# Laporan Tugas Besar 1 IF 2211
# Strategi Algoritma
# Penerapan Algoritma Greedy dalam Robocode Tank Royale

**Oleh:**
**Kelompok D1753ALG, yang terdiri atas:**
**Kenneth Poenadi - 13523040**
**Bob Kunanda - 13523086**
**Muhammad Zahran Ramadhan Ardiana - 13523104**

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

**2025**

# Daftar Isi

**BAB 1** **5**
**Deskripsi Tugas** **5**
Gambar 1. Visualisasi Robocode Tank Royale 5
**BAB 2** **12**
**Landasan Teori** **12**
2.1 Dasar Teori Algoritma Greedy 12
2.2 Cara Kerja Program 15
1. Cara Kerja Program Robocode 15
a. Struktur Permainan 15
2. Method Method yang ada pada Robocode Tank Royale 16
3. Implementasi Algoritma Greedy pada Bot Robocode 19
a. Elemen Algoritma Greedy dalam Robocode 19
b. Strategi Greedy yang Dapat Diterapkan 19
4. Cara Menjalankan Bot Robocode 20
2.3 Struktur Kerja Program pada Robocode Tank Royale 21
**BAB 3** **22**
**Aplikasi Strategi Greedy pada Bot** **22**
3.1 Mapping Persoalan Greedy yang ada pada Robocode Tank Royale 22
3.2 Bot B3040PKK 23
3.2.1 Penjelasan Robot B3040PKK 23
3.2.2 Elemen - Elemen Greedy 23
3.2.3 Pencarian Greedy yang tepat 26
Gambar 2. Himpunan Kandidat dalam Robocode Tank Royale 26
Gambar 3. Locking kepada musuh yang discan pertama kali 27
Gambar 4. Bot diharapkan dapat melakukan locking pertama kali mendapatkan scan musuh 28
Gambar 5. Bot diharapkan dapat menghadap musuh dan lock musuh dan ram hingga titik darah penghabisan 29
Gambar 6. Bila musuh sudah yang awalnya kita lock sudah tidak ditemukan, putar terus radar hingga menemukan musuh baru dan lakukan locking ke musuh tersebut 29
3.2.4 Efektivitas dan Efisiensi 29
3.3 Bot D1753ALG 30
3.3.1 Penjelasan Robot D1753ALG 30
3.3.2 Elemen - Elemen Greedy 30
Gambar 7. Contoh Bagaimana Bot Mengkalkulasi Lokasi yang Dituju Musuh 31
Gambar 8. Formulasi Rumus Penembakkan Bot 32
3.3.3 Pencarian Greedy yang tepat 34
Gambar 9. Pencarian Musuh dengan Cara Mengscan Sekitar 34

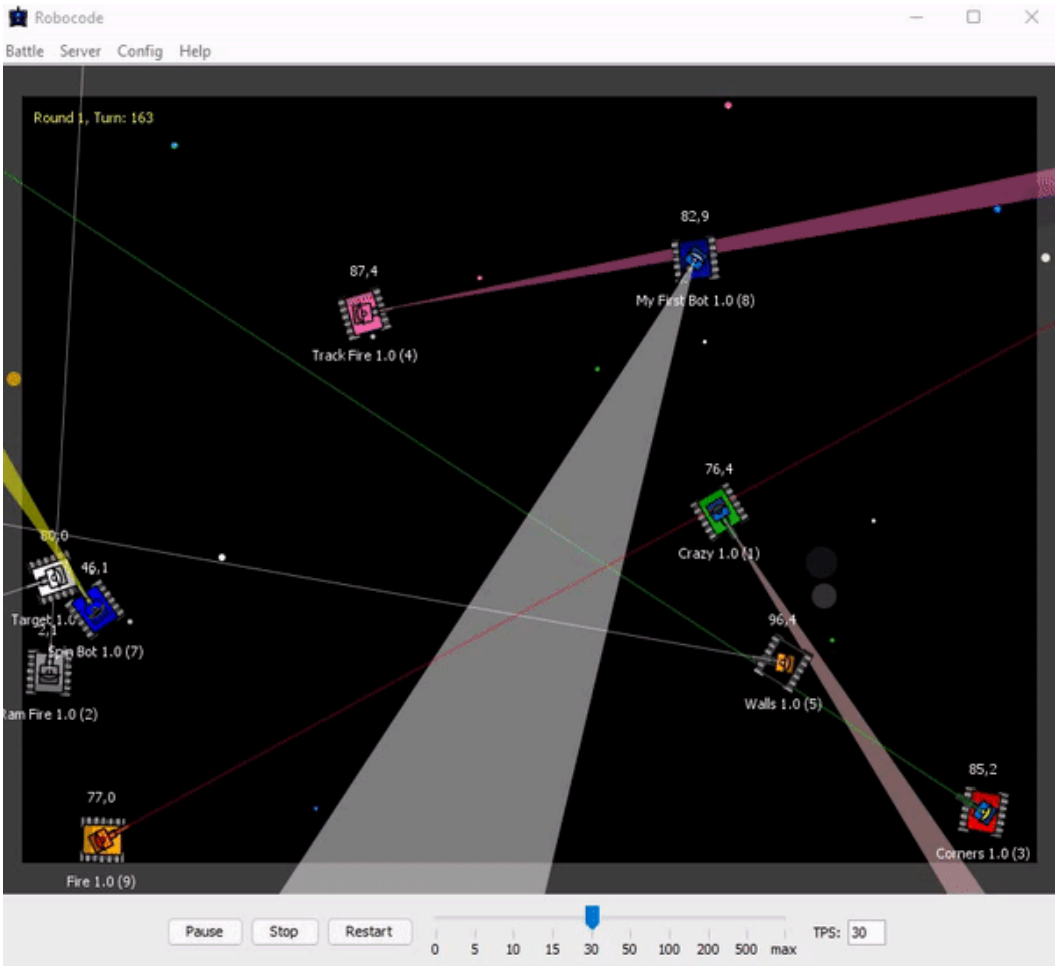
Gambar 10. Mendekati MusuhGambar 11. Bergerak untuk Menghindar dari Musuh35
3.3.4 Efektivitas dan Efisiensi 36
3.4 Bot F3812FIO 37
3.4.1 Penjelasan Robot F3812FIO 37
3.4.2 Elemen - Elemen Greedy 37
3.4.3 Efektivitas dan Efisiensi 38
3.5 Bot Jeb_ 38
3.5.1 Penjelasan Jeb_ 38
3.5.2 Elemen - Elemen Greedy 39
3.5.3 Efektivitas dan Efisiensi 41
**BAB 4 43**
**Implementasi & Pengujian 43**
4.1 Bot 3040 PKK 43
4.1.1 Implementasi 43
4.1.2 Penjelasan Struktur Data, Cara Kerja Robot, Fungsi, dan Prosedur 47
Gambar 11. Visualisasi Kelas Inheritance dari Robocode Tank Royale 48
Gambar 12. Ukuran Bot dalam Unit di Robocode Tank Royale 50
Gambar 13. Visualisasi Angle To Enemy 51
Gambar 14. Visualisasi Radar yang udah menemukan musuh 52
Gambar 15. Visualisasi Extra Turn Target kepada musuh 53
Gambar 16. Visualisasi Hadap Target kepada musuh 54
4.2 Bot D1753ALG 56
4.2.1 Implementasi 56
4.2.2 Penjelasan Struktur Data, Cara Kerja Robot, Fungsi, dan Prosedur 59
Gambar 17. Visualisasi Kelas Inheritance dari Robocode Tank Royale 62
4.3 Bot F3812FIO 63
Gambar 18. Visualisasi Kelas Inheritance dari Robocode Tank Royale 63
4.3.1 Implementasi 64
4.3.2 Penjelasan Struktur Data, Cara Kerja Robot, Fungsi, dan Prosedur 69
4.4 Jeb_ 71
4.4.1 Implementasi 71
4.4.2 Penjelasan Struktur Data, Cara Kerja Robot, Fungsi, dan Prosedur 73
Gambar 19. Visualisasi Kelas Inheritance dari Robocode Tank Royale 74
4.5 Pengujian 78
Round 1: Tubes1_STIMA 78
Gambar 20. Round 1 Pertandingan Robot - Robot Kelompok D1753ALG 78
Round 2: Tubes1_STIMA 78
Gambar 21. Round 2 Pertandingan Robot - Robot Kelompok D1753ALG 79
Round 3: Tubes1_STIMA 79
Gambar 22. Round 3 Pertandingan Robot - Robot Kelompok D1753ALG 79
**BAB 5 80**

**KESIMPULAN DAN SARAN** **80**
5.1 Kesimpulan 80
5.2 Saran 80
**LAMPIRAN** **81**
Github: https://github.com/BobSwagg13/Tubes1_D1753ALG 81
Youtube Video: Tugas Besar 1 STIMA 2025, Kelompok D1753ALG 81

# BAB 1

# Deskripsi Tugas

Pemanfaatan Algoritma Greedy dalam Tugas Besar 1 Strategi Algoritma




*Gambar 1. Visualisasi Robocode Tank Royale*

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

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

Komponen-komponen dari permainan ini antara lain:

1. Rounds dan Turns

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

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

   Pada setiap turn, sebuah bot dapat:

   - Menggerakkan bot, memindai musuh, dan menembakkan senjata.
   - Bereaksi terhadap peristiwa seperti saat bot terkena peluru atau bertabrakan dengan bot lain atau dinding.

- Perintah untuk bergerak, berputar, memindai, menembak, dan sebagainya dikirim ke server untuk setiap turn.

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

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

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

2. Batas Waktu Giliran

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

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

3. Energi

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

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

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

4. Peluru

   Semakin banyak energi (daya tembak) yang digunakan untuk menembakkan peluru, semakin berat peluru tersebut dan semakin lambat gerakannya. Namun, peluru yang lebih berat juga menghasilkan lebih banyak kerusakan dan memungkinkan bot mendapatkan lebih banyak energi saat mengenai bot musuh.

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

5. Panas Meriam (Gun Heat)

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

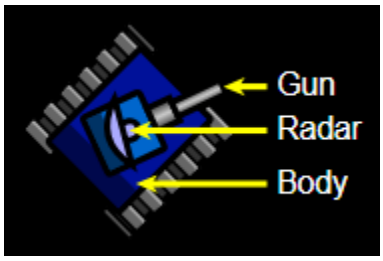
6. Tabrakan

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

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

7. Bagian Tubuh Tank

   Tubuh tank terdiri dari 3 bagian:




   *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 lainya yang masih bertahan pada ronde tersebut mendapatkan **50 poin.**
- **Last Survival Bonus:** Bot terakhir yang bertahan pada suatu ronde akan mendapatkan **10 poin** dikali dengan banyaknya musuh.
- **Ram Damage:** Bot mendapatkan **poin sebesar 2 kalinya *damage*** yang dibuat kepada bot musuh dengan cara menabrak.
- **Ram Damage Bonus:** Apabila musuh terbunuh dengan cara ditabrak, bot mendapatkan **poin sebesar 30% dari *damage*** yang dibuat kepada musuh yang terbunuh.

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

# BAB 2

# Landasan Teori

## 2.1 Dasar Teori Algoritma Greedy

Algoritma greedy merupakan metode yang paling populer dan sederhana untuk memecahkan persoalan optimasi (persoalan mencari solusi optimal). Hanya terdapat dua macam persoalan optimasi: 1. Maksimasi (maximization) 2. Minimasi (minimization). Algoritma greedy adalah algoritma yang memecahkan persoalan secara **langkah per langkah (step by step)** sedemikian sehingga, pada setiap langkah: 1. mengambil pilihan yang terbaik yang dapat diperoleh pada saat itu tanpa memperhatikan konsekuensi ke depan (prinsip **"take what you can get now!**") 2. dan "berharap" bahwa dengan memilih optimum lokal pada setiap langkah akan berakhir dengan **optimum global**.

Terdapat beberapa elemen - elemen yang ditinjau untuk melakukan algoritma greedy. Berikut 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 di optimisasi oleh fungsi objektif.

Contoh Persoalan yang diselesaikan dalam Algoritma Greedy:

**Contoh 1 (Greedy Optimum)**: tersedia banyak koin 1, 5, 10, 25

• Uang senilai A = 32 dapat ditukar dengan banyak cara berikut:

- 32 = 1 + 1 + … + 1 (32 koin)
- 32 = 5 + 5 + 5 + 5 + 10 + 1 + 1 (7 koin)
- 32 = 10 + 10 + 10 + 1 + 1 (5 koin)
- … dst

• Solusi minimum: 32 = 25 + 5 + 1 + 1 (4 koin)

Persoalan diatas diselesaikan bukan dengan Algoritma Greedy melainkan dengan Algoritma Brute Force yang mencoba solusi satu - satu sampai ketemu hasil yang optimal.

Pada persoalan penukaran uang:

1. Himpunan kandidat: himpunan koin atau pecahan uang yang merepresentasikan nilai 1, 5, 10, 25, paling sedikit mengandung satu koin untuk setiap nilai.
2. Himpunan solusi: koin-koin yang terpilih.

3. Fungsi solusi: memeriksa apakah total nilai koin yang dipilih tepat sama jumlahnya dengan nilai uang yang ditukarkan.
4. Fungsi seleksi: pilihlah koin yang bernilai tertinggi dari himpunan koin yang tersisa.
5. Fungsi kelayakan: memeriksa apakah koin yang baru dipilih apabila dijumlahkan dengan semua koin yang sudah berada di dalam himpunan tidak melebihi jumlah uang yang harus dibayar.
6. Fungsi obyektif: jumlah koin yang digunakan minimum.

Namun perlu diperhatikan bahwa optimum global belum tentu merupakan solusi optimum (terbaik), bisa jadi merupakan solusi sub-optimum atau pseudo-optimum.

Alasan:

1. Algoritma greedy tidak beroperasi secara menyeluruh terhadap semua kemungkinan solusi yang ada (sebagaimana pada metode exhaustive search).
2. Terdapat beberapa fungsi SELEKSI yang berbeda, sehingga kita harus memilih fungsi yang tepat jika kita ingin algoritma menghasilkan solusi optiamal.

Jadi, pada sebagian persoalan, algoritma greedy tidak selalu berhasil memberikan solusi yang optimal, namun sub-optimal.

**Contoh 2 (Greedy Non Optimum):** Tinjau persoalan penukaran uang berikut:

- Koin: 5, 4, 3, dan 1 (dalam jumlah yang banyak)
  Uang yang ditukar A = 7.
  Solusi greedy: 7 = 5 + 1 + 1 ( 3 koin) → tidak optimal
  Solusi optimal: 7 = 4 + 3 ( 2 koin)
- Koin: 10, 7, 1 (dalam jumlah yang banyak)
  Uang yang ditukar: 15
  Solusi greedy: 15 = 10 + 1 + 1 + 1 + 1 + 1 (6 koin) → tidak optimal
  Solusi optimal: 15 = 7 + 7 + 1 (hanya 3 koin)
- Koin: 15, 10, dan 1 (dalam jumlah yang banyak)
  Uang yang ditukar: 20

Solusi greedy: 20 = 15 + 1 + 1 + 1 + 1 + 1  (6 koin) → tidak optimal
Solusi optimal: 20 = 10 + 10  (2 koin)

- Jika solusi terbaik mutlak tidak terlalu diperlukan, maka algoritma greedy dapat digunakan untuk menghasilkan solusi hampiran (approximation),
- daripada menggunakan algoritma yang kebutuhan waktunya eksponensial untuk menghasilkan solusi yang eksak.
- Misalnya mencari tur dengan bobot minimal pada persoalan TSP untuk jumlah simpul (n) yang banyak dengan algoritma brute force dibutuhkan waktu komputasi yang lama untuk menemukannya.
- Dengan algoritma greedy, meskipun tur dengan berbobot minimal tidak dapat ditemukan, namun solusi dengan algoritma greedy dianggap sebagai hampiran solusi optimal.

Namun bila algoritma greedy dapat menghasilkan solusi optimal, maka keoptimalannya itu harus dapat dibuktikan secara matematis. Membuktikan optimalitas algoritma greedy secara matematis adalah tantangan tersendiri.  Lebih mudah memperlihatkan algoritma greedy tidak selalu optimal dengan menunjukkan counterexample (contoh kasus yang menunjukkan solusi yang diperoleh tidak optimal), misalnya pada persoalan penukaran uang di atas (kadang-kadang solusinya optimal kadang-kadang tidak).

## 2.2 Cara Kerja Program

**Cara Kerja Program Robocode dan Implementasi Algoritma Greedy pada Bot**

**1. Cara Kerja Program Robocode**

Robocode adalah permainan pemrograman di mana pemain membuat bot dalam bentuk tank virtual yang bertarung di arena. Pertempuran berlangsung dalam bentuk Battle Royale, di mana bot akan bertarung hingga hanya tersisa satu pemenang.

**a. Struktur Permainan**

- **Rounds dan Turns**: Pertempuran terdiri dari beberapa ronde yang terdiri dari beberapa giliran.
- **Batas Waktu Giliran**: Setiap giliran memiliki batas waktu eksekusi antara 30-50 ms.
- **Energi**: Bot memiliki energi awal 100 poin yang akan berkurang jika terkena tembakan atau bertabrakan dengan bot lain.
- **Peluru**: Peluru yang lebih berat bergerak lebih lambat tetapi menghasilkan lebih banyak kerusakan.
- **Panas Meriam**: Bot tidak dapat menembak jika meriamnya terlalu panas.
- **Tabrakan**: Bot yang menabrak dinding atau bot lain akan menerima kerusakan.
- **Bagian Tubuh Tank**: Terdiri dari body (gerakan utama), gun (senjata), dan radar (sensor untuk mendeteksi musuh).
- **Pergerakan dan Berbelok**: Bot dapat bergerak maju dan mundur serta berbelok dengan kecepatan yang bergantung pada kecepatannya saat itu.
- **Pemindaian**: Radar bot dapat mendeteksi musuh dalam jangkauan 1200 piksel.
- **Skor**: Bot mendapatkan poin berdasarkan damage yang diberikan, bertahan lebih lama, dan berbagai faktor lainnya.

### 2. Method Method yang ada pada Robocode Tank Royale

Method - Method Dasar Robocode Tank Royale:

| **Method - Method Dasar** |
|---|
| **Run()** |
| **Start()** |
| **Go()** |

Method - Method Kontrol Gerakan Bot

| **Method - Method Kontol Gerakan Bot** |
|---|
| **Forward(double distance)** |

| |
|---|
| **Back (double distance)** |
| **SetForward(double distance)** |
| **SetBack(double distance)** |
| **TurnLeft(double degrees)** |
| **TurnRight(double degrees)** |
| **SetTurnRight(double degrees)** |
| **SetTurnLeft(double degrees)** |

Method - Method Kontrol Senjata dan Radar

| **Method - Method Kontol Gerakan Bot** |
|---|
| **TurnGunLeft(double degrees)** |
| **TurnGunRight(double degrees)** |
| **SetTurnGunLeft(double degrees)** |
| **SetTurnGunRight(double degrees)** |
| **TurnRadarLeft(double degrees)** |
| **TurnRadarRight(double degrees)** |
| **SetTurnRadarLeft(double degrees)** |
| **SetTurnRadarRight(double degrees)** |
| **SetAdjustGunForRobotTurn(boolean independent)** |
| **SetAdjustRadarForGunTurn(boolean independent)** |

Metode Menembak dan Interaksi dengan Musuh

| **Method - Method Menembak dan Interaksi dengan Musuh** |
|---|
| **Fire(double power)** |
| **SetFire(double power)** |
| **FireBullet(double power)** |

| |
|---|
| **GetGunHeat()** |
| **SetAdjustRadarForGunTurn(boolean independent)** |

Metode Informasi Status Bot

| **Method - Method Informasi Status Bot** |
|---|
| **getX()** |
| **getY()** |
| **getVelocity()** |
| **getHeading()** |
| **getGunHeading()** |
| **getRadarHeading()** |
| **getEnergy()** |
| **getBattleTime()** |
| **getGunTurnRemaining()** |
| **getRadarTurnRemaining()** |

Metode Pemrosesan Event (Event Handling)

| **Method - Method Pemrosesan Event (Event Handling)** |
|---|
| **onScannedBot(ScannedBotEvent event)** |
| **onHitByBullet(HitByBulletEvent event)** |
| **onHitWall(HitWallEvent event)** |
| **onHitBot(HitBotEvent event)** |
| **onBulletHit(BulletHitEvent event)** |
| **onBulletMissed(BulletMissedEvent event)** |
| **onBulletHitBullet(BulletHitBulletEvent event)** |

| |
|---|
| **onRobotDeath(RobotDeathEvent event)** |
| **onWin(WinEvent event)** |
| **onDeath(DeathEvent event)** |

Metode Lanjutan

| **Method - Method Lanjut** |
|---|
| **Stop()** |
| **Resume()** |
| **isRunning()** |
| **DistanceTo(double x, double y)** |
| **BearingTo(double x, double y)** |
| **NormalizeRelativeAngle(double angle)** |

### 3. Implementasi Algoritma Greedy pada Bot Robocode

Algoritma greedy merupakan teknik yang memilih solusi terbaik pada setiap langkah tanpa mempertimbangkan konsekuensi jangka panjang, mempertimbangkan langkah lain(melupakan cara lain).

#### a. Elemen Algoritma Greedy dalam Robocode

1. **Himpunan Kandidat (C)**: Tindakan yang dapat dilakukan bot (bergerak, menembak, memindai).
2. **Himpunan Solusi (S)**: Serangkaian tindakan yang telah dipilih oleh bot.
3. **Fungsi Solusi**: Memeriksa apakah strategi bot memberikan hasil terbaik.
4. **Fungsi Seleksi**: Memilih tindakan berdasarkan strategi greedy tertentu.
5. **Fungsi Kelayakan**: Memeriksa apakah tindakan dapat dilakukan tanpa risiko besar.
6. **Fungsi Obyektif**: Memaksimalkan skor yang diperoleh bot.

