah gerak (toggle variabel movingForward) untuk mencegah tabrakan berulang.

#### **7. OnBotDeath(BotDeathEvent e)**

- Fungsi ini dipanggil saat ada bot lawan yang mati.
- Memperbarui status aktif bot tersebut menjadi nonaktif di dictionary.
- Jika bot yang mati merupakan target utama, maka bot mengatur ulang target ke null dan kembali ke mode Scanning.

#### **8. Class Enemy**

Kelas ini digunakan untuk menyimpan informasi detail mengenai bot musuh yang telah dipindai radar. Data ini digunakan untuk pemilihan target berdasarkan jarak rata-rata ke bot musuh lainnya.

##### **Atribut:**

- active : boolean  
Status apakah bot musuh masih hidup (aktif) atau sudah mati (nonaktif).
- id : integer  
Identifier unik dari bot musuh tersebut.
- ex, ey : double  
Posisi koordinat (X, Y) terakhir bot musuh yang terdeteksi.
- energy : double  
Energi tersisa pada bot musuh.

- speed : double  
Kecepatan terakhir dari bot musuh.
- direction : double  
Arah terakhir dari bot musuh.

#### Konstruktor:

- Enemy(int id, double ex, double ey, double energy, double speed, double direction)  
Menginisialisasi semua atribut bot musuh dengan data terbaru ketika pertama kali terdeteksi.

#### Method:

- Update(double ex, double ey, double energy, double speed, double direction)  
Memperbarui data posisi, energi, kecepatan, dan arah dari bot musuh setiap kali radar kembali mendeteksi bot yang sama.

## 4.5 Pengujian

Pengujian yang dilakukan berupa 3 *game* permainan, dengan satu permainan terdiri dari beberapa ronde. Keempat bot yang telah dibuat diuji dengan sistem pertandingan 1v1v1v1, seperti tertera pada tabel berikut:

No

Screenshot Hasil Pengujian

1.

Results for 10 rounds

Rank	Name	Total Score	Survival	Surv. Bonus	Bullet D...	Bullet Bo...	Ram Dmg.	Ram Bonus	1sts	2nds	3rds
1	Trishula 1.0	2402	1000	150	1072	117	64	0	4	3	2
2	N1993R 1.0	1850	800	60	692	81	209	8	0	4	4
3	HitnRunBot 1.0	1752	450	0	904	74	307	17	3	1	2
4	Forza Ferrari 1.0	1558	400	30	828	51	218	30	2	1	1

2.

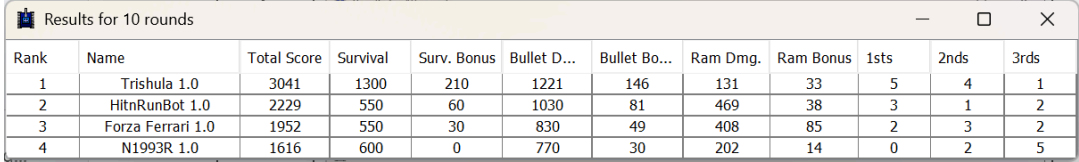
Results for 10 rounds

Rank	Name	Total Score	Survival	Surv. Bonus	Bullet D...	Bullet Bo...	Ram Dmg.	Ram Bonus	1sts	2nds	3rds
1	N1993R 1.0	2430	950	90	906	91	366	26	3	4	2
2	Forza Ferrari 1.0	2403	550	30	1068	41	553	160	3	2	2
3	Trishula 1.0	2304	1000	150	945	59	116	34	2	2	4
4	HitnRunBot 1.0	1775	350	30	756	40	559	39	2	2	2

3.

Results for 10 rounds

Rank	Name	Total Score	Survival	Surv. Bonus	Bullet D...	Bullet Bo...	Ram Dmg.	Ram Bonus	1sts	2nds	3rds
1	Trishula 1.0	2471	1100	120	1012	130	108	0	3	5	1
2	HitnRunBot 1.0	2416	600	90	988	103	533	102	4	0	4
3	N1993R 1.0	1828	750	60	798	24	196	0	2	2	3
4	Forza Ferrari 1.0	1689	350	0	734	21	492	91	1	3	2

4.											
	<p style="text-align: center;"><b>Akumulasi Poin</b></p> <p style="text-align: center;"><b>Trishula: 10218</b></p> <p style="text-align: center;"><b>HitnRun: 8172</b></p> <p style="text-align: center;"><b>N1993R: 7724</b></p> <p style="text-align: center;"><b>Forza Ferrari: 7602</b></p>										

**Tabel 4.5 Hasil Pengujian Bot**

Berdasarkan hasil pengujian sebanyak empat kali, bot Trishula berhasil meraih peringkat satu sebanyak tiga kali dan selalu mencatat nilai survival tertinggi pada setiap pertandingan. Hal ini disebabkan oleh filosofi bot Trishula yang selalu mencari titik dengan jarak rata-rata terjauh dari semua bot, sehingga memungkinkan bot ini bertahan hidup lebih lama. Dengan waktu bertahan yang lebih lama, bot Trishula juga memiliki peluang untuk menghasilkan damage yang lebih besar dibandingkan bot lain. Hampir semua kondisi, baik pada spawn point maupun saat in-match, menghasilkan nilai optimal bagi bot Trishula. Namun, kondisi seperti ketika tiga bot lainnya sudah saling membunuh terlebih dahulu dapat membuat bot Trishula kalah dalam hal damage poin meskipun memiliki nilai survival yang tinggi. Selain itu, kondisi di mana bot Trishula spawn di sudut dengan tiga bot rammer mengelilinginya juga dapat mengakibatkan bot ini tidak dapat melarikan diri untuk bertahan hidup.

Sementara itu, bot N1993R hanya berhasil meraih peringkat pertama sekali, namun konsisten memperoleh nilai Survival Score tertinggi kedua pada setiap pertandingan. Strategi bot N1993R berfokus pada pencarian tembok terdekat untuk bergerak secara osilasi, dan berubah ke mode ram saat mendeteksi adanya bot yang mendekat. Pendekatan ini optimal apabila bot N1993R muncul di lokasi yang jauh dari musuh, sehingga memungkinkan freehit atau pertarungan 1v1 ketika hanya ada satu bot lawan yang mendekat. Namun, bila bot N1993R muncul di area yang ramai, strategi tersebut bisa terganggu karena bot ini akan langsung mengunci salah satu target, sehingga bot lawan lainnya mendapatkan kesempatan melakukan freehit. Selain itu, pada situasi akhir pertandingan ketika hanya tersisa satu bot yang menjaga jarak, perolehan damage poin pun cenderung tidak maksimal, terutama jika gerakan bot lawan bersifat acak.

Bot Forza Ferrari dan HitNRunBot tidak berhasil meraih peringkat pertama pada pertandingan manapun, meskipun keduanya sempat menduduki peringkat kedua pada salah satu pertandingan. Keduanya unggul dalam perolehan damage point yang tinggi dari ram damage beserta bonusnya. Strategi optimal bagi kedua bot ini terjadi ketika posisi spawn bot tersebar secara merata, sehingga memungkinkan terjadinya pertarungan 1v1 tanpa gangguan dari bot lain. Dengan demikian, bot dapat bertahan lebih lama dan menghasilkan poin lebih tinggi. Namun, apabila spawn point menunjukkan konsentrasi bot yang tinggi, kemungkinan kedua bot ini akan tereliminasi lebih cepat karena situasi lock target yang berubah menjadi 2v1, sehingga perolehan poin pun tidak optimal.

Secara keseluruhan, keempat bot telah menerapkan strategi greedy dengan optimal. Namun, kondisi lingkungan seperti spawn point, strategi bot lawan, bot lawan yang tersisa, dan faktor-faktor lain dapat menambah atau mengurangi efektivitas strategi masing-masing bot.

## BAB 5 KESIMPULAN

Setelah melakukan pengujian menyeluruh terhadap keempat bot yang telah dibuat, dapat disimpulkan bahwa masing-masing bot berhasil menerapkan algoritma greedy dengan variasi strategi yang unik. Keempat strategi greedy tersebut, yaitu **Least Risky Target**, **Furthest Point**, **Ramming (Most Damage Per Second)**, dan **Switch Wall Navigation (Safest Movement Pattern)**, telah mampu menunjukkan perilaku dan performa yang sesuai dengan desain heuristik masing-masing.