**b. Strategi Greedy yang Dapat Diterapkan**

1. **Menembak dengan Optimal**
    - Jika musuh dalam jangkauan tembakan, gunakan daya tembak optimal untuk memaksimalkan damage.
    - Pilih daya tembak yang memberikan keuntungan energi terbesar dengan memperhitungkan panas meriam.
2. **Lock musuh terdekat dan serang**
    - Selalu *scan* untuk mendapatkan musuh terdekat.
    - Jika musuh memiliki energi yang lebih rendah, tabrakan tank langsung ke musuh.
3. **Pemanfaatan Radar yang Efektif**
    - Menggunakan strategi pemindaian terus-menerus untuk mendeteksi musuh.
    - Jika tidak ada musuh yang terdeteksi, bot akan bergerak ke arah yang lebih strategis.
4. **Menabrak Musuh Secara Strategis (Ramming)**
    - Jika energi bot lebih tinggi dari musuh dan berada dalam jangkauan dekat, tabrak musuh untuk mendapatkan poin tambahan.
5. **Bertahan untuk Maksimalkan Survival Score**
    - Bot berusaha untuk mencari tempat yang tidak ada musuh dan menghindari segala macam serangan yang akan diberikan oleh lawannya.

**4. Cara Menjalankan Bot Robocode**

1. **Instalasi Robocode Tank Royale**
    - Unduh dan instal Robocode Tank Royale dari github resmi Robocode Tank Royale
    - Pastikan Java, C#, dotnet yang sesuai sudah diinstal di sistem.
2. **Membuat Bot**
    - Buat bot dengan metode untuk bergerak, menembak, dan memindai musuh.
    - Implementasikan strategi greedy pada setiap aksi bot.
3. **Menjalankan Simulasi**

- Kompilasi kode dengan melakukan "./<nama bot>.cmd pada folder yang berisi .cmd untuk meng compile botnya. Jalankan aplikasi dengan file jarnya.
- Uji bot dengan berbagai strategi untuk menemukan kombinasi yang optimal.

4. **Optimasi dan Penyempurnaan**
    - Evaluasi hasil pertandingan dan lakukan perbaikan berdasarkan kelemahan yang ditemukan.
    - Sesuaikan parameter greedy untuk meningkatkan skor dan daya tahan bot.

Dengan pendekatan greedy, bot dapat mengambil keputusan cepat dan efisien untuk bertahan dan mendapatkan skor tertinggi dalam pertandingan Robocode.

## 2.3 Struktur Kerja Program pada Robocode Tank Royale

Berbeda dengan Robocode Classic yang hanya menggunakan Java, Robocode Tank Royale menggunakan gabungan bahasa dari Java dan Csharp untuk dapat menjalankan gamenya sesuai ketentuan gamenya. Pastinya membingungkan bagaimana Java dan Csharp dapat saling melakukan inheritance satu sama lainnya. Java dalam Robocode Tank Royale digunakan untuk menjalankan game utama, karena Robocode Tank Royale berjalan di Java 11 atau lebih baru dan dapat dijalankan di Windows, macOS, dan Linux. Selain itu, Java juga digunakan untuk mengembangkan bot yang berpartisipasi dalam pertandingan menggunakan Bot API untuk Java (JVM).

Dengan Bot API untuk Java, bot dapat berkomunikasi dengan game server melalui WebSocket API menggunakan JSON, tanpa perlu menangani detail komunikasi secara manual. Selain Java, bahasa lain yang berbasis JVM seperti Kotlin, Scala, Groovy, Jython, dan Clojure juga dapat digunakan untuk membuat bot.

Di sisi lain, Csharp dalam Robocode Tank Royale digunakan untuk mengembangkan game engine dan server pertandingan, serta menyediakan Bot API untuk platform .Net. Bot yang dibuat dengan Csharp (atau bahasa lain di ekosistem .Net seperti F# dan Visual Basic) juga menggunakan WebSocket untuk berkomunikasi dengan server.

Tidak ada konsep inheritance langsung antara Java dan Csharp dalam Robocode Tank Royale. Sebaliknya, bot dari berbagai bahasa berjalan secara independen dan berkomunikasi dengan server menggunakan protokol yang sama. Ini memungkinkan bot dari Java dan Csharp (serta

bahasa lain yang mendukung WebSocket API) untuk bertarung dalam satu arena tanpa bergantung pada pewarisan kelas satu sama lain.

# BAB 3

# Aplikasi Strategi Greedy pada Bot

## 3.1 Mapping Persoalan Greedy yang ada pada Robocode Tank Royale

Dalam Robocode Tank Royale, algoritma Greedy bisa menjadi pendekatan yang efektif untuk mengambil keputusan secara cepat tanpa harus mengevaluasi semua kemungkinan yang ada. Pendekatan ini bekerja dengan memilih solusi terbaik secara lokal pada setiap langkah dengan harapan mendapatkan solusi optimal secara global. Dalam konteks ini, kita bisa menerapkan algoritma Greedy untuk berbagai aspek strategi pergerakan dan serangan tank.

Untuk itu kita harus menganalisis 6 elemen greedy yang harus diperhatikan dalam penerapannya pada Robocode Tank Royale. Berikut adalah 6 elemen greedy yang harus ditinjau.

1. Himpunan Kandidat: Himpunan kandidat adalah seluruh aksi yang dapat dilakukan oleh tank dalam permainan, seperti maju, mundur, berputar, menghindar, atau menembak. Setiap aksi yang tersedia merupakan opsi dalam proses pengambilan keputusan, dan bot harus memilih salah satu berdasarkan strategi yang diterapkan.

2. Himpunan Solusi: Himpunan solusi terdiri dari serangkaian keputusan yang telah diambil selama permainan. Setiap keputusan yang dibuat baik itu menghindar, menyerang, atau bergerak ke posisi tertentu akan membentuk strategi keseluruhan yang akan menentukan performa bot dalam pertempuran.

3. Fungsi Solusi: Fungsi yang menentukan apakah himpunan solusi yang telah terbentuk memenuhi kriteria kemenangan atau tidak. Dalam Robocode, misalnya, solusi dapat dianggap valid jika bot mampu bertahan lebih lama dan memberikan damage yang cukup pada lawan, bot mampu mendapatkan poin yang maksimal sehingga mendapatkan ranking yang tinggi.

4. Fungsi Seleksi: Dari semua fungsi solusi yang tersedia, fungsi ini memilih yang paling menguntungkan berdasarkan informasi terbaru, seperti memilih untuk menembak jika musuh dalam jangkauan atau menghindar jika berada dalam ancaman serangan.

5. Fungsi Kelayakan: Memastikan bahwa aksi yang dipilih masih dalam batasan permainan. Misalnya, tank tidak boleh keluar dari arena, menembak ketika kehabisan energi, atau terus bergerak ke arah yang justru membahayakan dirinya sendiri.

6. Fungsi Objektif: Menentukan tujuan utama dari strategi Greedy, misalnya memaksimalkan damage ke musuh sambil meminimalkan risiko terkena tembakan. Setiap langkah yang diambil harus selalu mendekatkan bot ke tujuan tersebut. Inti dari segala diatas adalah untuk mendapatkan poin yang maksimal untuk memenangkan game.

## 3.2 Bot B3040PKK

### 3.2.1 Penjelasan Robot B3040PKK

B3040PKK adalah robot yang mengimplementasikan strategi greedy yang mengunci musuh yang di *scan* pertama kali olehnya. B3040PKK tergolong sebagai robot yang agresif, karena akan sering melakukan ramming dan menembak secara agresif kepada musuh yang sedang di locknya Terinspirasi dari nama kelompok yang berupa plat nomor mobil dari seseorang di anggota kelompok kami, B3040PKK adalah plat nomor salah satu motor dari anggota kami, karena sering melaju dengan tindakan agresif juga di jalan raya, kami pun berinisiatif untuk memberi nama robot ini sesuai dengan plat nomor motor tersebut.

### 3.2.2 Elemen - Elemen Greedy

Tentunya, kami perlu untuk meninjau elemen - elemen greedy yang harus kami terapkan dalam bot ini, karena kami dari awal menginginkan bot yang bertindak secara agresif kami akan

memilih strategi yang agresif. Oleh karena itu beirkut adalah elemen - elemen Greedy yang kami pertimbangkan untuk bot ini:

1. **Himpunan Kandidat:**

Himpunan kandidat dalam strategi greedy ini adalah semua musuh yang terdeteksi oleh bot melalui *scan*. Setiap musuh yang muncul dalam jangkauan *scan* menjadi kandidat potensial untuk dikunci dan diserang. Bot tidak akan mempertimbangkan musuh yang belum terdeteksi, melainkan hanya berfokus pada musuh pertama yang terlihat. Jika musuh yang kita lock tersebut sudah tidak ditemukan dalam waktu tertentu, bot akan mencari kandidat lain untuk dijadikan target baru. Sesuai dengan greedy, yang **melupakan** segala kemungkinan lain dan hanya mempertimbangkan satu *enemy* yang kita lock tadi.

2. **Himpunan Solusi:**

Himpunan solusi dalam strategi greedy untuk bot B3040PKK adalah semua kemungkinan urutan tindakan yang dapat dilakukan bot untuk menargetkan dan menyerang musuh dengan agresif. Himpunan solusi ini mencakup:

1. **Menemukan target pertama** → Bot akan mengunci musuh pertama yang terdeteksi oleh *scan*.
2. **Menyerang target yang di lock** → Bot akan langsung menembak ke arah target yang telah dikunci.
3. **Melakukan ramming ke target** → Bot akan bergerak menuju musuh dengan tujuan menabrak lawan sambil tetap menembak yang disesuaikan terhadap jarak besar tembakannya.
4. **Mengevaluasi durasi lock (10 turn)** → Jika setelah 10 turn target tidak lagi terscan, bot akan mencari target baru.
5. **Mengunci target baru** → Jika target baru ditemukan, proses kembali ke langkah 2. Jika tidak, bot akan terus mencari target lain.
6. **Mengulang proses hingga semua musuh dikalahkan atau bot mati.**

Himpunan solusi ini memastikan bahwa bot selalu dalam kondisi menyerang dan tidak pernah berhenti untuk menunggu musuh datang.

3. **Fungsi Solusi:**

Fungsi solusi dalam bot ini bertujuan untuk menilai apakah bot berhasil mencapai hasil yang optimal berdasarkan mekanisme pertempuran yang diterapkan. Solusi dianggap berhasil jika:

- Bot mendapatkan poin yang tinggi → Bot harus mengeliminasi sebanyak mungkin musuh dalam waktu sesingkat mungkin untuk memaksimalkan skornya.
- Bot bertahan lebih lama di arena → Semakin lama bot bertahan sambil terus menyerang, semakin besar peluangnya untuk mendapatkan skor lebih tinggi.
- Strategi agresif bekerja dengan baik → Jika bot berhasil terus menyerang tanpa banyak jeda dan tetap menargetkan musuh secara efektif, maka strategi greedy yang digunakan terbukti efisien.
- Musuh yang dikunci berhasil dikalahkan dalam waktu singkat → Jika bot mampu menghabisi musuh yang dikunci tanpa perlu banyak peralihan target, maka pendekatan greedy ini dapat dikatakan optimal.

4. **Fungsi seleksi:**

Fungsi seleksi pada Bot B3040PKK adalah:

- Bot akan memilih musuh pertama yang terdeteksi oleh scan.
- Jika target tidak bisa dikalahkan dalam 10 turn, bot akan mencari target baru yang pertama terdeteksi berikutnya.
- Bot selalu fokus menyerang target yang telah dikunci dan melakukan ramming karena poin dari ramming yang besar sebelum berpindah ke target lain.

5. **Fungsi kelayakan:**

Musuh yang dipilih masih ada di arena dan belum mati.

- Jika musuh yang dikunci sudah mati, maka target tidak lagi layak, dan bot harus menunggu 10 turn sebelum melakukan scan ulang untuk mencari target baru.

Musuh masih berada dalam jangkauan bot.

- Jika musuh bergerak terlalu jauh dan keluar dari jangkauan bot, maka target tersebut tidak lagi layak untuk diserang, dan bot harus mencari target baru.
- Jika musuh berada di area yang tidak bisa diakses atau terhalang rintangan yang tidak bisa ditembus, bot akan tetap mengejar sebelum akhirnya mencari target lain setelah musuh mati.

Bot hanya boleh mengunci satu musuh dalam satu waktu.

- Jika bot sudah mengunci musuh, maka target tersebut tetap layak selama belum mati atau keluar dari jangkauan.

- Bot tidak boleh mengganti target sebelum musuh yang dikunci dikalahkan.

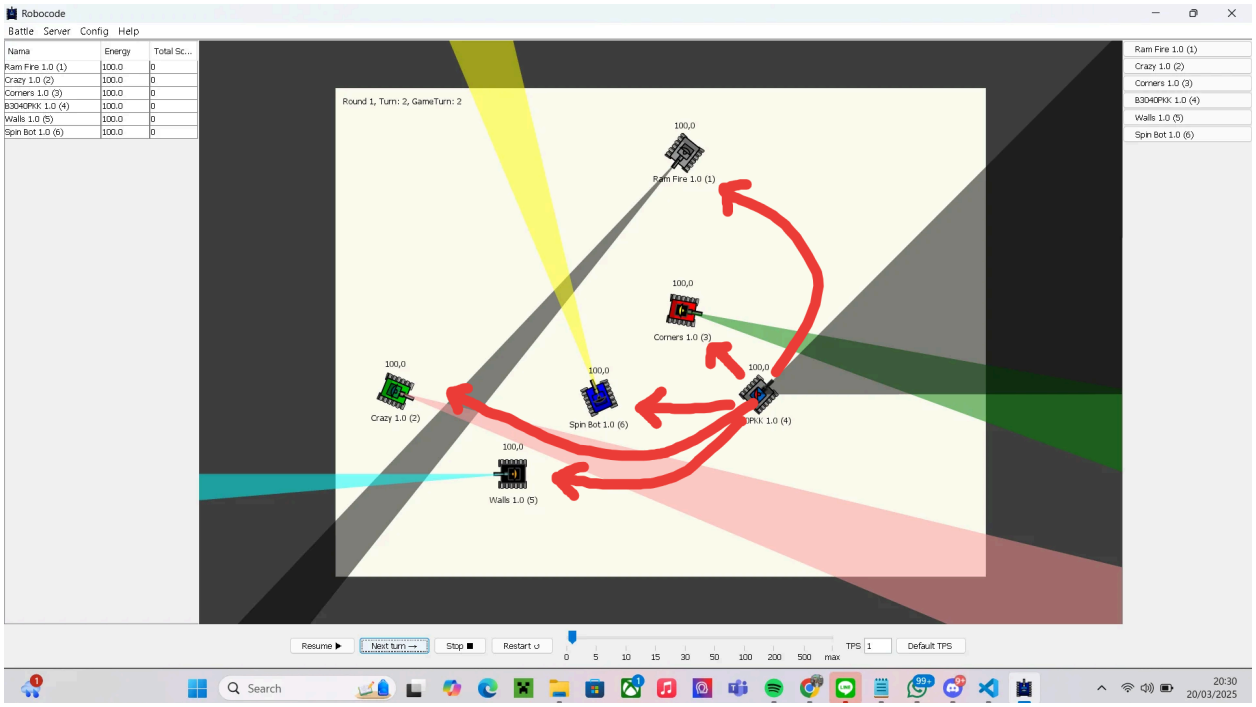
Jika target sudah mati, bot akan menunggu 10 turn sebelum mencari musuh lain.

- Setelah target mati, bot tidak langsung melakukan scan ulang, tetapi menunggu **10 turn** sebagai bagian dari strateginya.
- Jika setelah 10 turn tidak ada musuh baru yang terdeteksi, bot akan terus bergerak dan melakukan scan ulang sampai menemukan target baru.

6. **Fungsi obyektif:**

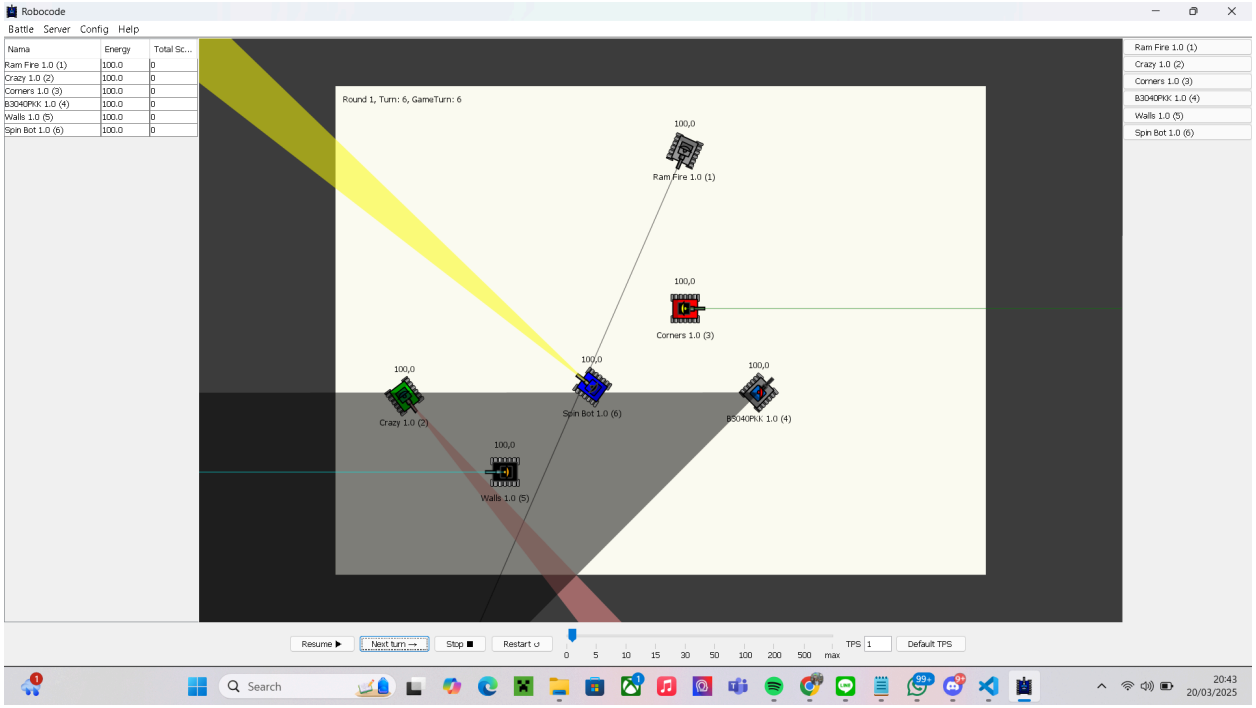
Fungsi objektif dalam hal ini tentunya adalh fungsi yang memberikan nilai teroptimal, nilai yang paling tinggi, poin dalam survival point, ram damage, dan bullet damage yang tinggi.

### 3.2.3 Pencarian Greedy yang tepat



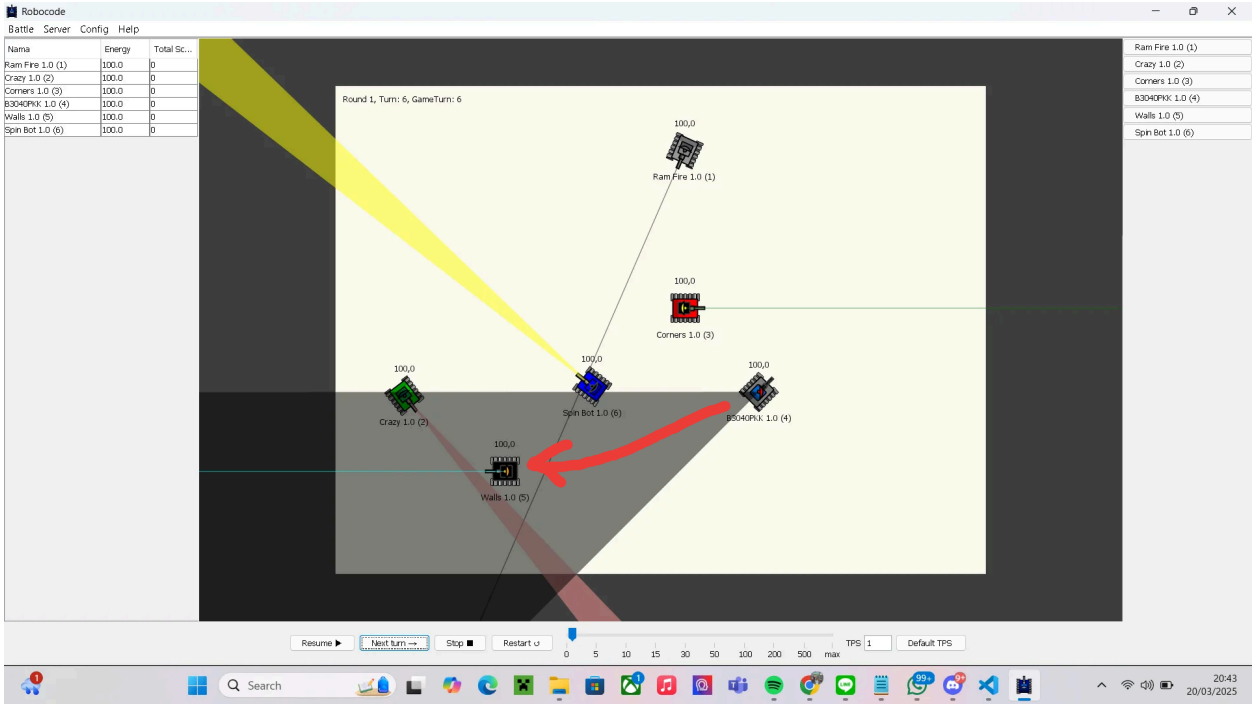

*Gambar 2. Himpunan Kandidat dalam Robocode Tank Royale*

Gambar diatas menunjukkan bahwa terdapat banyak macam kandidat yang dapat kita lakukan locking, dalam hal ini kemudian kita akan melakukan locking terhadap musuh yang pertama kali kita scan.

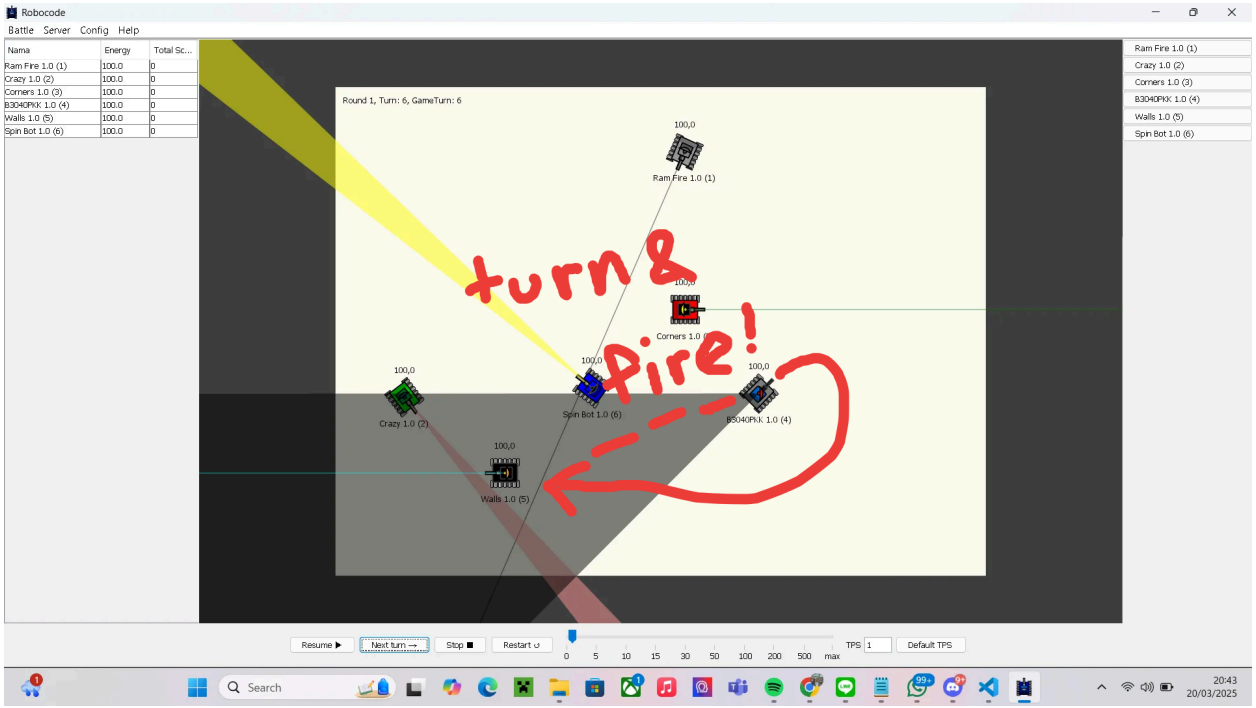



*Gambar 3. Locking kepada musuh yang discan pertama kali*

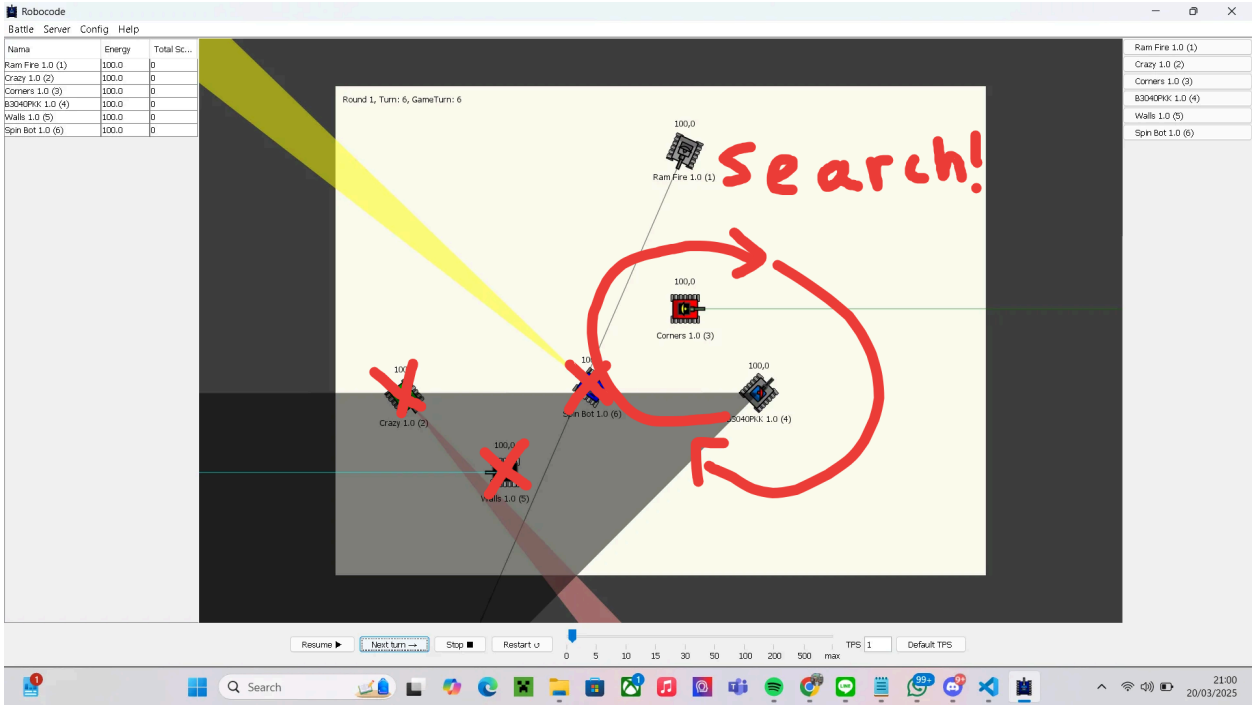
Implementasi bot nantinya diharapkan dapat melakukan locking secara permanen kepada musuh yang didapatkan pertama kalinya melalui radarnya.




*Gambar 4. Bot diharapkan dapat melakukan locking pertama kali mendapatkan scan musuh*



*Gambar 5. Bot diharapkan dapat menghadap musuh dan lock musuh dan ram hingga titik darah penghabisan*




*Gambar 6. Bila musuh sudah yang awalnya kita lock sudah tidak ditemukan, putar terus radar hingga menemukan musuh baru dan lakukan locking ke musuh tersebut*

### 3.2.4 Efektivitas dan Efisiensi

Strategi greedy yang diterapkan dalam bot B3040PKK memiliki efisiensi tinggi dalam memilih target dengan cepat dan memastikan serangan agresif secara konsisten. Dengan langsung mengunci musuh pertama yang terdeteksi, bot mengurangi waktu pencarian target dan tetap fokus menyerang tanpa mempertimbangkan alternatif lain. Pendekatan ini juga membuat pergerakan bot lebih efisien karena selalu menuju target untuk melakukan ramming, sekaligus menyesuaikan daya tembakan berdasarkan jarak guna memaksimalkan damage. Namun, strategi ini memiliki keterbatasan dalam fleksibilitas, terutama jika musuh yang dikunci bergerak terlalu cepat atau sulit dikejar, sehingga bot kehilangan momentum serangan dan harus melakukan pencarian ulang setelah 10 turn. Selain itu, ketergantungan pada deteksi awal membuat bot berisiko mengejar target yang kurang optimal tanpa mempertimbangkan pilihan lain. Dari sisi efektivitas, strategi ini sangat baik dalam skenario pertarungan 1v1 atau melawan musuh yang cenderung statis, tetapi kurang optimal dalam menghadapi lawan yang memiliki mobilitas tinggi atau dalam pertempuran dengan banyak musuh. Alternatif strategi seperti pemilihan target yang lebih dinamis, pergerakan yang lebih adaptif, dan prediksi pergerakan musuh dapat meningkatkan efektivitas bot secara keseluruhan. Dengan pendekatan yang lebih fleksibel, bot

dapat bertahan lebih lama di arena dan meningkatkan peluang kemenangan dalam berbagai kondisi pertempuran.

## 3.3 Bot D1753ALG

### 3.3.1 Penjelasan Robot D1753ALG

D1753ALG adalah robot yang menggunakan strategi greedy yang mengoptimasi pencarian musuh, ,mengoptimasi akurasi penembakan, mengoptimasi peluru, dan meminimalkan peluru musuh. Sifat dari D1753ALG adalah agresif tetapi tetap bermain pintar.

### 3.3.2 Elemen - Elemen Greedy

1. **Himpunan Kandidat:**

   Himpunan kandidat yang dari bot ini ada beberapa, untuk penembakan dan pencarian himpunan kandidatnya adalah sudut untuk menembak musuh, untuk peluru adalah besarnya, dan untuk pergerakkan adalah sudut arah bot dan tipe gerakan yang akan dilakukan.

2. **Himpunan Solusi:**

   Himpunan solusi yang dari bot ini ada beberapa, untuk penembakan dan pencarian himpunan kandidatnya adalah sudut untuk yang optimal agar tembakan mengenai musuh, untuk peluru adalah ukuran yang optimal , dan untuk pergerakkan adalah sudut arah bot dan tipe gerakan yang akan dilakukan.

3. **Fungsi Solusi:**

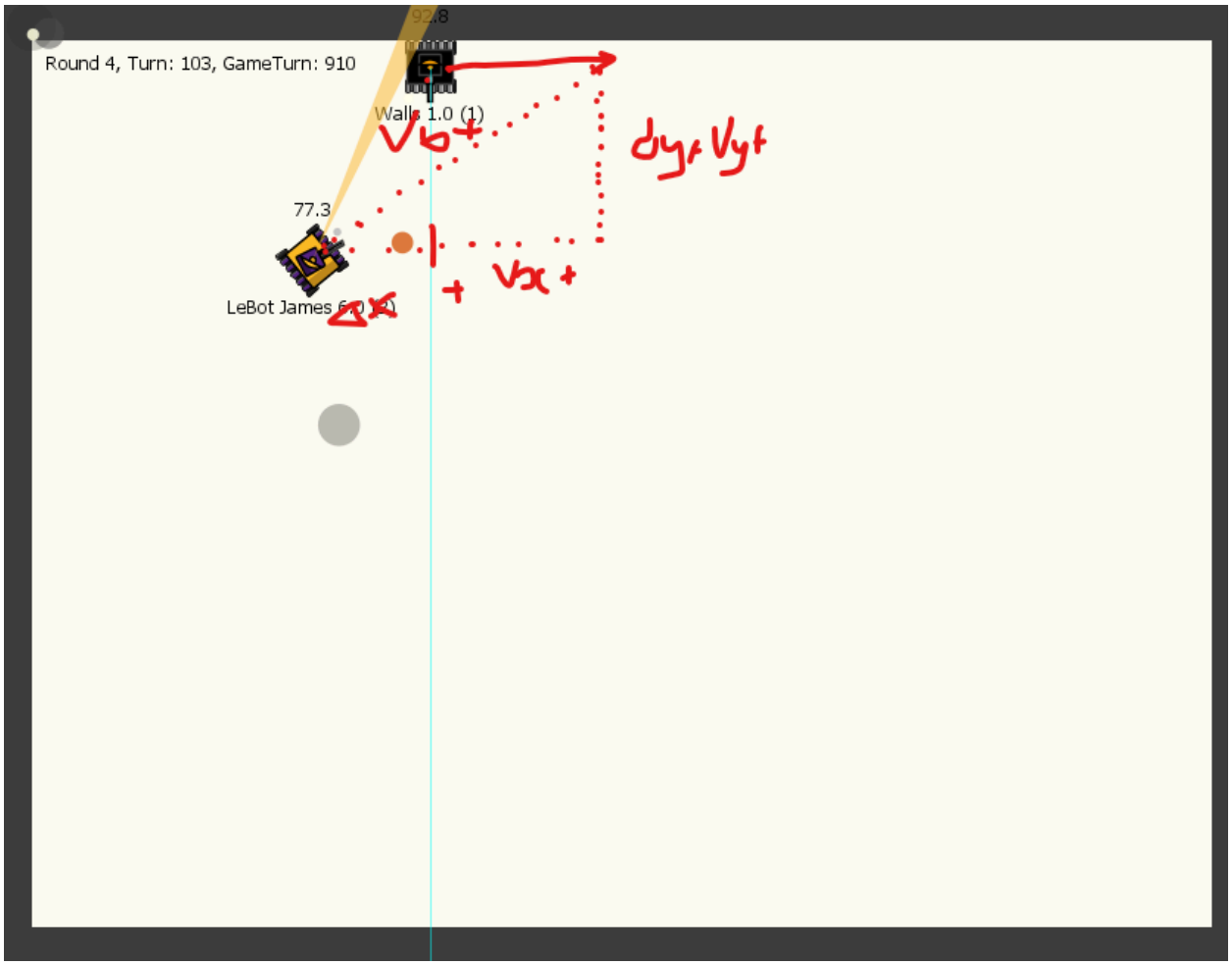
   Fungsi solusi dari bot ini adalah agar bot mendapatkan poin tertinggi, poin yang terkena minimal, waktu pada arena terlama.

4. **Fungsi Seleksi:**

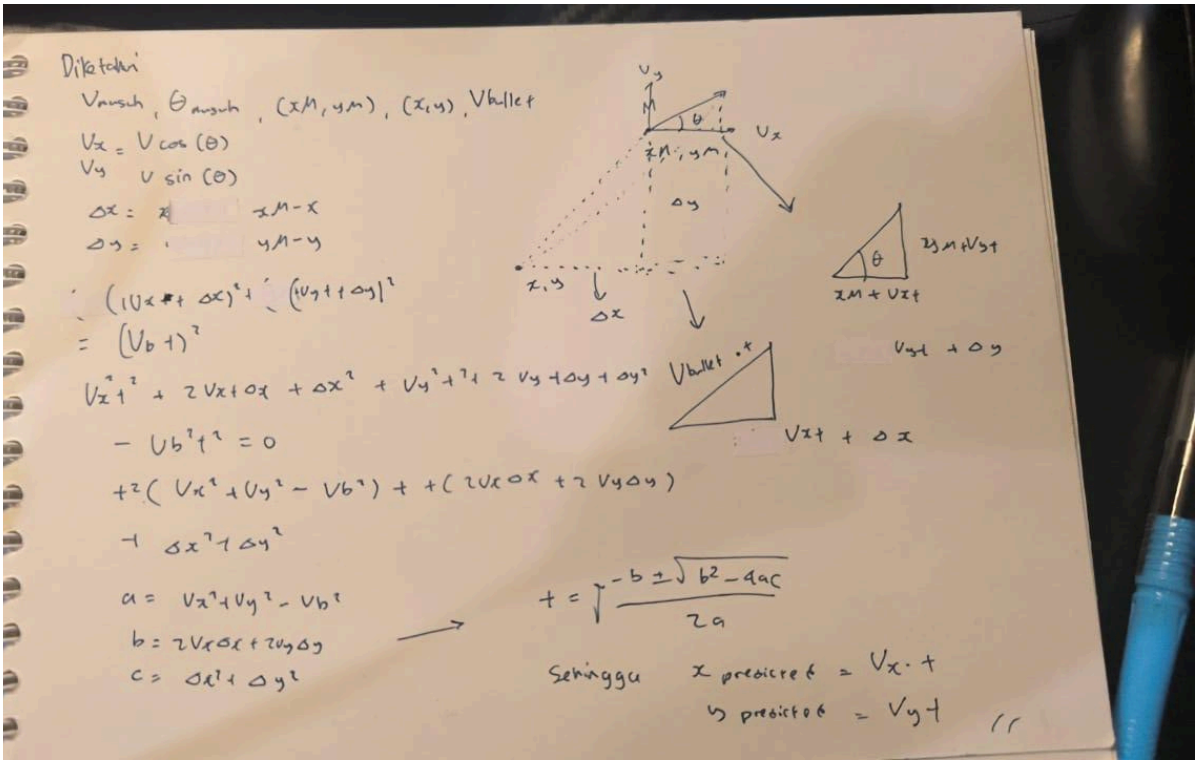
   Fungsi seleksi terdapat beberapa sebagai berikut:

   1. Pencarian musuh: pengambilan sudut pencarian berikutnya adalah berdasarkan lokasi musuh sebelumnya agar radar dapat mengikuti musuh dengan cara seakurat mungkin.

2. Penembakan musuh: sudut penembakan didapatkan menggunakan data musuh yang terdeteksi pada radar yang mengcakup lokasi, arah, kecepatan, dan kecepatan peluru yang akan ditembakkan




*Gambar 7. Contoh Bagaimana Bot Mengkalkulasi Lokasi yang Dituju Musuh*




*Gambar 8. Formulasi Rumus Penembakkan Bot*

3. Peluru yang ditembakkan: pengambilan ukuran peluru yang ditembakkan berdasarkan jarak dengan musuh sehingga apabila musuh yang diserang lebih dekat, peluru yang digunakan akan peluru yang lebih besar karena lebih besar kemungkinan mengenai musuh saat dekat.
4. Pergerakan: karena pada pergerakan ingin membuat bot dipermudah untuk mengenai musuh saat penembakkan namun saat bersamaan menghindari peluru musuh, maka pergerakkan akan mendekati musuh sampai jarak 200 satuan dan setelah sampai bergeser tegak lurus dari musuh agar menghindari tembakkan musuh.

**5. Fungsi Kelayakan:**

1. Pencarian musuh: jika sudut yang dipilih sesuai dengan rumus dan dekat dari lokasi sebelumnya maka sudut berputar tersebut valid.
2. Penembakkan musuh: jika bot yang dipilih merupakan bot yang terlihat pada radar dan lokasi yang terprediksi tidak melebihi batas peta, maka lokasi tersebut valid.

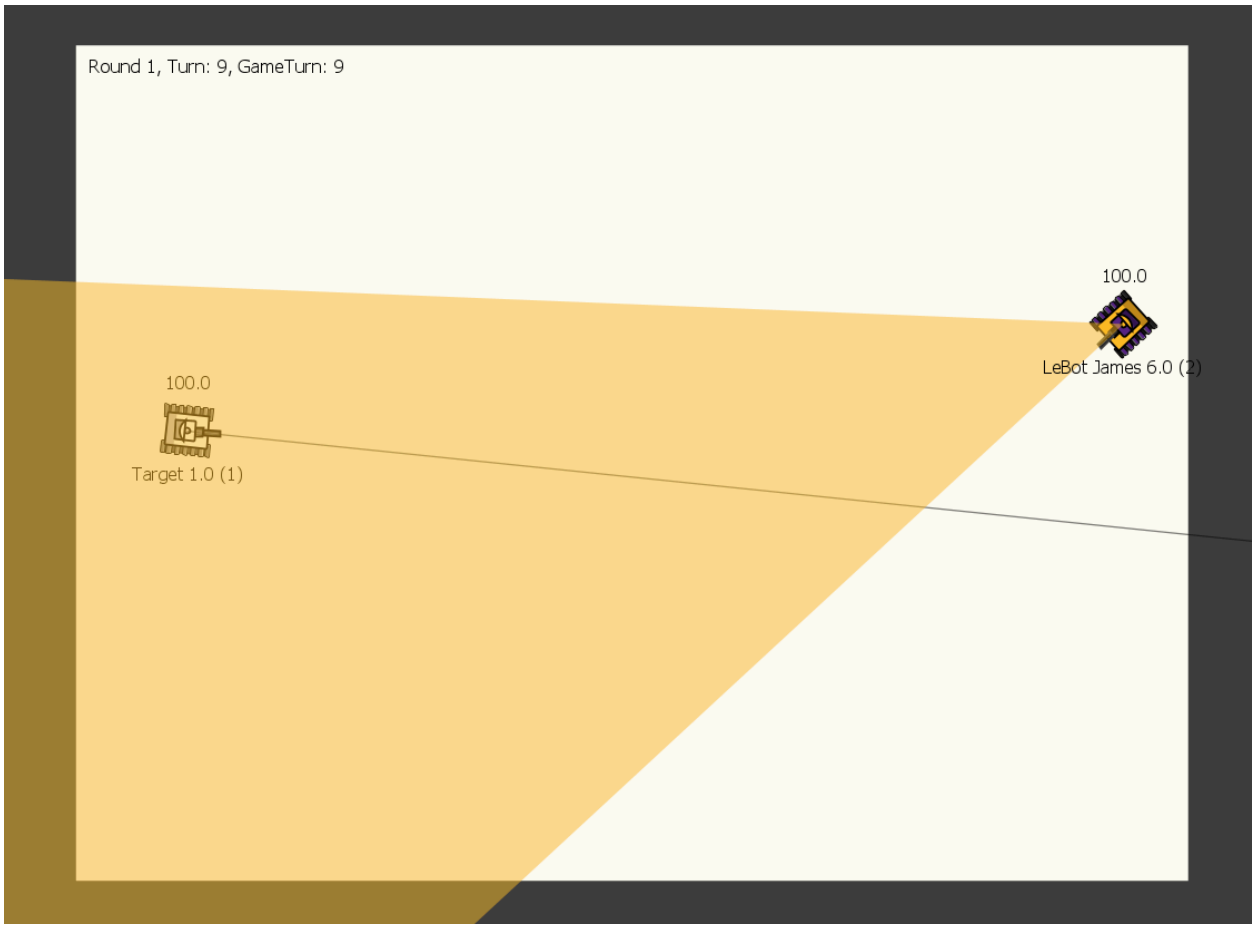
3. Peluru yang ditembakkan: pengambilan peluru harus sesuai dengan rumus dan tidak melebih 3 dan tidak di bawah 0.1.
4. Pergerakan: walaupun pergerakan cenderung tetap mendekati bot musuh, tapi tetap menjaga jarak dengan musuh sebanyak 200 satuan agar bot musuh tidak terbantu dengan ramming.