Hasil pengujian secara nyata membuktikan bahwa implementasi algoritma greedy pada bot dapat secara efektif menghasilkan nilai yang optimal berdasarkan skenario pertandingan tertentu. Namun, perlu dicatat bahwa pencapaian nilai optimal dengan algoritma greedy tidak selalu menjadi satu-satunya faktor penentu kemenangan dalam Robocode Tank Royale. Faktor lain seperti ketepatan prediksi gerakan lawan, kemampuan adaptasi terhadap situasi dinamis, efisiensi penggunaan energi, dan manuver taktis juga memainkan peran penting dalam menentukan hasil akhir sebuah pertandingan.

Dari pengamatan selama pengujian, diidentifikasi bahwa tidak ada satu strategi greedy tunggal yang sempurna di setiap situasi; masing-masing strategi memiliki keunggulan dan kelemahannya sendiri tergantung pada gaya bermain musuh dan setup pertandingan. Oleh karena itu, penggunaan strategi greedy yang optimal harus disesuaikan dengan konteks dan tujuan pertandingan yang spesifik. Keberuntungan mendapatkan posisi pada *spawn point* juga menjadi faktor dalam perolehan kemenangan maupun kekalahan pada permainan ini.

Implementasi algoritma greedy dalam berbagai variasi strategi telah berhasil dikembangkan dan diuji dengan baik. Keempat bot ini mampu mencapai tujuan heuristiknya masing-masing secara efektif, dan menyediakan fondasi kuat untuk eksplorasi lebih lanjut dalam mengembangkan strategi yang lebih kompleks dan adaptif.

## LAMPIRAN

[Repository GitHub](#)

[Link Video YouTube](#)

## DAFTAR PUSTAKA

1. "Understanding HawkOnFire." *Robowiki*, [https://robowiki.net/wiki/HawkOnFire/Understanding\\_HawkOnFire](https://robowiki.net/wiki/HawkOnFire/Understanding_HawkOnFire). Accessed 24 Mar. 2025.
2. "Understanding Coriantumr." *Robowiki*, [https://robowiki.net/wiki/Coriantumr/Understanding\\_Coriantumr](https://robowiki.net/wiki/Coriantumr/Understanding_Coriantumr). Accessed 24 Mar. 2025.
3. "Minimum Risk Movement." *Robowiki*, [https://robowiki.net/wiki/Minimum\\_Risk\\_Movement](https://robowiki.net/wiki/Minimum_Risk_Movement). Accessed 24 Mar. 2025.
4. "Melee Strategy." *Robowiki*, [https://robowiki.net/wiki/Melee\\_Strategy](https://robowiki.net/wiki/Melee_Strategy). Accessed 24 Mar. 2025.

5. "Movement." *Robowiki*, <https://robowiki.net/wiki/Movement>. Accessed 24 Mar. 2025.
6. "Rolling Averages." *Robowiki*, [https://robowiki.net/wiki/Rolling\\_Averages](https://robowiki.net/wiki/Rolling_Averages). Accessed 24 Mar. 2025.
7. "Circular Targeting." *Robowiki*, [https://robowiki.net/wiki/Circular\\_Targeting](https://robowiki.net/wiki/Circular_Targeting). Accessed 24 Mar. 2025.
8. "Linear Targeting." *Robowiki*, [https://robowiki.net/wiki/Linear\\_Targeting](https://robowiki.net/wiki/Linear_Targeting). Accessed 24 Mar. 2025.
9. "GoTo." *Robowiki*, <https://robowiki.net/wiki/GoTo>. Accessed 24 Mar. 2025.
10. "Robocode Tank Royale Docs." *Robocode*, <https://robocode-dev.github.io/tank-royale/>. Accessed 22 Mar. 2025
11. "Robocode.TankRoyale.BotApi" Nuget, <https://www.nuget.org/packages/Robocode.TankRoyale.BotApi>. Accessed 23 Mar. 2025