**6. Fungsi Obyektif:**

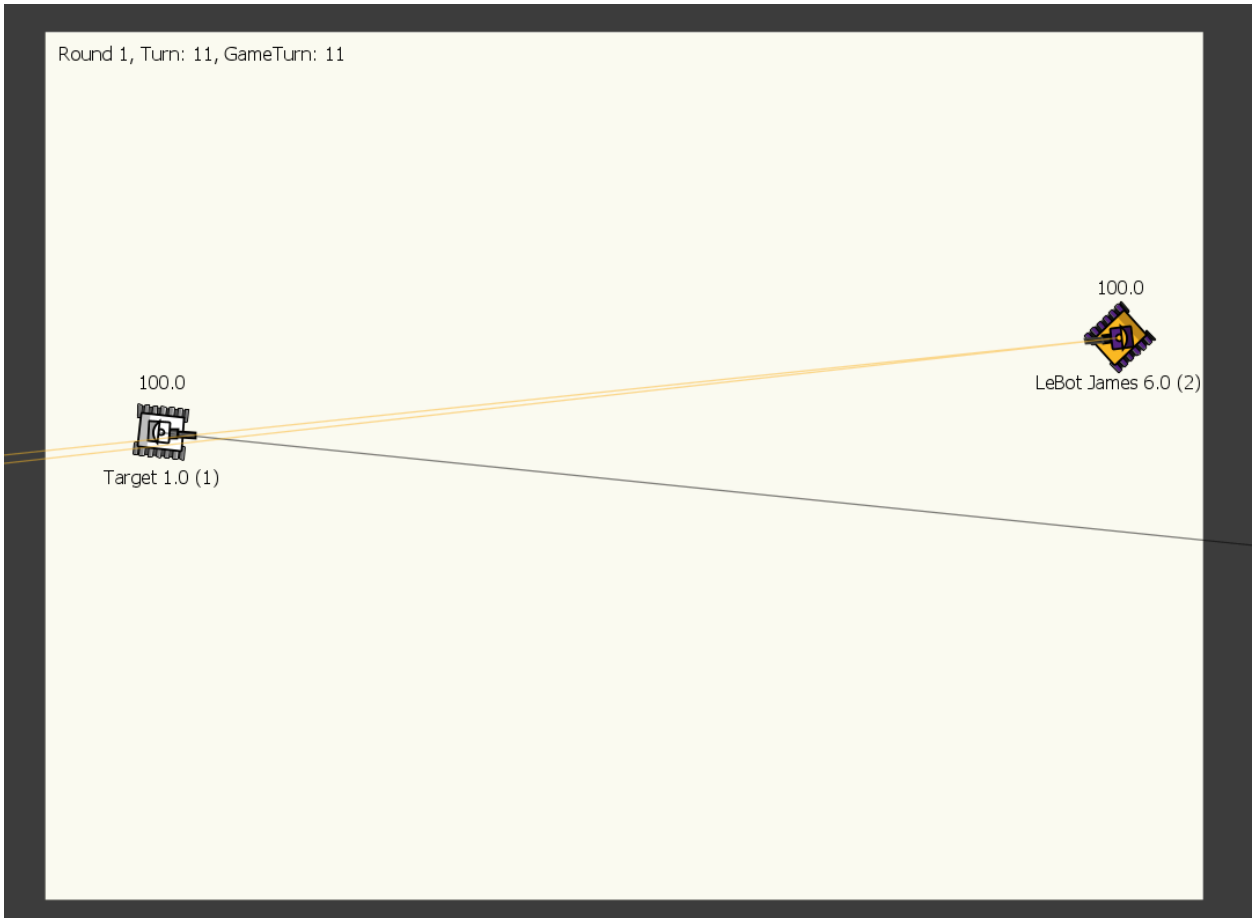
Pada keseluruhan onjektif dari bot ini adalah untuk memaksimalkan poin dan lama dalam arena beserta meminimalkan peluru yang terkena, untuk spesifikasi setiap komponen sebagai berikut:

1. Pencarian Musuh: memaksimalkan waktu radar yang terkena musuh agar informasi yang didapatkan dari musuh maksimal juga.
2. Penembakkan musuh: memaksimalkan akurasi dari penembakkan karena setiap penembakkan yang tidak kena akan mengurang poin dan mengenai peluru akan menambahkan poin.
3. Peluru yang ditembakkan: memaksimalkan poin dan akurasi karena semakin besar peluru semakin lama kecepatannya sehingga semakin jauh prediksi akan kurang ampuh sehingga dibuat peluru semakin besar semakin dekat musuh.
4. Pergerakan: memaksimalkan akurasi penembakkan dan meminimalkan peluru yang diterima. Akurasi penembakkan ditingkatkan dengan mendekati musuh dan peminimalan peluru yang diterima dilakukan dengan cara terus bergerak.

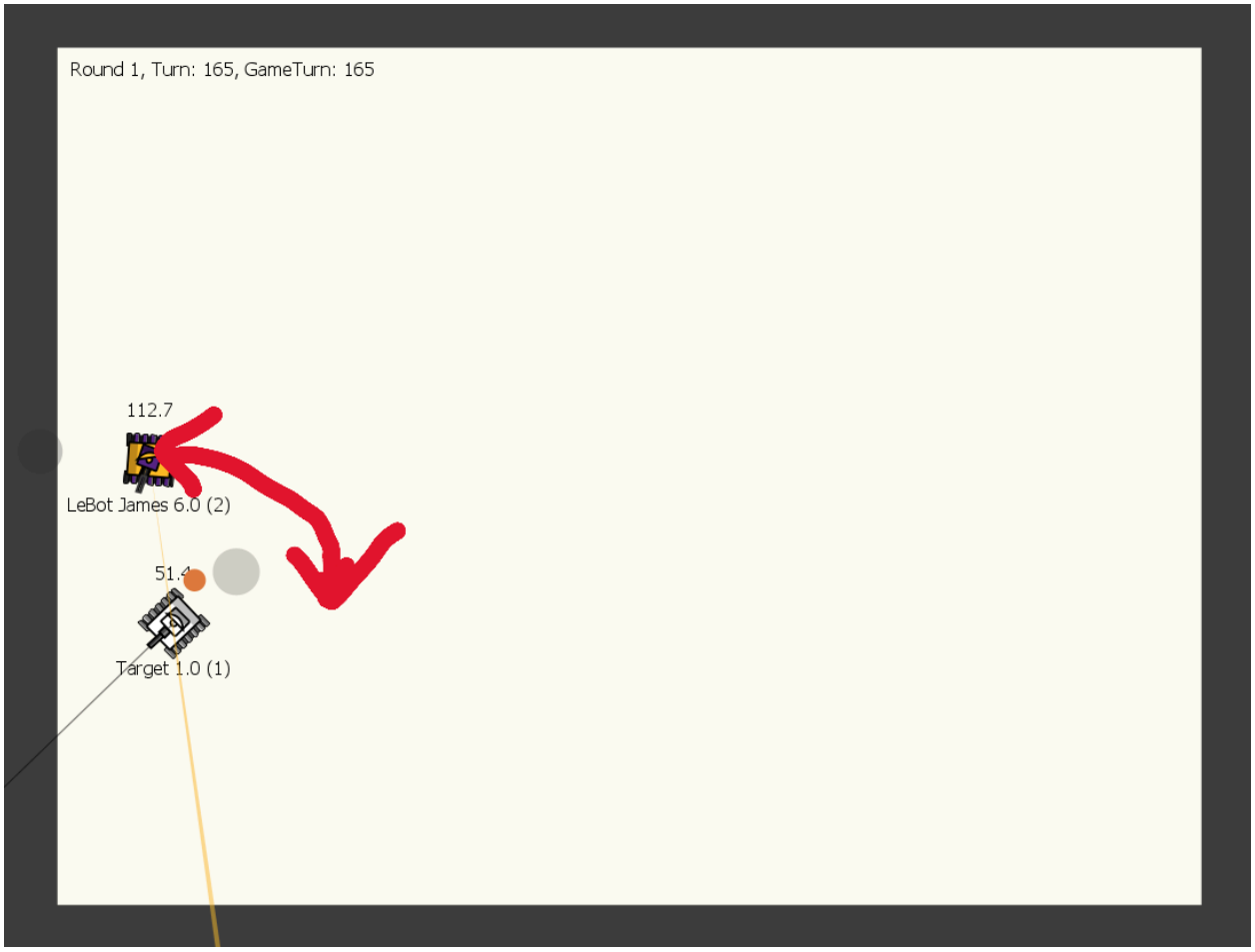
### 3.3.3 Pencarian Greedy yang tepat




*Gambar 9. Pencarian Musuh dengan Cara Mengscan Sekitar*




*Gambar 10. Mendekati Musuh*




*Gambar 11. Bergerak untuk Menghindar dari Musuh*

### 3.3.4 Efektivitas dan Efisiensi

Penaplikasian algoritma greedy pada robo D1753ALG menghasilkan hasil yang secara keseluruhan efektif karena dapat menentukan semua pilihannya dengan O(1). Dari segi penyerangan robot ini sudah sangat efektif karena algoritma pencarian dan penembakannya. Keefektivitas robot ini ditingkatkan ketika dilawankan dengan robot yang diam atau memiliki pergerakan yang konstan. Pada segi pertahanan robot ini efisien dalam menghindari tembakan musuh dengan cara robot ini bergerak yang memiliki unsur acak. Namun terdapat juga kelemahan dari robot ini, yaitu ketika dihadapkan dengan musuh yang menggunakan strategi menabrak robot ini dapat tersangkut pada tabrakan robot musuh tersebut.

## 3.4 Bot F3812FIO

### 3.4.1 Penjelasan Robot F3812FIO

F3812FIO adalah robot yang menggunakan statistik pergerakan musuh dan menentukan perpindahan musuh untuk optimisasi tembakan terhadap musuh yang cenderung memiliki pola tertentu yang berulang / tidak terlalu random dalam pergerakannya. Pergerakan bot ini sendiri adalah dengan terus mengikuti musuh tetapi diberikan jarak tertentu tidak sampai menabrak. Penamaan robot ini mirip seperti Bot B3040PKK dimana menggunakan salah satu plat nomor motor anggota kami.

### 3.4.2 Elemen - Elemen Greedy

Dalam penyimpinan statistik musuh dan optimisasi pertempuran dalam robocode ini, penjelasan elemen-elemen greedy yang digunakan adalah sebagai berikut:

1. **Himpunan Kandidat:**

   Bot akan mendeteksi musuh yang muncul di radar dan menyimpan informasi dalam struktur data List/Array, di mana setiap musuh diidentifikasi berdasarkan ID mereka. Setiap entri menyimpan data statistik pergerakan seperti posisi dalam 1000 gerakan terakhir, kecepatan, arah, perubahan arah, dan posisi musuh. Dengan cara ini, robot dapat menentukan pergerakan setiap musuh secara terpisah.

2. **Himpunan Solusi:**

   Bot akan menentukan strategi terbaik berdasarkan data yang telah dikumpulkan. Bot akan memilih target utama, yaitu musuh pertama yang terdeteksi, dan mengunci radar. Radar tetap terkunci hingga musuh tersebut hilang dari radar sebanyak tiga kali berturut-turut. Jika bot musuh yang dikunci hilang, bot akan mencari target lain yang masih terdeteksi.

3. **Fungsi Solusi**

   Memastikan bahwa setiap langkah yang diambil bot tetap dalam batasan optimal. Jika jarak dengan musuh terlalu dekat, robot akan mundur untuk menghindari tabrakan. Sebaliknya, jika musuh berada terlalu jauh, bot akan bergerak mendekat. Pergerakan ini tidak dilakukan secara lurus, tetapi dengan sedikit belokan agar tidak terkena peluru dari arah yang dituju.

4. **Fungsi Seleksi**

Prediksi posisi musuh dihitung menggunakan data 10 gerakan terakhir. Terdapat fungsi yang memilih *best match index* atau perbedaan terkecil pada gerakan terakhir dengan statistik pergerakan musuh. Terdapat pertimbangan bahwa semakin jauh musuh, semakin maju frame pergerakan (perpindahan) yang digunakan dalam prediksi untuk meningkatkan akurasi tembakan.

5. **Fungsi Kelayakan**

   Setiap keputusan bot tetap valid dan tidak melanggar batasan pergerakan yang ada. Bot harus menjaga jarak aman dengan musuh agar tidak terjadi tabrakan, tetapi tetap berada dalam posisi yang memungkinkan untuk menyerang dan bergerak secara bebas..

6. **Fungsi Objektif**

   Bot berfokus pada optimalisasi serangan agar tembakan lebih efektif. Bot akan menyesuaikan kekuatan tembakan berdasarkan jarak musuh, dengan daya tembak yang lebih besar jika musuh berada dalam jarak dekat. Selain itu, bot tidak menargetkan posisi musuh saat ini secara langsung tetapi perkiraan posisi musuh beberapa frame kedepannya.

### 3.4.3 Efektivitas dan Efisiensi

Strategi greedy yang diterapkan dalam bot F3812FIO memiliki efisiensi tinggi ketika dihadapkan dengan bot-bot yang cenderung memiliki pola gerakan yang sama. Akurasi prediksi pada bot musuh yang bertipe seperti spinbot, walls, ataupun crazy memiliki akurasi yang cukup tinggi karena pola gerakan yang mereka bentuk. Namun strategi ini masih memiliki kekurangan yang cukup besar ketika dihadapkan ke bot musuh yang cenderung terus mengikuti atau dengan bot yang memiliki tingkat kerandoman yang tinggi. Bot akan kesulitan memprediksi posisi musuh dan terkadang menembak ke ruang yang sangat kosong akibat kesalahan prediksi tersebut.

## 3.5 Bot Jeb_

### 3.5.1 Penjelasan Jeb_

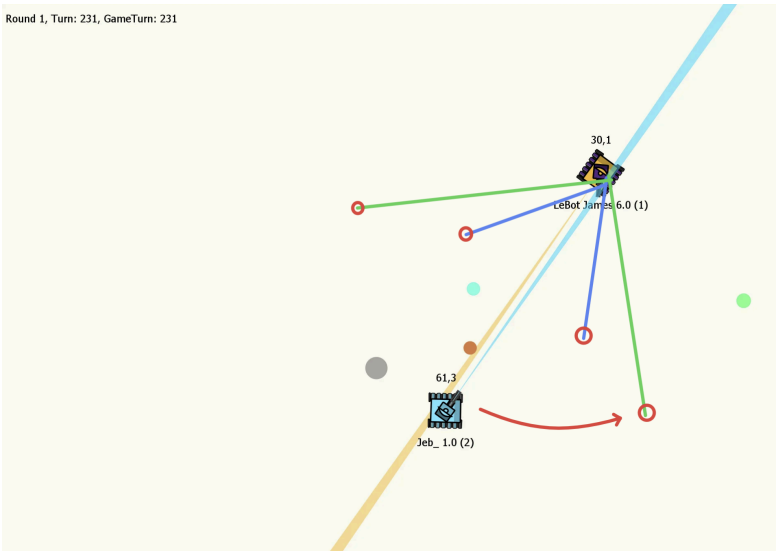
Jeb_ adalah bot yang terinspirasi karena mencoba-coba merubah warna di pertengahan battle dan teringat oleh suatu domba berasal dari Minecraft. Jeb_ sendiri adalah perpaduan dan optimisasi dari bot-bot sebelumnya. Ia mengimplementasikan *Smart Shooter* yang berasal dari Bot D1753ALG di mana ia menggunakan prediksi kecepatan dan arah musuh. Ia juga mengimplementasikan sebagian movement dari Bot F3182FIO yang mengikuti musuh dengan tetap menjaga jaraknya serta sistem *lock scanning* dari Bot B3040PKK.

Secara singkat, optimisasi yang dilakukan untuk mengikuti musuh adalah dengan cara membuat 4 poin lokasi yang dapat dituju oleh bot Jeb_ berdasarkan posisi serta sudut relatif ke musuh. Poin lokasi tersebut dipilih berdasarkan posisi yang valid dan tidak dekat dengan tembok. Optimisasi dalam penembakan juga dilakukan dengan cara mendekati musuh secara bertahap apabila 3 kali *miss* dan kembali ke jarak normal apabila 3 kali *hit*.

### 3.5.2 Elemen - Elemen Greedy

1. **Himpunan Kandidat:**

Bot akan menyimpan 4 titik yang bergantung pada posisi serta sudut relatif musuh dari bot. Titik-titik tersebut merupakan koordinat yang menjadi kandidat untuk menjadi tujuan bot berjalan.




Titik 1 dan 2 (pada garis hijau) merupakan titik dengan sudut relatif 135 derajat dari arah radar bot terhadap musuh dengan jarak *d* (dapat berubah-ubah). Titik 3 dan 4 merupakan titik dengan sudut relatif 160 derajat.

2. **Himpunan Solusi:**

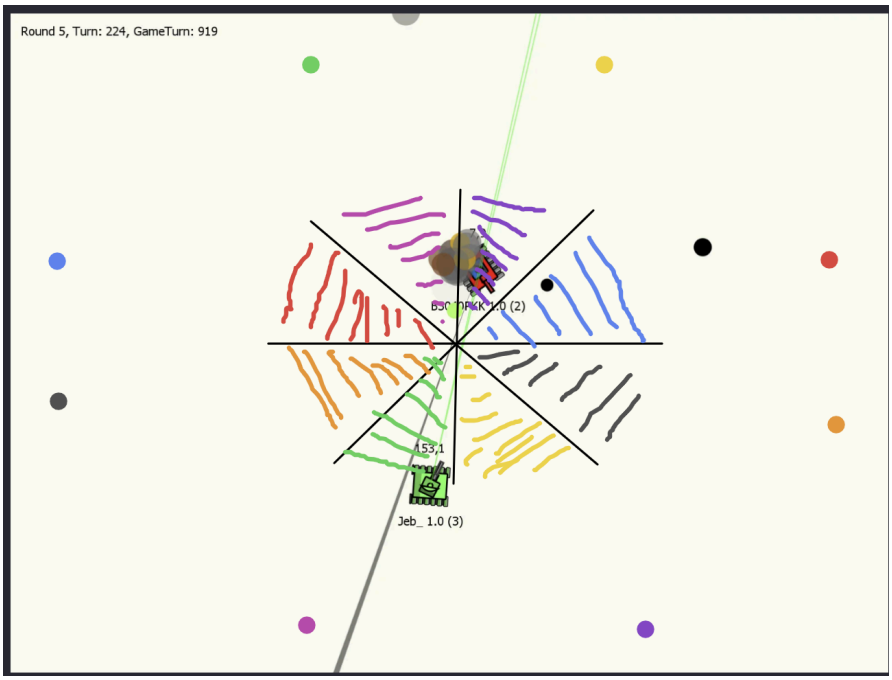
Bot akan memilih semua titik-titik yang valid yaitu titik yang tidak terlalu mendekati tembok. Titik 1 dan 2 sebagai prioritas utama dan titik 3 dan 4 ketika hanya 1 dan 2 tidak bisa dikunjungi/tidak valid. Sama seperti Bot F3812FIO, bot akan memilih target utama, yaitu musuh pertama yang terdeteksi, dan mengunci radar hingga hilang dari radar sebanyak 3 kali berturut-turut.

### 3. Fungsi Solusi

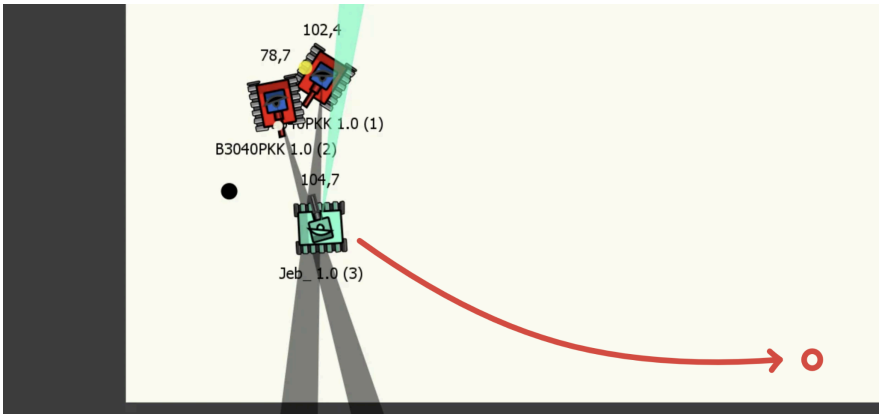
Bot akan terus mengecek validitas posisi dari titik-titik yang ada selama ia mengunci radarnya pada musuh. Ketika semua titik yang ada tidak valid, maka tandanya bot sedang berada di dekat pojok arena dan akan beralih ke mode *runAway()*.

### 4. Fungsi Seleksi

Ketika sedang mode *runAway()*, bot akan memilih titik-titik sesuai sudut relatif terhadap musuh yang sedang dikunci oleh radar. Ada 8 titik kabur yaitu sebagai berikut yang ditandai oleh warna-warna berbeda:




Warna-warna yang direpresentasikan pada tengah gambar merupakan posisi/sudut relatif musuh berada. Warna tersebut terbagi setiap 45 derajat dari arah bot. Seperti pada di gambar bawah ini bot melihat musuh di arah 45 - 90 derajat (ungu) maka ia akan pergi ke arah posisi titik ungu (500, 40) sampai 4 titik kandidat sebelumnya kembali valid.




Dalam seleksi musuh, di awal-awal bot Jeb_ akan memprioritaskan bot pertama yang ia deteksi tetapi ia akan merubah prioritas seleksinya terhadap bot yang ia tabrak atau yang menabraknya. Pemilihan posisi arah tembakan bot Jeb_ sama seperti pada bot D1753ALG di mana ia akan menghitung kecepatan dan arah musuh serta jarak terhadap bot musuh tersebut.

**5. Fungsi Kelayakan**

Fungsi Bot Jeb_ memiliki default jarak terhadap bot musuh sebesar 250 ketika total bot musuh kurang dari 3 dan 450 ketika total bot musuh lebih dari atau sama dengan 3. Ketika bot Jeb_ berusaha menembak musuh tetapi *miss* sebanyak 2 kali berturut-turut ia akan menganggap jarak tersebut tidaklah tepat dan akan mencoba mengurangi jarak sebesar 30 (merubah nilai *d* menjadi *d-30*) setiap kali ia anggap posisinya tidak tepat. Ia juga akan menambah jarak ketika tertembak musuh atau ia sering mengenai musuh.

**6. Fungsi Objektif**

Objektif dari bot Jeb_ adalah tetap berfokus untuk menjaga jaraknya dan terus bergerak. Ketika dalam situasi ditabrak secara terus menerus, bot Jeb_ akan memasuki fase stuck dan mencoba *escape* dengan cara merubah arah *body* sesuai arah *body* bot musuh yang menabrak lalu maju sebesar 100 diikuti dengan *KeepDistance()* di mana fungsi tersebut mengarahkan bot ke 4 titik kandidat pada bot musuh.

### 3.5.3 Efektivitas dan Efisiensi

Strategi greedy yang diterapkan dalam bot Jeb_memiliki efisiensi yang sangat tinggi terhadap bot musuh yang memiliki gerakan lurus seperti *walls* ataupun bot yang bertipe mengikuti dengan jarak kurang dari 150. Titik-titik yang dituju oleh bot Jeb_ cenderung terus berubah berdasar sudut dan posisi bot musuh sehingga membuat Jeb_

terus bergerak dan terkadang berosilasi atau mengorbit. Gerakan-gerakan tersebut yang membuat bot Jeb_ sulit terkena tembakan yang *direct* ke arahnya. Tetapi bot Jeb_ masih memiliki kelemahan ketika ditandingkan dengan banyak bot. Ditabrak oleh dua bot sekaligus membuatnya bot Jeb_ bingung untuk mempriortiaskan musuh dan sulit untuk *escape* karena terjebak.

# BAB 4

# Implementasi & Pengujian

## 4.1 Bot 3040 PKK

### 4.1.1 Implementasi

Pseudocode dalam notasi pseudocode

```
PROGRAM B3040PKK

VAR
   turnDirection : INTEGER <- 1
   lockedTargetId : INTEGER <- -1 /* artinya tidak ada yang sedang di lock*/
   lostTargetCount : INTEGER <- 0 /* itung udah berapa turn ga dapet musuh */
    maxLostTargetCount : CONSTANT INTEGER <- 3 /* ngelebihin 3 turn artinya langsung dia
ditandain kalo lost target */
   EnergiMusuh : DOUBLE <- 100


   CALL StartBot() /* panggil startbot buat mulai aplikasi */


PROCEDURE StartBot()

   SET BodyColor <- RED
   SET GunColor <- YELLOW
   SET RadarColor <- BLACK

   WHILE IsRunning DO
      IF Energy < 5 THEN
         CALL HindariSerangan()
      ELSE IF RadarTurnRemaining = 0 THEN
         SetTurnRadarRight(INFINITY) /* muter selamanya */
         CALL JauhiTembok()
       IF
      CALL Go() /* jalankan perintah lain yang ada */
```

```
    WHILE


PROCEDURE OnScannedBot(e) /* ketemu bot (radar) */

   IF Energy < 5 THEN
      SetForward(100)
      SetTurnRight(100)
   ELSE IF lockedTargetId = -1 OR lockedTargetId = e.ScannedBotId THEN
      lockedTargetId <- e.ScannedBotId
      lostTargetCount <- 0

      angleToEnemy <- Direction + BearingTo(e.X, e.Y)
      radarTurn <- NormalizeRelativeAngle(angleToEnemy - RadarDirection)
      extraTurn <- MIN(ATAN(40.0 / DistanceTo(e.X, e.Y) * 180.0 / PI), 45)
      radarTurn <- radarTurn + (IF radarTurn < 0 THEN -extraTurn ELSE extraTurn)

      SetTurnRadarLeft(radarTurn)
      CALL HadapTarget(e.X, e.Y)
      distance <- DistanceTo(e.X, e.Y)
      SetForward(distance + 2)

      IF distance < 100 THEN
         Fire(3)
      ELSE IF distance < 300 THEN
         Fire(1.5)
      ELSE
         Fire(1)
       IF
   ELSE
      lostTargetCount <- lostTargetCount + 1
      IF lostTargetCount >= maxLostTargetCount THEN
         lockedTargetId <- -1
         SetTurnRadarRight(INFINITY)
       IF
    IF


PROCEDURE OnHitBot(e)
```

```
    IF e.IsRammed THEN
       lockedTargetId <- e.VictimId
       lostTargetCount <- 0
     IF
    CALL HadapTarget(e.X, e.Y)
    Fire(3)


PROCEDURE OnHitWall(e)

    PRINT "Ouch! I hit a wall, must turn back!"
    CALL JauhiTembok()


PROCEDURE HadapTarget(x, y)

    bearing <- BearingTo(x, y)
    turnDirection <- IF bearing >= 0 THEN 1 ELSE -1
    SetTurnLeft(bearing)


PROCEDURE JauhiTembok()

    margin <- 50
    arenaWidth <- 800
    arenaHeight <- 600
    x <- X
    y <- Y

    IF x < margin AND y > arenaHeight - margin THEN SetTurnRight(45)
    ELSE IF x > arenaWidth - margin AND y > arenaHeight - margin THEN SetTurnLeft(45)
    ELSE IF x < margin AND y < margin THEN SetTurnRight(135)
    ELSE IF x > arenaWidth - margin AND y < margin THEN SetTurnLeft(135)
    ELSE IF x < margin THEN SetTurnRight(90)
    ELSE IF x > arenaWidth - margin THEN SetTurnLeft(90)
    ELSE IF y > arenaHeight - margin THEN SetTurnRight(180)
    ELSE IF y < margin THEN SetTurnLeft(180)
     IF
```

**Pseudocode dalam bahasa indonesia dengan penjelasan:**

```
Mulai program
    Buat bot dengan warna:
        - Body: Merah
        - Gun: Kuning
        - Radar: Hitam
    Selama bot masih berjalan:
        Jika energi kurang dari 5:
            - Putar ke kanan 100 derajat
            - Bergerak mundur 120 unit
        Jika radar tidak lagi berputar:
            - Putar radar ke kanan tanpa batas
            - Hindari tembok
        Jalankan perintah yang sudah disiapkan

Ketika bot mendeteksi bot lain:
    Jika energi kurang dari 5:
        - Maju 100 unit
        - Putar ke kanan 100 derajat
    Jika belum mengunci target atau target masih sama:
        - Kunci target dengan ID musuh
        - Reset hitungan kehilangan target
        - Hitung sudut menuju musuh
        - Sesuaikan radar agar tetap mengunci musuh
        - Hadap ke arah musuh
        - Maju mendekati musuh
        Jika jarak ke musuh < 100:
            - Tembak dengan kekuatan 3
        Jika jarak ke musuh < 300:
            - Tembak dengan kekuatan 1.5
        Jika jarak ke musuh >= 300:
            - Tembak dengan kekuatan 1
    Jika kehilangan target terlalu lama:
        - Reset target
        - Putar radar ke kanan tanpa batas

Ketika bot bertabrakan dengan bot lain:
    Jika bot menabrak lawan secara langsung:
        - Jadikan musuh tersebut sebagai target baru
```

    - Reset hitungan kehilangan target
  Hadap ke arah musuh
  Tembak dengan kekuatan 3

Ketika bot menabrak dinding:
  - Cetak pesan "Ouch! I hit a wall, must turn back!"
  - Hindari tembok dengan cara yang sesuai

Fungsi HadapTarget(x, y):
  - Hitung sudut menuju target
  - Putar bot ke arah target

Fungsi JauhinTembok():
  Jika bot terlalu dekat dengan sudut arena:
    - Sesuaikan putaran agar menghindari sudut
  Jika bot terlalu dekat dengan tepi:
    - Sesuaikan putaran agar kembali ke tengah arena

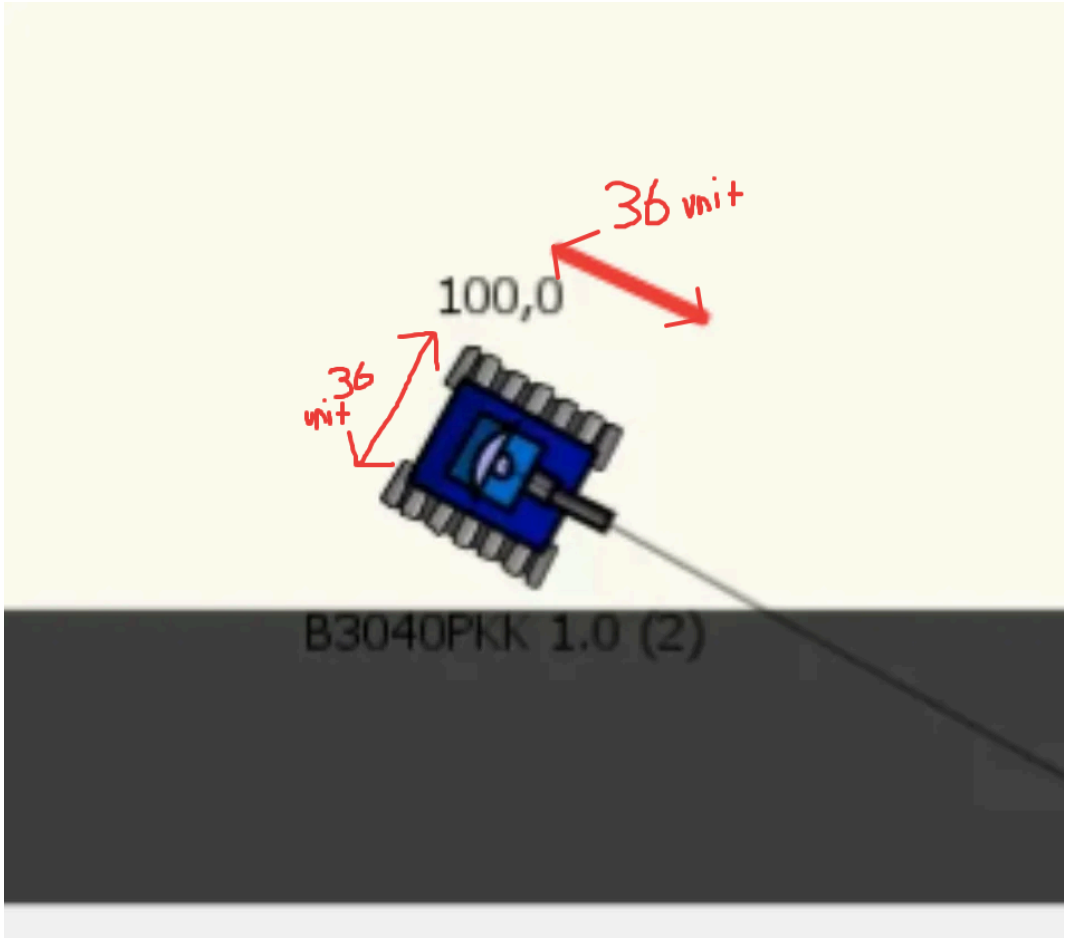
#### 4.1.2 Penjelasan Struktur Data, Cara Kerja Robot, Fungsi, dan Prosedur

*Gambar 11. Visualisasi Kelas Inheritance dari Robocode Tank Royale*

public class **Robot**
extends _Robot
implements IInteractiveRobot, IPaintRobot, IBasicEvents3, IInteractiveEvents, IPaintEvents

| Variabel | Type | Keterangan |
|---|---|---|
| *turnDirection* | *int* | Mengubah arah gerak |
| *targetLocked* | *bool* | Status sudah mengunci musuh atau belum |
| *lockedTargetId* | *int* | ID musuh yang ditargetkan |
| *lostTargetCount* | *int* | Jumlah siklus ID musuh yang ditarget tidak terdeteksi |

Program B3040PKK pertama diawali dengan memanggil fungsi static void main (args[]) yang akan memanggil fungsi start yang kemudian akan memulai pergerakan dari robot.Dalam metode Run(), robot mengatur warna tubuh, senjata, dan radar, mengecek terlebih dahulu apakah robot berenergi kurang dari 5, bila iya akan memasuki mode putar untuk menggocek musuh, bila energy masi di atas 5, dia akan memanggil perintah SetTurnRadarRight(Double.PositiveInfinity) yang menandakan bahwa radar akan diputar secara positive infinity yang menandakan radar akan berputar selamanya, SetTurnRadarRight yang menandakan digeser ke kanan selamanya, dan Keunikan dari Set dalam Robocode adalah kemampuannya untuk menjalankan beberapa perintah secara bersamaan. Jadi, robot bisa memutar badan, menggerakkan roda, dan menyesuaikan radar sekaligus. Set ini sangatlah penting untuk mendukung locking dari radar kita. Setelah itu dia akan memanggil JauhinTembok() untuk ngecek apakah dia sekarang berada di dekat tembok atau bukan. Lalu apabila radar berhasil mendapatkan musuh maka akan memanggil public override void OnScannedBot(ScannedBotEvent e), ScannedBotEvent e sendiri memiliki beberapa atribut yang dapat didapatkan public ScannedRobotEvent(String name, double energy, double bearing, double distance, double heading, double velocity), yang memiliki banyak juga method yang dapat digunakan memanfaatkan e ini, yang selengkapnya dapat dilihat pada ScannedRobotEvent (Robocode 1.9.5.0 API). Dalam method OnScannedRobot, pertama akan mengecek terlebih dahulu apakah energy kita dibawah 5, bila iya akan melakukan putaran cantik untuk menggocek musuh, bila tidak akan mengecek apakah kita sudah memiliki targetlockedid atau belum (-1 bila belom ada, lainnya bila ada), atau bila losttargetcount sudah melebihi maxlosttargetcount yaitu 3 yang artinya tidak boleh kehilangan musuh lebih dari 3 turn, bila sudah memiliki targetlockedid atau losttargetcount kurang dari 3 maka mulai masuk ke dalam algoritma untuk melakukan locking terhadap lawan. Pertama - tama, saya akan menjelaskan tentang dimensi tank

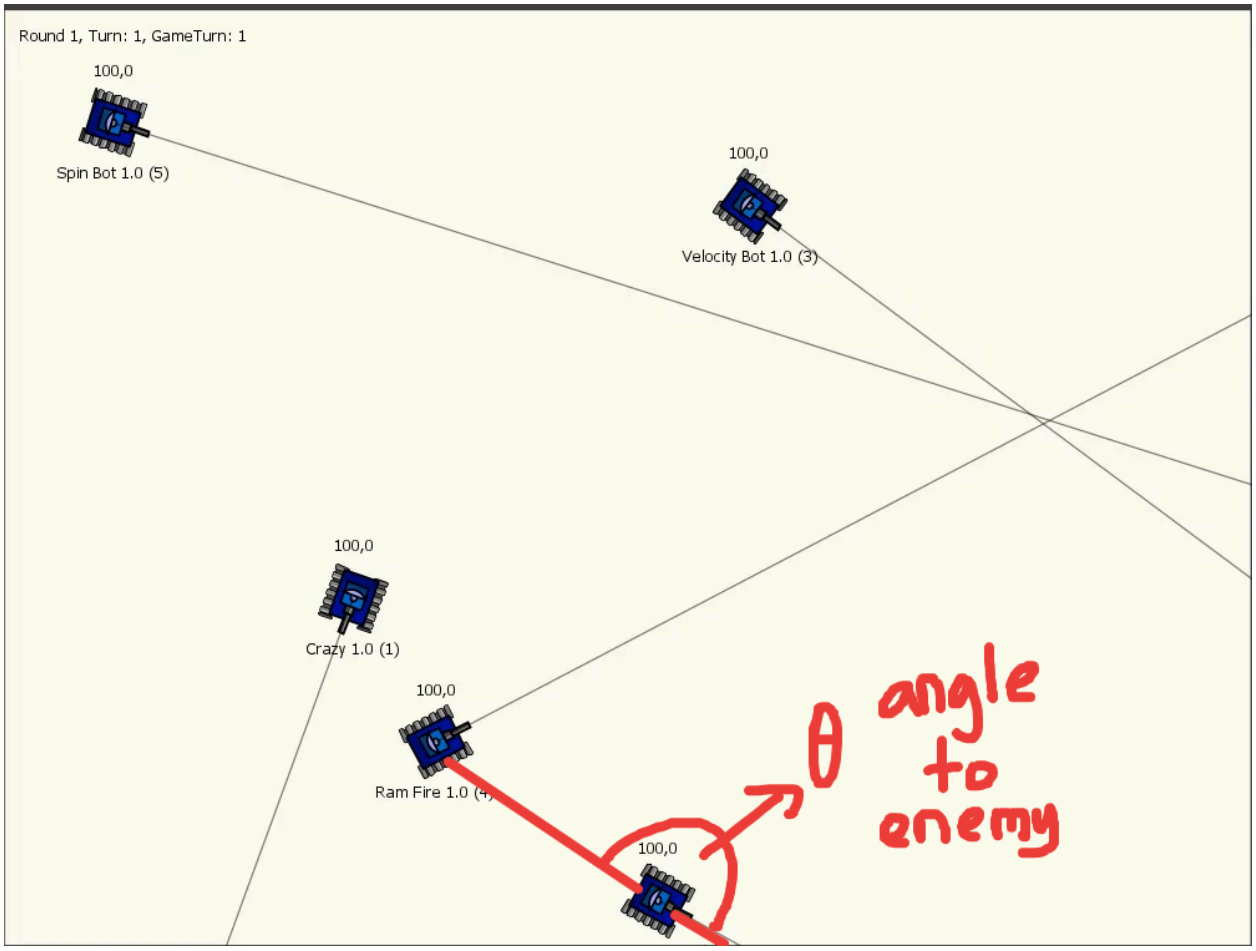



*Gambar 12. Ukuran Bot dalam Unit di Robocode Tank Royale*

Dalam Robocode Tank Royale, setiap bot adalah objek berbentuk kotak berukuran 36x36 unit dalam game. Selain ukuran, kita juga memiliki berbagai informasi penting tentang bot, seperti:

- Posisi (X, Y) → Menentukan lokasi bot dalam arena.
- Arah (Direction) → Mengarah kemana bot bergerak, dinyatakan dalam derajat (0° = kanan, 90° = atas, 180° = kiri, 270° = bawah).
- Kecepatan (Speed) → Berapa cepat bot bergerak, dengan nilai maksimum 8 unit per tick. Karena bot memiliki ukuran tetap, pergerakan dan strategi sangat dipengaruhi oleh ruang yang tersedia di arena.

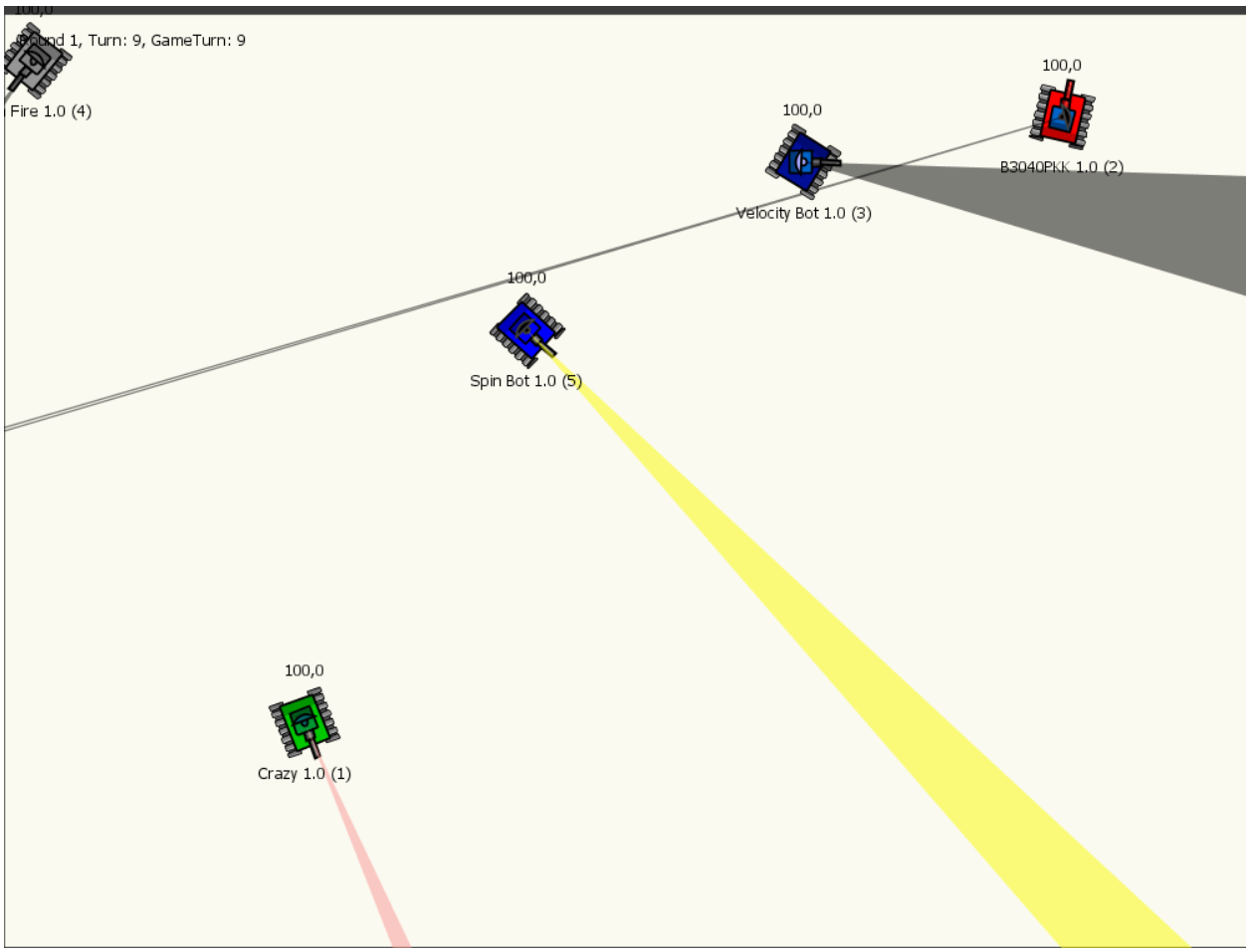
Dengan begitu, kita dapat melakukan perhitungan untuk mendapatkan berapa derajat yang diperlukan untuk menghadap ke musuh




*Gambar 13. Visualisasi Angle To Enemy*

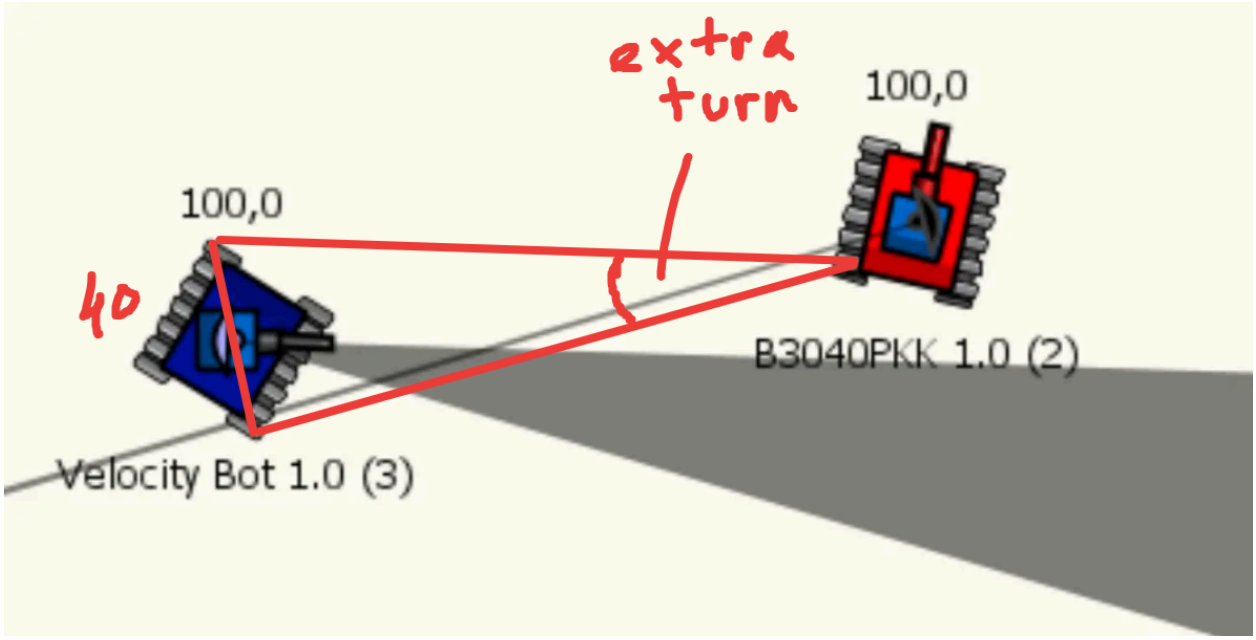
Untuk menghitung angle to enemy dari posisi sekarang ini diperlukan bearing dan direction. Bearing adalah sudut relatif posisi musuh terhadap arah bot saat ini, bukan terhadap arena. Karena itu, untuk mendapatkan sudut absolut musuh dalam sistem koordinat arena, kita perlu menambahkan Direction (arah bot dalam arena) dengan Bearing. Dengan kata lain, Angle to Enemy dihitung sebagai Direction + Bearing agar kita bisa mengetahui posisi musuh dalam referensi global. Jika tidak ditambahkan dengan Direction, kita hanya tahu musuh berada di kanan atau kiri bot, tetapi tidak tahu lokasi pastinya dalam arena.Setelah mendapatkan angleToEnemy (sudut absolut ke musuh dalam arena), langkah selanjutnya adalah menghitung berapa derajat radar harus diputar agar tetap mengunci target. RadarDirection adalah sudut absolut radar saat ini dalam arena, mirip dengan Direction untuk bot. Jika RadarDirection sudah mengarah ke angleToEnemy, berarti radar sudah tepat mengunci musuh. Tapi jika tidak, kita harus menyesuaikannya. Kita ingin tahu seberapa jauh radar harus diputar agar mengarah tepat ke musuh. Jika angleToEnemy lebih besar dari RadarDirection, radar harus diputar ke kanan. Jika angleToEnemy lebih kecil dari RadarDirection, radar harus diputar ke kiri. Jika hasilnya 0, berarti radar sudah tepat mengarah ke musuh. Dengan menghitung angleToEnemy -

RadarDirection, kita mendapatkan beda sudut antara radar dan musuh. Jangan lupa untuk normalizerelativeangle dari kedua tersebut, Masalahnya, dalam sistem sudut 360°, nilai yang kita hitung bisa menjadi lebih dari 360° atau negatif. Fungsi NormalizeRelativeAngle() memastikan nilai sudut tetap dalam rentang -180° hingga 180°, sehingga radar mengambil jalur terpendek untuk mengarah ke musuh.




*Gambar 14. Visualisasi Radar yang udah menemukan musuh*

Berikutnya menghitung radar turn, berfungsi untuk menambahkan sedikit putaran ekstra pada radar agar tetap mengunci musuh dengan lebih baik. Dapat kita lihat pada gambar 10, scan sudah berhasil menemukan keberadaaan musuh, namun bila kita lihat radar sekarang cukup miring tidak tepat di tengah musuh, disinilah peran dari extraturn untuk menjadi pelengkap dalam penghitungan rumus,




*Gambar 15. Visualisasi Extra Turn Target kepada musuh*

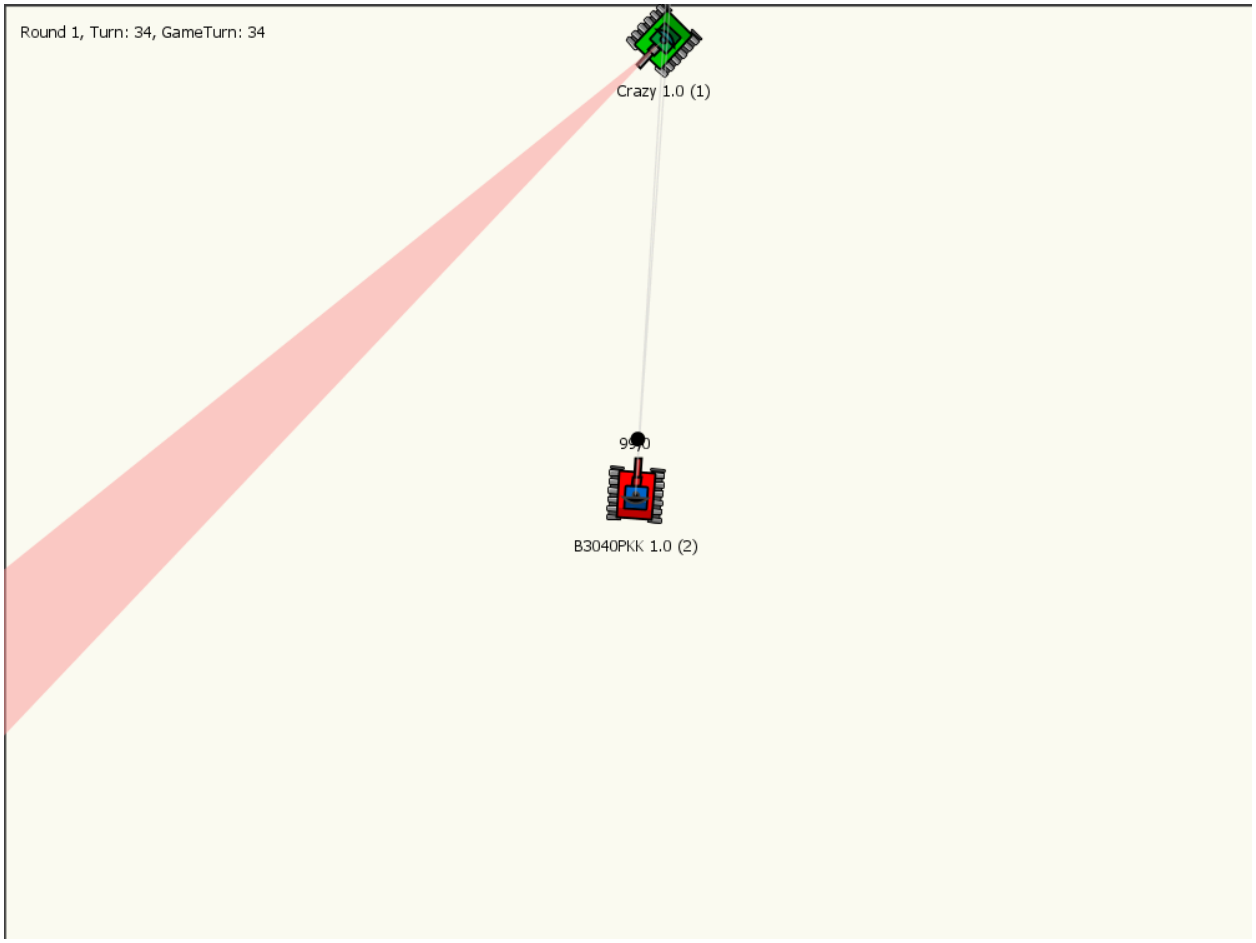
Dengan pengetahuan matematika kita, untuk mendapatkan derajat dari extraTurn, kita harus memanfaatkan tangen (tan) karena kita berurusan dengan sudut dalam segitiga siku-siku.
Dalam konteks ini:
40.0 adalah lebar bot, yang bisa dianggap sebagai panjang sisi depan (opposite) dalam segitiga siku-siku. DistanceTo(e.X, e.Y) adalah jarak ke musuh, yang bisa dianggap sebagai panjang sisi samping (adjacent). Sudut yang dicari adalah sudut kecil di antara garis radar dan garis lurus ke musuh.
Dari trigonometri, kita tahu bahwa:

$$\tan(\theta) = \frac{\text{opposite}}{\text{adjacent}}$$

Maka, untuk mendapatkan θ (extraTurn dalam radian), kita gunakan arcus tangen (atan).
JANGAN LUPA karena hasil dalam aTan adalah radian sedangkan kita menggunakan derajat pada Robocode Tank Royale sehingga kita harus kalikan dengan 180/ PI untuk mengubahnya menjadi derajat.

Setelah menjalankan fungsi tersebut, robot akan melakukan adjust radarTurn untuk menambah keakuratan scan dengan memanggil SetTurnRadarLeft sebagai counter dari setTurnTadarRight yang berjalan selamanya, setelah itu robot baru akan memanggil method HadapTarget yang

berfungsi untuk menghadap ke lawan yang sedang kita lock tadi dengan method yang disediakan oleh robocode yaitu BearingTo(e.X,e.Y).




*Gambar 16. Visualisasi Hadap Target kepada musuh*

Lalu, karena robot ingin mendapatkan banyak poin dengan memberikan damage yang sebesar - besarnya, maka kita akan melakukan ramming dan juga tembakan untuk memaksimalkan poin ke lawan, oleh karena itu kita membutuhkan fungsi untuk forwarding, namun PENTING untuk mengingat bahwa kita harus membuat fungsi maju kita dengan "SET" karena bila tidak, akan bergerak seperti FireRam yang tidak dapat meng adjust pengetahuannya terhadap musuh apabila musuhnya bergerak, dengan menggunakan Set, robot kita dapat mengupdate informasinya tentang bot setiap turn sehingga dia dapat manuver untuk mengikuti arah musuhnya bergerak. Kita juga akan melakukan penembakkan sambil menuju ke lokasi musuh. PENTING JUGA untuk mengetahui bahwa "MUSUH tidak akan dapat jalan" apabila kita sudah ram dia, maka ramming sangatlah penting untuk counter musuh yang memiliki greedy yang memanfaatkan gerakannya untuk memaksimalkan poin. Bot juga akan melakukan penyesuaian fire powernya

sesuai dengan jaraknya. Bila jarak < 100 unit maka bot akan menembak dengan power 3, bila jarak < 300 menembak dengan power 1.5 , bila lebih dari 300 akan menembak dengan power 1. Apabila pada awal lockedtargetid tidak -1 atau losttargetcount >= 3 maka bot menambah losttargetcount akan bertambah dan bot akan melakukan setradarturnright positive infinity untuk scan secara keseluruhan mencari masi ada atau tidak.

Setelah itu terdapat method - method logic lainnya juga seperti public override void OnHitBot(HitBotEvent e), untuk method HitBotEvent e sendiri memiliki beberapa perbedaan atribut yang dapat diambil, seperti:

```
public HitRobotEvent(String name,
                     double bearing,
                     double energy,
                     boolean atFault)
```

Namun yang kita akan gunakan disini adalah method yang terdapat pada fungsi bot yaitu e.IsRammed yang berfungsi untuk mengembalikan boolean jika bot berhasil menabrak musuh (e.IsRammed == true), maka kunci target tersebut. Bila tidak maka dia akan hadap saja ke target dan melanjutkan penembakkan.

Terdapat logic untuk menghindari wall juga yang ada pada public override void OnHitWall(HitWallEvent e), berfungsi apabila bot kita tertabrak kepada wall, akan memanggil jauhitembok apabila terkena, fungsi jauhitembok namun juga sudah berjalan semenjak program jalan.

Method JauhinTembok() merupakan mekanisme pertahanan untuk menghindari bot menabrak tembok arena dan memastikan pergerakannya tetap optimal. Dengan mendeteksi posisi bot terhadap batas arena, method ini mengatur sudut belokan agar bot dapat menjauh dari dinding tanpa kehilangan terlalu banyak mobilitas. Saat bot berada di pojok arena, metode ini menginstruksikan bot untuk berbelok dengan sudut besar, seperti 135 derajat, guna keluar dengan cepat dari posisi yang bisa membatasi pergerakannya. Sebagai contoh, jika bot berada di sudut kiri bawah, ia akan berbelok 135 derajat ke kanan agar segera menjauhi tembok. Jika bot hanya berada di dekat salah satu sisi arena tetapi tidak di pojok, maka sudut belokan yang digunakan lebih kecil, yaitu 90 derajat, agar bot dapat kembali ke area tengah tanpa membuang terlalu banyak waktu dan ruang. Jika bot terlalu dekat dengan sisi atas atau bawah, maka ia akan berbalik arah dengan sudut 180 derajat untuk segera menjauhi batas tersebut. Dengan menerapkan strategi ini, bot dapat bergerak secara lebih efisien, menghindari potensi kebuntuan akibat menabrak tembok, dan tetap berada dalam posisi optimal untuk menyerang serta bermanuver di sekitar lawan. Hal ini sangat penting untuk memastikan bot tetap memiliki kontrol atas pergerakannya di medan pertempuran dan tidak terjebak di area yang dapat membatasi serangannya.

## 4.2 Bot D1753ALG

### 4.2.1 Implementasi

Notasi Pseudocode

```
PROGRAM D1753ALG

VAR
   xEnemy, yEnemy, distanceToEnemy, directionEnemy, speedEnemy, bulletSpeed : DOUBLE
<- 0
   shotsFired, shotsMissed : FLOAT <- 0
   turnDirection, collisionCounter : INTEGER <- 1, 0
   stuck : BOOLEAN <- FALSE
   random : OBJECT <- Random()


   CALL StartBot()


PROCEDURE StartBot()

   SET AdjustGunForBodyTurn <- TRUE
   SET AdjustRadarForBodyTurn <- TRUE
   SET AdjustRadarForGunTurn <- TRUE
   SET stuck <- FALSE

   SET TurretColor <- PURPLE
   SET RadarColor <- YELLOW
   SET BulletColor <- ORANGE
   SET ScanColor <- YELLOW
   SET TracksColor <- PURPLE
   SET GunColor <- PURPLE
   SET BodyColor <- YELLOW

   WHILE IsRunning DO
      SET TurnRadarRight(360 * turnDirection)
      CALL Go()

      IF distanceToEnemy < 200 OR stuck THEN
```

```
        CALL StrafeMove()
      ELSE
        SetForward(100)
       IF
    WHILE


PROCEDURE OnScannedBot(e)

   SET xEnemy <- e.X
   SET yEnemy <- e.Y
   SET distanceToEnemy <- DistanceTo(xEnemy, yEnemy)
   SET directionEnemy <- e.Direction
   SET speedEnemy <- e.Speed

   CALL LockRadarOnTarget(xEnemy, yEnemy)
   CALL SmartShot(xEnemy, yEnemy, speedEnemy, directionEnemy, distanceToEnemy)

   IF distanceToEnemy < 200 OR stuck THEN
      SetTurnLeft(BearingTo(xEnemy, yEnemy) + 90)
   ELSE
      SetTurnLeft(BearingTo(xEnemy, yEnemy))
    IF


PROCEDURE OnHitBot(e)

   SET collisionCounter <- collisionCounter + 1
   IF collisionCounter > 4 THEN
      PRINT "Stuck ramming an enemy! Escaping."
      SET stuck <- TRUE
      SET collisionCounter <- 0
    IF


PROCEDURE LockRadarOnTarget(x, y)

   angleToEnemy <- Direction + BearingTo(x, y)
   radarTurn <- NormalizeRelativeAngle(angleToEnemy - RadarDirection)
   extraTurn <- MIN(ATAN(36.0 / DistanceTo(x, y)), 45)
```

```
    radarTurn <- radarTurn + (IF radarTurn < 0 THEN -extraTurn ELSE extraTurn)
    SetTurnRadarLeft(radarTurn)


PROCEDURE SmartShot(x, y, speed, direction, distance)

    bullet <- CalcBullet(distance)
    bulletSpeed <- CalcBulletSpeed(bullet)

    enemyAngleRad <- direction * PI / 180
    enemyVx <- speed * COS(enemyAngleRad)
    enemyVy <- speed * SIN(enemyAngleRad)

    dx <- x - X
    dy <- y - Y

    a <- enemyVx^2 + enemyVy^2 - bulletSpeed^2
    b <- 2 * (enemyVx * dx + enemyVy * dy)
    c <- dx^2 + dy^2

    d <- b^2 - 4 * a * c
    IF d < 0 THEN RETURN

    t1 <- (-b + SQRT(d)) / (2 * a)
    t2 <- (-b - SQRT(d)) / (2 * a)

    timeToImpact <- IF (t1 > 0 AND t2 > 0) THEN MIN(t1, t2) ELSE MAX(t1, t2)
    IF timeToImpact < 0 THEN RETURN

    enemyX <- x + enemyVx * timeToImpact
    enemyY <- y + enemyVy * timeToImpact

    enemyX <- LIMIT(enemyX, 0, 800)
    enemyY <- LIMIT(enemyY, 0, 600)

    CALL TurnGunToFaceTarget(enemyX, enemyY)
    Fire(bullet)
    shotsFired <- shotsFired + 1
```

```
PROCEDURE OnBulletHitWall(e)

   shotsMissed <- shotsMissed + 1


PROCEDURE OnRoundEnded(e)

   shotPercentage <- 100 * (shotsFired - shotsMissed) / shotsFired
   PRINT "I hit " + shotPercentage + " of my shots"


PROCEDURE OnTick(e)

   IF e.TurnNumber % 10 = 0 THEN
      SET stuck <- FALSE
    IF


PROCEDURE StrafeMove()

   Forward(100 + 100 * random.NextDouble())
   Back(100 + 100 * random.NextDouble())


PROCEDURE OnWonRound(e)

   TurnLeft(36_000)
```

### 4.2.2 Penjelasan Struktur Data, Cara Kerja Robot, Fungsi, dan Prosedur

**PENJELASAN TENTANG RADAR dijelaskan hanya 1x karena menggunakan logic scanning yang sama.**

Program D1753ALG diawali dengan pemanggilan fungsi `static void Main(string[] args)`, yang bertugas untuk menjalankan bot dengan memanggil `new D1753ALG().Start()`. Ketika bot mulai beroperasi, ia memasuki metode `Run()`, di mana berbagai pengaturan dasar dilakukan, termasuk pengaturan warna tubuh, radar, dan senjata untuk memberikan identitas visual pada bot saat

bertarung. Pada tahap awal, bot mengaktifkan fitur AdjustGunForBodyTurn, AdjustRadarForBodyTurn, dan AdjustRadarForGunTurn, yang memungkinkan bagian tubuh, senjata, dan radar untuk beroperasi secara independen satu sama lain, sehingga bot dapat bergerak bebas tanpa mempengaruhi arah senjatanya. Setelah itu, bot akan terus memutar radar secara penuh dengan SetTurnRadarRight(360 * turnDirection), yang berfungsi untuk memastikan bahwa bot dapat mendeteksi keberadaan musuh di sekelilingnya secara maksimal. Dalam mode bertahan, bot akan menggunakan strategi strafe movement, yaitu bergerak maju dan mundur secara acak untuk menghindari serangan musuh, sedangkan dalam mode agresif, bot akan langsung maju mendekati lawan dengan `SetForward(100)`.

Ketika musuh terdeteksi melalui event `OnScannedBot(ScannedBotEvent e)`, bot akan memperbarui informasi mengenai posisi dan kecepatan musuh. Menggunakan rumus bearing to enemy, bot menghitung sudut yang diperlukan agar dapat mengunci target dengan lebih akurat sama seperti robot 1 (B3040PKK). Jika musuh berada dalam jarak dekat (< 100 unit), bot akan berbelok ke arah yang tegak lurus terhadap musuh untuk menghindari tembakan langsung. Sebaliknya, jika musuh berada dalam jarak yang lebih jauh, bot akan langsung menghadapkan tubuhnya ke musuh untuk memaksimalkan peluang serangan. Untuk memastikan radar tetap terkunci pada musuh, bot menggunakan metode `LockRadarOnTarget()`, di mana radar akan diputar sedemikian rupa agar tetap fokus pada pergerakan lawan. Teknik ini sangat penting dalam Robocode Tank Royale, karena radar yang tidak terkunci akan menyebabkan bot kehilangan jejak musuh dalam pertempuran. Setelah radar berhasil mengunci target, bot akan menghitung prediksi titik tumbukan peluru dengan mempertimbangkan kecepatan dan arah pergerakan musuh menggunakan metode `SmartShot()`. Dalam metode ini, bot menggunakan rumus persamaan kuadratik yang dapat dilihat pada gambar 7 dan 8 untuk menentukan titik tembak terbaik berdasarkan kecepatan proyektil dan kecepatan musuh. Jika solusi dari persamaan memiliki nilai negatif atau tidak valid, bot tidak akan menembak untuk menghindari pemborosan peluru. Namun, jika perhitungan menunjukkan waktu tumbukan yang akurat, bot akan menyesuaikan sudut tembakan dengan `TurnGunToFaceTarget()`, lalu menembakkan peluru dengan kekuatan yang disesuaikan berdasarkan jarak musuh menggunakan metode `CalcBullet()`.

Dalam pertempuran, bot juga memiliki mekanisme untuk mendeteksi apakah ia mengalami kesulitan dalam bergerak, terutama saat menabrak dinding atau musuh. Jika bot berulang kali bertabrakan dengan lawan lebih dari empat kali, maka ia akan menandai dirinya sebagai "stuck" dan secara otomatis mengubah pola gerakannya untuk keluar dari situasi tersebut. Deteksi ini dilakukan dalam metode `OnHitBot(HitBotEvent e)`, di mana bot akan meningkatkan nilai `collisionCounter`, dan jika melebihi ambang batas tertentu, bot akan berusaha untuk melepaskan diri dari tabrakan dengan memodifikasi arah gerakannya. Selain itu, bot juga memiliki mekanisme penghindaran tembok melalui metode `OnHitWall(HitWallEvent e)`, yang memastikan bahwa bot tidak terjebak di pinggir arena. Saat bot terlalu dekat dengan batas arena,

ia akan memutar tubuhnya dengan sudut yang cukup besar untuk menjauh dari tembok, sehingga pergerakannya tetap lancar.

Selain bertahan dan menyerang, bot juga mencatat statistik mengenai akurasi tembakannya. Setiap kali bot menembak, jumlah tembakan yang dilepaskan (`shotsFired`) akan bertambah, dan jika peluru meleset serta mengenai dinding, jumlah tembakan yang gagal (`shotsMissed`) juga akan dicatat melalui event `OnBulletHitWall(BulletHitWallEvent e)`. Setelah ronde selesai, bot akan menghitung persentase akurasi tembakan dengan rumus `(shotsFired - shotsMissed) / shotsFired * 100`, yang kemudian ditampilkan dalam konsol melalui event `OnRoundEnded(RoundEndedEvent e)`. Dengan demikian, bot dapat mengevaluasi performanya dalam pertempuran dan menentukan apakah strategi yang digunakan cukup efektif atau masih perlu perbaikan. Sebagai tambahan, jika bot berhasil memenangkan ronde, ia akan memutar tubuhnya sebanyak `36000` derajat sebagai bentuk selebrasi dalam metode `OnWonRound(WonRoundEvent e)`.

Secara keseluruhan, D1753ALG merupakan bot yang menggabungkan strategi bertahan, menyerang, dan penguncian target dengan radar yang canggih. Dengan memanfaatkan perhitungan matematis untuk menyesuaikan tembakan, mekanisme penghindaran tembok, serta deteksi kondisi "stuck", bot ini mampu beradaptasi dalam berbagai situasi pertempuran. Sistem pencatatan statistiknya juga memungkinkan bot untuk menganalisis dan meningkatkan akurasi tembakannya di setiap pertandingan, menjadikannya lawan yang cukup tangguh dalam Robocode Tank Royale.

*Gambar 17. Visualisasi Kelas Inheritance dari Robocode Tank Royale*

### 4.3 Bot F3812FIO

*Gambar 18. Visualisasi Kelas Inheritance dari Robocode Tank Royale*

### 4.3.1 Implementasi

Pseudocode Program F3812FIO

```
PROGRAM F3812FIO

VAR enemyPosition : Point2D <- new Point2D(300, 400) moveLength : DOUBLE <- 0 enemyDistance :
DOUBLE idealDistance : DOUBLE <- 200 targetLocked : BOOLEAN <- false lockedTargetId :
INTEGER <- -1 lostTargetCount : INTEGER <- 0 patternLength : INTEGER <- 20 maxLostTargetCount
: INTEGER <- 3 moveIncrement : INTEGER <- 1 rand : Random <- new Random() p : INTEGER <- 0
frame : INTEGER <- 15 enemyPatterns : LIST OF LIST OF EnemyMovement lastPattern : LIST OF
EnemyMovement <- new List() enemyHistory : LIST OF EnemyMovement <- new List()
lastEnemyDirection : DOUBLE <- 0

FUNCTION Main(args : STRING[])    new F3812FIO().Start()

CONSTRUCTOR F3812FIO()    CALL base(BotInfo.FromFile("F3812FIO.json"))

PROCEDURE Run()    SET BodyColor <- Color.Gray SET AdjustGunForBodyTurn <- true SET
AdjustRadarForBodyTurn <- true SET AdjustRadarForGunTurn <- true SET MaxRadarTurnRate <- 45

FOR i <- 0 TO 9
   ADD new List<EnemyMovement>() TO enemyPatterns
 FOR

WHILE IsRunning DO
   IF NOT targetLocked THEN
      SetTurnRadarRight(360)
      SetTurnRight(45)
    IF

   CALL GoToEnemy()
   CALL Go()
 WHILE



PROCEDURE OnScannedBot(e : ScannedBotEvent)    IF lockedTargetId = -1 OR lockedTargetId =
e.ScannedBotId THEN SET lockedTargetId <- e.ScannedBotId

   SET headingChange <- NormalizeRelativeAngle(e.Direction - lastEnemyDirection)
   SET lastEnemyDirection <- e.Direction
```

```
ADD new EnemyMovement(e.Speed, headingChange, e.Direction, new Point2D(e.X, e.Y)) TO
enemyPatterns[lockedTargetId]

IF enemyPatterns[lockedTargetId].Count > 1000 THEN
    REMOVE first element FROM enemyPatterns[lockedTargetId]
 IF

ADD new EnemyMovement(e.Speed, headingChange, e.Direction, new Point2D(e.X, e.Y)) TO
lastPattern

IF lastPattern.Count > patternLength OR lostTargetCount >= maxLostTargetCount THEN
    SET lastPattern <- new List<EnemyMovement>()
 IF

SET lostTargetCount <- 0

SET radarTurn <- NormalizeRelativeAngle(RadarBearingTo(e.X, e.Y))
SET extraTurn <- MIN(ATAN(36.0 / DistanceTo(e.X, e.Y)), 45)

IF radarTurn < 0 THEN
    SET radarTurn <- radarTurn - extraTurn
ELSE
    SET radarTurn <- radarTurn + extraTurn
 IF

SetTurnRadarLeft(radarTurn)

SET enemyPosition <- new Point2D(e.X, e.Y)
SET enemyDistance <- DistanceTo(e.X, e.Y)

IF enemyDistance < 30 THEN
    SET frame <- 11
ELSE IF enemyDistance < 120 THEN
    SET frame <- 15
ELSE IF enemyDistance < 200 THEN
    SET frame <- 22
ELSE
    SET frame <- 40
 IF

PRINT enemyHistory.Count

IF enemyPatterns[lockedTargetId].Count > 70 AND lastPattern.Count = patternLength THEN
    SET matchIndex <- FindBestMatch(lastPattern)
```

```
        SET predictedPos <- PredictFuturePosition(matchIndex, new Point2D(e.X, e.Y), e.Direction)

        CALL gunToEnemy(predictedPos.X, predictedPos.Y)
        CALL GoToEnemy()

        IF enemyDistance < 50 THEN
            SetFire(3)
        ELSE IF enemyDistance < 100 THEN
            SetFire(2.3)
        ELSE
            SetFire(1.5)
         IF

        CALL Go()

        PRINT "Predicted"
        PRINT "Enemy Position: " + e.X + " " + e.Y
        PRINT "Predicted Position: " + predictedPos.X + " " + predictedPos.Y
    ELSE IF lastPattern.Count = patternLength THEN
        CALL GoToEnemy()
        SetFire(1)
        CALL Go()
        PRINT "Normal"
     IF
ELSE
    SET lostTargetCount <- lostTargetCount + 1

    IF lostTargetCount >= maxLostTargetCount THEN
        SET lockedTargetId <- -1
        SetTurnRadarRight(Double.PositiveInfinity)
     IF
 IF



PROCEDURE gunToEnemy(x : DOUBLE, y : DOUBLE)    SET gunTurn <-
NormalizeRelativeAngle(GunBearingTo(x, y)) SetTurnGunLeft(gunTurn)

WHILE GunTurnRemaining > 0 DO
    CALL Go()
 WHILE
```

```
FUNCTION FindBestMatch(pattern : LIST OF EnemyMovement) : INTEGER    SET bestMatchIndex <-
-1 SET bestScore <- Double.MaxValue

SET begin <- 0
SET last <- 0

IF enemyPatterns[lockedTargetId].Count < 1000 THEN
   SET last <- enemyPatterns[lockedTargetId].Count - lastPattern.Count * 2
   SET begin <- 0
ELSE
   SET last <- enemyPatterns[lockedTargetId].Count - lastPattern.Count
   SET begin <- patternLength
 IF

FOR i <- begin TO last - 1 DO
   SET score <- 0

   FOR j <- 0 TO pattern.Count - 1 DO
      SET velocityDiff <- ABS(enemyPatterns[lockedTargetId][i + j].Velocity - lastPattern[j].Velocity)
      SET headingChangeDiff <- ABS(enemyPatterns[lockedTargetId][i + j].HeadingChange -
lastPattern[j].HeadingChange)
      SET headingDiff <- ABS(enemyPatterns[lockedTargetId][i + j].HeadingDriection -
lastPattern[j].HeadingDriection)

      SET score <- score + velocityDiff + headingDiff + headingDiff
    FOR

   IF score < bestScore THEN
      SET bestScore <- score
      SET bestMatchIndex <- i
    IF
 FOR

RETURN bestMatchIndex



FUNCTION PredictFuturePosition(matchIndex : INTEGER, currentPos : Point2D, currentHeading :
DOUBLE) : Point2D    IF matchIndex = -1 THEN RETURN new Point2D(300, 400)  IF

SET predictedPos <- new Point2D(currentPos.X, currentPos.Y)
SET predictedHeading <- currentHeading

SET i <- matchIndex + patternLength
```

```
SET velocity <- enemyPatterns[lockedTargetId][i].Velocity
SET headingChange <- enemyPatterns[lockedTargetId][i].HeadingChange

IF matchIndex + patternLength + frame >= enemyPatterns[lockedTargetId].Count THEN
   SET p <- matchIndex + patternLength + frame - enemyPatterns[lockedTargetId].Count
ELSE
   SET p <- matchIndex + patternLength + frame
 IF

PRINT p + " frame"

SET displacement <- enemyPatterns[lockedTargetId][p].Position -
enemyPatterns[lockedTargetId][matchIndex + patternLength].Position
SET predictedPos <- enemyPosition + displacement

RETURN predictedPos



PROCEDURE GoToPosition(x : DOUBLE, y : DOUBLE)    SET bodyTurn <-
NormalizeRelativeAngle(BearingTo(x, y)) SET distance <- DistanceTo(x, y)

SET gunTurn <- NormalizeRelativeAngle(GunBearingTo(x, y))
SetTurnGunLeft(gunTurn)
SetTurnLeft((bodyTurn - 45) + (45 * new Random().Next(2)))
SetForward((distance - 100))



PROCEDURE GoToEnemy()    CALL GoToPosition(enemyPosition.X, enemyPosition.Y)

PROCEDURE OnHitBot(e : HitBotEvent)    SetBack(50)

PROCEDURE OnHitWall(e : HitWallEvent)    // Empty implementation

PROCEDURE OnHitByBullet(e : HitByBulletEvent)    // Empty implementation

CLASS EnemyMovement VAR Velocity : DOUBLE HeadingChange : DOUBLE HeadingDriection :
DOUBLE Position : Point2D

CONSTRUCTOR EnemyMovement(velocity : DOUBLE, headingChange : DOUBLE, heading :
DOUBLE, position : Point2D)

   SET Velocity <- velocity
```

```
    SET HeadingChange <- headingChange
    SET HeadingDriection <- heading
    SET Position <- new Point2D(position.X, position.Y)


  CLASS

CLASS Point2D VAR X : DOUBLE Y : DOUBLE

CONSTRUCTOR Point2D(x : DOUBLE, y : DOUBLE)

    SET X <- x
    SET Y <- y


OPERATOR -(p1 : Point2D, p2 : Point2D) : Point2D

    RETURN new Point2D(p1.X - p2.X, p1.Y - p2.Y)


OPERATOR +(p1 : Point2D, p2 : Point2D) : Point2D

    RETURN new Point2D(p1.X + p2.X, p1.Y + p2.Y)


  CLASS
```

### 4.3.2 Penjelasan Struktur Data, Cara Kerja Robot, Fungsi, dan Prosedur

Robot F3812FIO ini dimulai dengan memanggil fungsi Main yang akan membuat objek baru dari kelas F3812FIO dan menjalankan fungsi Start() untuk memulai robot. Di dalam konstruktor F3812FIO, robot mengambil konfigurasi dari file "F3812FIO.json". Fungsi Run() adalah fungsi utama yang mengatur semua cara kerja robot, di sini robot diatur warna tubuhnya jadi abu-abu, terus diatur juga agar gerakan senjata dan radar ikut menyesuaikan kalau badan robot berputar dengan mengaktifkan AdjustGunForBodyTurn, AdjustRadarForBodyTurn, dan AdjustRadarForGunTurn. Kecepatan putar radar maksimal diatur 45 derajat per tick. Sebelum mulai looping utama, robot membuat 10 daftar kosong untuk menyimpan pola gerakan musuh nantinya.

Di dalam loop utama yang terus berjalan selama robot hidup, robot akan terus mencari musuh. Kalau belum ada target yang dikunci (targetLocked masih false), radar akan berputar 360 derajat dan badan robot berputar 45 derajat untuk memaksimalkan pencarian. Begitu musuh terdeteksi, fungsi OnScannedBot langsung dipanggil dan menerima informasi tentang musuh seperti posisi, kecepatan, dan arah gerak musuh. Robot punya sistem penguncian target yang pintar, dia bisa tetap fokus pada target yang sudah dikunci (ditandai dengan lockedTargetId) atau mengunci target baru kalau belum ada yang dikunci. Untuk musuh yang baru terdeteksi, robot mencatat data pergerakan mereka termasuk kecepatan, perubahan arah, dan posisi, lalu menyimpan informasi ini untuk dianalisis pola geraknya nanti.

Sistem radar robot menggunakan perhitungan matematika yang jitu untuk terus mengunci musuh. Robot menghitung bearing radar ke posisi musuh dan menambahkan sudut ekstra berdasarkan jarak musuh, memastikan musuh tetap dalam jangkauan radar. Sudut ekstra ini dihitung pakai fungsi arcus tangen: Math.Atan(36.0 / DistanceTo(e.X, e.Y)), di mana 36.0 adalah lebar tank di Robocode. Hasilnya dibatasi maksimal 45 derajat biar radar tidak berputar terlalu berlebihan. Robot kemudian mengupdate posisi musuh dan menghitung jarak, yang digunakan untuk menentukan berapa banyak frame ke depan yang akan diprediksi untuk penembakan.

Sistem penembakan adalah bagian paling canggih dari robot F3812FIO. Setelah mengumpulkan cukup banyak data gerakan musuh (lebih dari 70 catatan) dan mengamati pola lengkap (yang jumlahnya 20 gerakan), robot mengaktifkan penembakan prediktif. Fungsi FindBestMatch membandingkan pola gerakan musuh terbaru dengan pola-pola yang sudah tersimpan, menghitung perbedaan kecepatan, perubahan arah, dan arah gerakan untuk menemukan pola masa lalu yang paling mirip. Fungsi PredictFuturePosition kemudian menggunakan pola yang cocok ini untuk memperkirakan di mana musuh akan berada setelah beberapa frame ke depan, jumlahnya ditentukan berdasarkan jarak ke musuh saat ini. Jarak dekat menggunakan prediksi lebih pendek (sedekat 11 frame saat sangat dekat), sedangkan jarak jauh menggunakan frame lebih banyak (hingga 40) untuk mengakomodasi waktu tempuh peluru yang lebih lama.

Fungsi gunToEnemy memutar senjata dengan tepat ke arah posisi yang diprediksi dan menunggu sampai putarannya selesai sebelum menembak. Kekuatan tembakan disesuaikan berdasarkan jarak musuh: kekuatan 3.0 untuk musuh yang jaraknya kurang dari 50 unit, 2.3 untuk jarak di bawah 100 unit, dan 1.5 untuk jarak yang lebih jauh, untuk mengoptimalkan damage sekaligus menghemat energi. Kalau data pola belum cukup tapi sudah ada pola minimal (sebanyak patternLength), robot akan menembak dengan kekuatan lebih rendah (1.0) sambil terus mengumpulkan data pola.

Sistem pergerakan robot diatur melalui beberapa fungsi yang bekerja sama. GoToPosition menghitung sudut putaran dan jarak maju yang diperlukan untuk mencapai koordinat tertentu di arena. Fungsi ini menggunakan BearingTo untuk menentukan arah ke target, kemudian menyesuaikan arah robot dengan sedikit pengacakan agar gerakannya tidak terlalu mudah

ditebak. GoToEnemy cuma memanggil GoToPosition dengan koordinat musuh saat ini. Penangan event OnHitBot dipicu ketika bertabrakan dengan robot lain, menyebabkan F3812FIO mundur 50 unit untuk membuat jarak. Penangan OnHitWall meski didefinisikan tapi tampak kosong dalam implementasi ini.

Algoritma pencocokan pola sangat canggih. FindBestMatch memeriksa bagian-bagian dari catatan gerakan musuh untuk menemukan kecocokan terdekat dengan pola saat ini. Algoritma ini menghitung skor berdasarkan perbedaan kecepatan, perubahan arah, dan arah untuk setiap kemungkinan kecocokan, memilih indeks dengan skor perbedaan total terendah. PredictFuturePosition kemudian menggunakan indeks kecocokan terbaik ini untuk menghitung di mana musuh kemungkinan besar akan berada di masa depan. Daripada hanya memproyeksikan ke depan dari posisi saat ini, algoritma ini dengan cerdik menghitung vektor perpindahan antara dua titik dalam pola historis (dipisahkan oleh nilai frame prediksi) dan menerapkan perpindahan ini ke posisi musuh saat ini. Pendekatan ini efektif memperhitungkan pola gerakan kompleks yang akan terlewat oleh prediksi linier sederhana.

Robot menggunakan dua kelas pendukung untuk mengelola data: EnemyMovement mencatat detail tentang gerakan musuh termasuk kecepatan, perubahan arah, arah, dan posisi, sementara Point2D menyediakan implementasi titik 2D kustom dengan operator yang dimodifikasi untuk penambahan dan pengurangan vektor. Kelas-kelas ini memungkinkan algoritma pelacakan dan prediksi pola canggih yang menjadi inti dari sistem penargetan robot. Melalui peningkatan berkelanjutan database polanya selama pertempuran, F3812FIO menjadi semakin akurat dalam memprediksi gerakan musuh dan mendaratkan tembakan yang berhasil, menunjukkan kemampuan belajar yang meningkatkan efektivitas tempurnya seiring waktu.

## 4.4 Jeb_

### 4.4.1 Implementasi

Pseudocode Jeb_

PROGRAM JebBot

VAR turnDirection : INTEGER <- 1 lockedTargetId : INTEGER <- -1 /* Tidak ada musuh yang dikunci / *lostTargetCount : INTEGER <- 0 /* Menghitung jumlah turn tanpa musuh / *maxLostTargetCount : CONSTANT INTEGER <- 3 /* Jika melebihi 3 turn tanpa musuh, target dianggap hilang */ enemyPosition : POINT2D <- (300, 400) enemyDistance : DOUBLE <- 0 idealDistance : DOUBLE <- 200 targetLocked : BOOLEAN <- FALSE moveIncrement : INTEGER <- 1 shotsMissed : INTEGER <- 0 shotsHit : INTEGER <- 0 stuck : BOOLEAN <- FALSE getRammed : BOOLEAN <- FALSE turnRammed : INTEGER <- 0 hue : DOUBLE <- 0

CALL StartBot() /* Memulai bot */

```
PROCEDURE StartBot()      SET BodyColor <- PINK SET GunColor <- BLUE SET RadarColor <-
PURPLE SET AdjustGunForBodyTurn <- TRUE SET AdjustRadarForBodyTurn <- TRUE SET
AdjustRadarForGunTurn <- TRUE SET MaxRadarTurnRate <- 45

WHILE IsRunning DO
   IF NOT targetLocked THEN
      SetTurnRadarRight(INFINITY) /* Radar berputar selamanya */
    IF
   CALL Go() /* Jalankan perintah lain */
 WHILE



PROCEDURE OnScannedBot(e)      IF EnemyCount > 2 THEN SET idealDistance <- 450 ELSE SET
idealDistance <- 250  IF

IF e.Energy = 0 THEN
   CALL SmartShot(e.X, e.Y, 0, 0, DistanceTo(e.X, e.Y))
 IF

IF lockedTargetId = -1 OR lockedTargetId = e.ScannedBotId THEN
   SET lockedTargetId <- e.ScannedBotId
   SET lostTargetCount <- 0
   SET enemyPosition <- (e.X, e.Y)
   SET enemyDistance <- DistanceTo(e.X, e.Y)
   CALL SmartShot(e.X, e.Y, e.Speed, e.Direction, enemyDistance)
   CALL KeepDistance()
   CALL Go()
ELSE
   INCREMENT lostTargetCount
   IF lostTargetCount >= maxLostTargetCount THEN
      SET lockedTargetId <- -1
      SetTurnRadarRight(INFINITY)
    IF
 IF



PROCEDURE KeepDistance()    IF stuck THEN RETURN  IF

IF enemyDistance > idealDistance THEN
   CALL GoToPosition(enemyPosition.X, enemyPosition.Y)
ELSE
```

```
   CALL runAway()
 IF
```

PROCEDURE GoToPosition(x, y) SET bodyTurn <- NormalizeRelativeAngle(BearingTo(x, y)) SET distance <- DistanceTo(x, y)

```
IF ABS(bodyTurn) > 90 THEN
   IF bodyTurn < 0 THEN
      SET bodyTurn <- -180 - bodyTurn
   ELSE
      SET bodyTurn <- 180 - bodyTurn
    IF
   SetTurnRight(bodyTurn)
   SetBack(distance)
ELSE
   SetTurnLeft(bodyTurn)
   SetForward(distance)
 IF
```

PROCEDURE runAway() IF RadarDirection > 0 AND RadarDirection <= 45 THEN CALL GoToPosition(500, 40) ELSE IF RadarDirection > 45 AND RadarDirection <= 90 THEN CALL GoToPosition(40, 350) ELSE IF RadarDirection > 90 AND RadarDirection <= 135 THEN CALL GoToPosition(300, 40) ELSE IF RadarDirection > 135 AND RadarDirection <= 180 THEN CALL GoToPosition(760, 350) ELSE IF RadarDirection > 180 AND RadarDirection <= 225 THEN CALL GoToPosition(760, 250) ELSE IF RadarDirection > 225 AND RadarDirection <= 270 THEN CALL GoToPosition(300, 560) ELSE IF RadarDirection > 270 AND RadarDirection <= 315 THEN CALL GoToPosition(500, 560) ELSE IF RadarDirection > 315 AND RadarDirection <= 360 THEN CALL GoToPosition(40, 250) IF

PROCEDURE SmartShot(x, y, speed, direction, distance) SET bulletPower <- 3 - (2.9 * distance / 800) IF stuck THEN SET bulletPower <- 3 ELSE IF Energy < 4 THEN SET bulletPower <- 0.3 ELSE IF Energy < 2 THEN SET bulletPower <- 0.1 IF Fire(bulletPower)

PROCEDURE OnHitBot(e) IF NOT getRammed THEN SET lockedTargetId <- e.VictimId SET enemyPosition <- (e.X, e.Y) Fire(3) INCREMENT collisionCounter IF collisionCounter > 3 THEN CALL unstuck() SET collisionCounter <- 0 IF SET getRammed <- TRUE IF

PROCEDURE unstuck() IF RadarDirection + 180 <= Direction OR Direction <= RadarDirection THEN TurnRight(180 - (RadarDirection - Direction + 360) % 360) SetForward(100) CALL

KeepDistance() ELSE TurnLeft(180 - (Direction - RadarDirection + 360) % 360) SetForward(100) CALL KeepDistance()  IF

### 4.4.2 Penjelasan Struktur Data, Cara Kerja Robot, Fungsi, dan Prosedur

*Gambar 19. Visualisasi Kelas Inheritance dari Robocode Tank Royale*

Bot Jeb_ terbentuk melalui fungsi main dan menjalankan fungsi *Start()* untuk memulai atau running bot. Pada fungsi *Run()* bot Jeb_ melakukan *adjust* pada *body, radar,* dan *turret*-nya agar masing-masing bergerak secara independen. Selain itu, fungsi *Run()* menginisiasi nilai *turnStuck* dan *turnRammed* yang nantinya akan digunakan pada *event OnTick* dan bot Jeb_ hanya memutar *Radar* ketika tidak menemukan target.

- **Variabel Global**

| Variabel | Type | Keterangan |
|---|---|---|
| *enemyDistance* | *double* | Menyimpan nilai jarak ke musuh yang discan |
| *targetLocked* | *bool* | Status sudah mengunci musuh atau belum |
| *lockedTargetId* | *int* | ID musuh yang ditargetkan |
| *lostTargetCount* | *int* | Jumlah siklus ID musuh yang ditarget tidak terdeteksi |
| *maxLostTargetCount* | *int* | Batas max kehilangan target musuh |
| *shotsMissed* | *int* | Jumlah tembakan yang tidak mengenai target |
| *shotHit* | *int* | Jumlah tembakan yang berhasil mengenai target |
| *d* | *double* | Menentukan jarak titik-titik relatif dari musuh |
| *extraD* | *double* | Mengubah variabel *d* dengan rentang nilai tertentu |
| *bulletSpeed* | *double* | Kecepatan peluru yang ditembak |
| *collisionCounter* | *int* | Jumlah tabrakan yang dialami |
| *stuck* | *bool* | Status apakah bot tidak bergerak atau normal |
| *hue* | *double* | Nilai *hue* yang digunakan pada warna bot |
| *getRammed* | *bool* | Status apakah bot baru saja ditabrak |
| *turnRammed* | *int* | Jumlah *turn* sejak bot terakhir ditabrak |
| *turnStuck* | *int* | Jumlah *turn* sejak bot terakhir mengalami *stuck* |

- ***Events***
  1. *OnScannedBot*
     Fungsi/*event* terpanggil ketika *scan* bot mendeteksi adanya bot lain. Bot akan menginisiasi atau meng-*update* variabel *lockedTargetId, enemyPosition, enemyDistance,* dan *lostTargetCount*. Fungsi ini merupakan salah satu fungsi yang terbilang hampir selalu terpanggil dan menjadi bagian paling inti dalam penentuan keputusan bot seperti menggerakan *radar* lebih lanjut untuk

*lock* bot musuh dan memanggil fungsi *SmartShoot()* serta *KeepDistance()*. Perubahan target musuh terjadi ketika *scan* tidak mendeteksi atau mendeteksi bot lain dengan Id yang berbeda pada *lockedTargetId.*

2. *OnBulletHit*
   Fungsi/*event* terpanggil ketika *bullet* yang ditembakan mengenai bot lain. Fungsi ini meng-*update shotsHit* ketika mengenai musuh sesuai *lockedTargetId*, menambah nilai *extraD* sebanyak 30 ketika *shotsHit* > 2 sekaligus mereset *shotsMissed* dan *shotsHit* menjadi 0. *extraD* yang bertambah berarti ketika akurasi sudah meningkat, bot akan menjaga jarak terhadap musuh secara bertahap.

3. *OnBulletHitWall*
   Fungsi/*event* terpanggil ketika *bullet* mengenai tembok arena. Fungsi ini menambah nilai *shotMissed* sebesar 1 dan mengurai *extraD* sebanyak 30 serta mereset *shotHit* menjadi 0 ketika jumlah *shotMissed* ≥ 2.

4. *OnHitByBullet*
   Fungsi/*event* terpanggil ketika bot terkena *bullet* bot lain. Fungsi ini hanya menambah nilai *extraD* sebesar 80 ketika *event* terjadi.

5. *OnHitBot*
   Fungsi/*event* terpanggil ketika bot menabrak atau ditabrak bot lain. Fungsi ini berperan penting untuk meminimalisir terjadinya *stuck* pada bot. Variabel yang di *update* pada fungi ini adalah *lockedTargetId, stuck, getRammed, collisionCounter,* dan *enemyPosition*. Bot akan mengubah targetnya ke arah bot yang menabraknya (termasuk radar dan *gun*) dan memasuki status *stuck* ketika *collisionCounter* > 2 dan akan memanggil fungsi *unstuck()* dengan passing parameter *bodyturn* yang nilainya telah dikalkulasikan. Ketika sebelumnya status *getRammed* adalah false, maka fungsi ini akan menjalankan bot ke arah berlawanan dari arah yang dituju untuk jarak yang singkat sebagai reflek.

6. *OnTick*
   Fungsi/*event* terpanggil setiap saat *turn* bertambah. Fungsi ini akan terus meng-*update* nilai dari variabel *d, turnStuck,* dan *turnRammed*. Selain itu, ia akan mengubah status *stuck* ketika *turnStuck* sudah melebih 20 turn dari saat pertamakali *stuck* dan mengubah status *getRammed* ketika *turn* sudah lebih dari 30 ketika pertama kali *getrammed* bernilai *true*. Terkahir yang terpenting adalah fungsi ini memanggil *letsGoJeb()* yang menjadi keunikan bot Jeb_.

7. *OnDeath*
   Fungsi/*event* terpanggil ketika bot telah hancur/mati. Fungsi mereset status *stuck* dan *getRammed* menjadi false. Ketika bot hancur dalam keadaan *stuck* atau *getRammed* bernilai *true*, maka akan terbawa ke *round* selanjutnya yang dapat menyebabkan masalah.

- **Fungsi**
  1. *void letsGoJeb()*

Fungsi yang dipanggil dari *event OnTick* yang berguna untuk merubah-ubah warna bot tiap kali fungsi ini dipanggil dengan memanfaatkan fungsi *HsvToRgb()* dan meng-*update* nilai variabel *hue* menjadi (*hue*+2)%360.

2. *void KeepDistance()*
   Fungsi yang mayoritas dipanggil dari *event OnScannedBot* yang berguna untuk menjaga jarak bot dari bot musuh yang ditargetkan atau di-*scan*. Fungsi ini menentukan 4 titik posisi relatif terhadap posisi dan sudut musuh. Titik-titik tersebut dikalkulasikan berdasarkan nilai variabel *d, enemyPoistion,* dan RadarDirection. Setelah titik-titik yang valid ditentukan, maka fungsi *GoToPosition()* akan dipanggil untuk menggerakan bot ke titik tersebut. Tetapi ketika tidak ada titik yang valid, maka fungsi *runAway*() akan dipanggil.

3. *runAway()*
   Fungsi ini digunakan untuk pergi mengarah ke arah titik kabur yang valid. Titik kabur yang dikalkulasikan pada fungsi ini ada 8 dan dipilih berdasarkan RadarDirection dari bot. Titik kabur yang dipilih akan dipassing dan digunakan pada fungsi *GoToPosition()*.

4. *void GoToPosition(double x, double y)*
   Fungsi ini digunakan untuk membuat robot bergerak suatu titik berdasarkan lokasi yang merupakan parameter ini.

5. *void unstuck(double dir)*
   Fungsi ini dipanggil dari event *OnHitBot* ketika *collisionCounter* telah lebih dari 2 kali. Parameter yang diterima merupakan data bertipe *double* sebagai representasi arah musuh menghadap. Arah tersebut digunakan untuk kabur dilanjutkan dengan memanggil fungsi *KeepDistance()*.

6. *void SmartShot(double x, double y, double speed, double direction, double distance)*
   Fungsi menerima data musuh melalui *OnScannedBot* dan menggunakan perumus pitagoras sehingga akan mengarahkan tembakan pada lokasi tebakan tersebut. Namun fungsi hanya dapat menebak secara pasti jika robot bergerak dengan kecepatan dan arah yang konstan, untuk lebih lanjut dapat melihat gambar 7 dan 8.

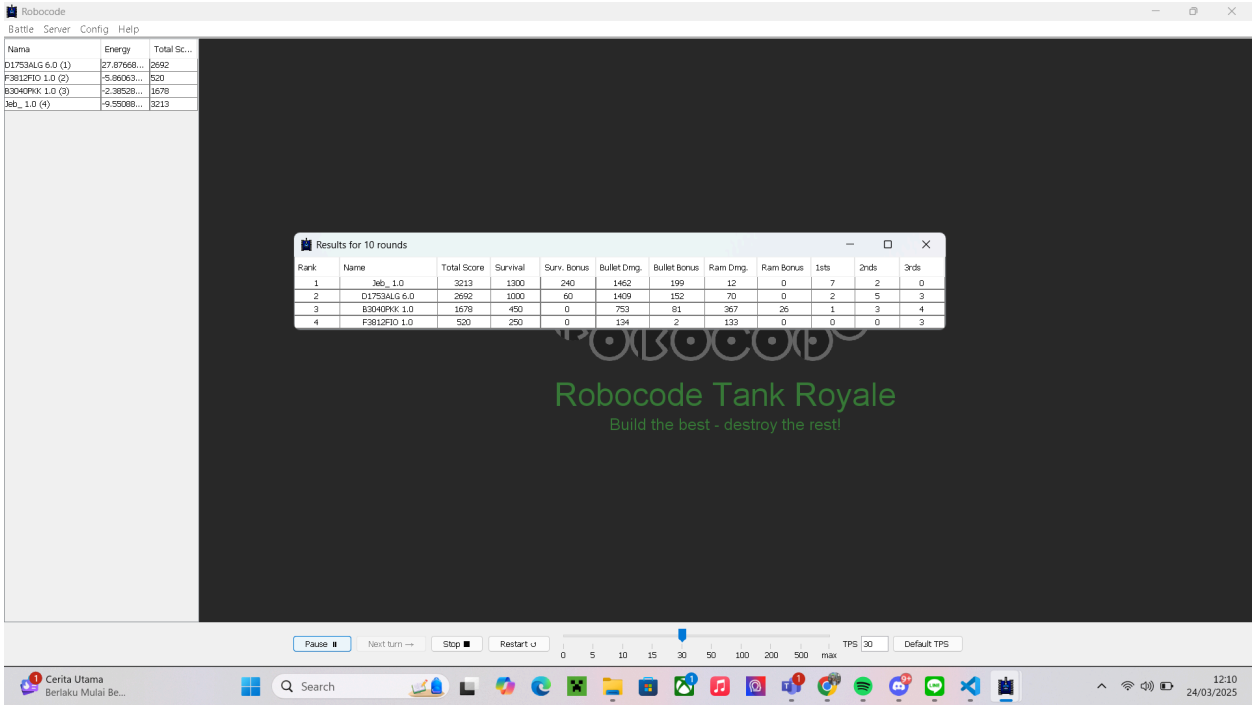
7. *void CalcBullet(double distance)*
   Fungsi yang digunakan untuk mengkalkulasi kecepatan dari peluru yang keluar berdasarkan fire power yang ditetapkan.

8. *void TurnGunToFaceTarget(double x, double y)*
   Fungsi untuk mengarahkan tembakan kepada target berdasarkan posisi target yang didapat melalui *OnScannedBot*.

9. *Color HsvToRgb()*
   Fungsi ini berguna untuk  untuk mengonversi warna dari model warna HSV (Hue, Saturation, Value) ke model warna RGB (Red, Green, Blue).

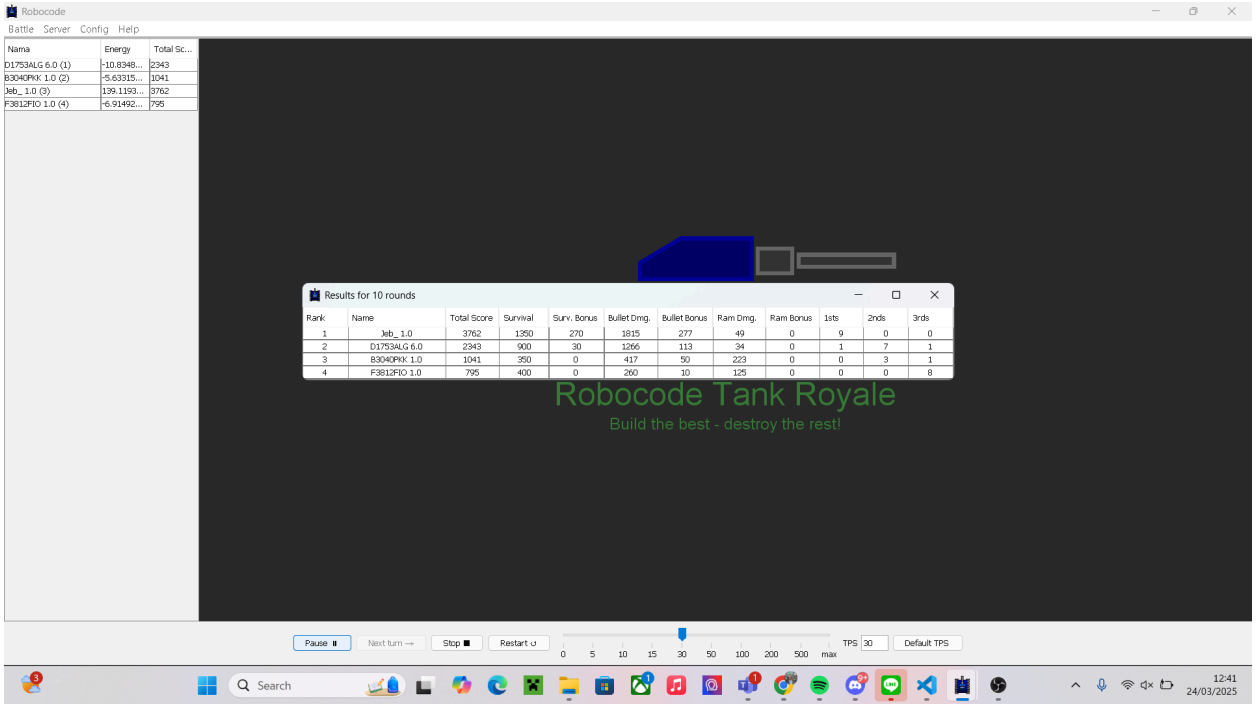
## 4.5 Pengujian

### Round 1: Tubes1_STIMA



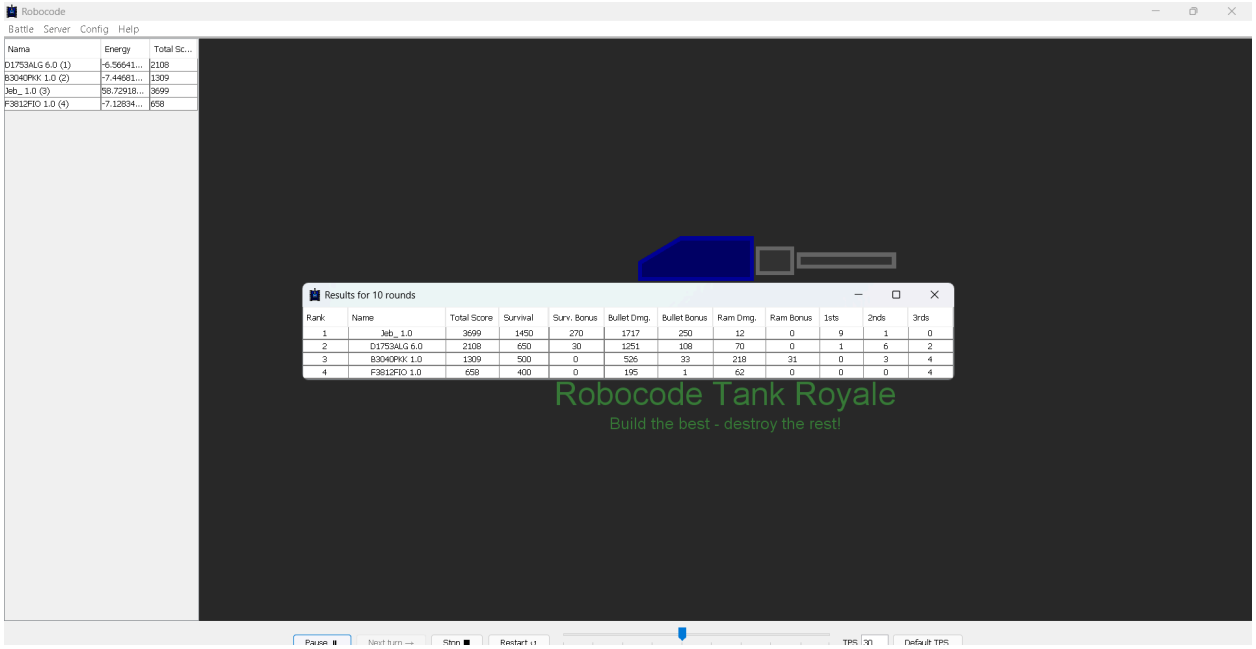

*Gambar 20. Round 1 Pertandingan Robot - Robot Kelompok D1753ALG*

### Round 2: Tubes1_STIMA




*Gambar 21. Round 2 Pertandingan Robot - Robot Kelompok D1753ALG*

## Round 3: Tubes1_STIMA




*Gambar 22. Round 3 Pertandingan Robot - Robot Kelompok D1753ALG*

# BAB 5

# KESIMPULAN DAN SARAN

## 5.1 Kesimpulan

Dalam tugas ini, kami telah menerapkan algoritma Greedy untuk mengembangkan bot dalam Robocode Tank Royale. Implementasi strategi ini memungkinkan bot untuk mengambil keputusan optimal di setiap langkah berdasarkan informasi yang tersedia saat itu. Dengan pendekatan ini, bot dapat bertindak cepat dalam memilih target, menembak secara efisien, serta melakukan ramming untuk memaksimalkan perolehan skor.

Dari hasil pengujian, strategi Greedy terbukti efektif dalam pertarungan 1v1 atau melawan musuh dengan mobilitas terbatas. Namun, dalam skenario yang lebih kompleks, seperti menghadapi musuh dengan strategi adaptif atau pertempuran dengan banyak lawan, bot mengalami keterbatasan akibat sifat Greedy yang hanya mempertimbangkan keuntungan jangka pendek.

Melalui tugas ini, kami juga memahami lebih dalam bagaimana algoritma Greedy bekerja dalam skenario praktis, termasuk kelebihan dan keterbatasannya dalam menghasilkan solusi optimal. Kami juga senang karena dapat mengeksplorasi strategi-strategi yang dapat dilakukan atau dikembangkan karena syarat pada tubes kali ini tidak terlalu spesifik sehingga kami banyak berdiskusi untuk merumusukan strategi atau metode ideal untuk hasil yang maksimal. Selain itu, karena pada tugas ini hanya diperbolehkan menggunakan algoritma greedy, namun kami sadar jika ingin membuat bot yang berpotensi untuk menjadi yang terbaik maka menelusuri strategi-strategi lain selain greedy dapat dilakukan untuk memaksimalkan hasil.

## 5.2 Saran

Saran untuk tugas besar ini adalah, lebih baik robot memiliki movement yang tidak dapat ditebak dan melakukan predictive shooting kepada musuh yang ada. Namun, kami juga memiliki saran terkait tugas besar ini yaitu, pembuatan 4 robot dengan 4 heuristik yang berbeda cukup sulit untuk mencari yang cukup jauh perbedaan heuristiknya dikarenakan tujuan dari game ini hanya mendapatkan poin terbanyak yang didapatkan oleh 2 cara, agresif atau survival point saja. Lalu pemberitahuan berapa bot yang akan ditandingkan dirahasiakan meskipun hasil dari pertandingan oleh satu bot dapat berbeda bergantung pada jumlah bot pada pertandingan sehingga saran kami adalah untuk menspesifikasi terkait jumlah bot pada pertandingan agar pembuat bot dapat menyesuaikan botnya agar dapat maksimal pada kondisi tersebut.

# LAMPIRAN

**Github:** https://github.com/BobSwagg13/Tubes1_D1753ALG

**Youtube Video:**https://youtu.be/sZzpA7PL9Ns?si=h0-anHqKQOK_0b8G