

**Tugas Besar 1 IF2211 Strategi Algoritma**  
**Pemanfaatan Algoritma Greedy dalam Pembuatan Bot**  
**Permainan Robocode Tank Royale**



Oleh:

**Kelompok 41 – BJB**

**Jonathan Kenan Budianto (13523139)**

**Benedictus Nelson (13523150)**

**Bryan P. Hutagalung (18222130)**

**Program Studi Teknik Informatika**

**Sekolah Teknik Elektro dan Informatika – Institut Teknologi Bandung**

**Jl. Ganesha 10, Bandung 40132**

**2025**

## Daftar Isi

<b>Daftar Isi.....</b>	<b>1</b>
<b>Bab I Deskripsi Tugas.....</b>	<b>3</b>
<b>Bab II Landasan Teori.....</b>	<b>9</b>
2.1 Algoritma Greedy.....	9
2.1.1 Teori Umum.....	9
2.1.2 Kelebihan dan Kekurangan.....	10
2.2 Program, Bot, dan Algoritma.....	11
2.2.1 Cara Kerja Program.....	11
2.2.2 Perlakuan Aksi Bot.....	11
2.2.3 Implementasi Algoritma Greedy pada Bot.....	12
2.2.4 Eksekusi Bot dan Program.....	13
<b>Bab III Metodologi Aplikasi Strategi Greedy.....</b>	<b>15</b>
3.1 Pemetaan Masalah menjadi Elemen Algoritma Greedy.....	15
3.1.1 Himpunan Kandidat.....	15
3.1.2 Himpunan Solusi.....	16
3.1.3 Fungsi Solusi.....	16
3.1.4 Fungsi Seleksi.....	17
3.1.5 Fungsi Kelayakan.....	18
3.1.6 Fungsi Objektif.....	18
3.2 Eksplorasi Alternatif Solusi Greedy.....	19
3.2.1 Alternatif Solusi 1 - Pencarian Posisi Optimal (BJB).....	19
3.2.2 Alternatif Solusi 2 - Gerakan Sudut dan Penembakan Berdasarkan Jarak (Kenan).....	20
3.2.3 Alternatif Solusi 3 - Penghindaran Tembakan (Nelson).....	21
3.2.4 Alternatif Solusi 4 - Pencarian Posisi Optimal (Bryan).....	22
3.3 Analisis Efisiensi dan Efektivitas.....	22
3.3.1 Alternatif Solusi 1 - Zoning dan Manuver (BJB).....	22

3.3.2 Alternatif Solusi 2 – Gerakan Sudut dan Penembakan Berdasarkan Jarak (Kenan).....	23
3.3.3 Alternatif Solusi 3 – Penghindaran Tembakan (Nelson).....	23
3.3.4 Alternatif Solusi 4 – Pencarian Posisi Optimal (Bryan).....	24
3.4 Pemilihan Strategi Greedy Utama.....	24
<b>Bab IV Implementasi dan Pengujian.....</b>	<b>26</b>
4.1 Pseudocode Implementasi Solusi.....	26
4.1.1 Alternatif Solusi 1 – Zoning dan Manuver (BJB).....	26
4.1.2 Alternatif Solusi 2 – Gerakan ke dan Penembakan Berdasarkan Jarak (Kenan). 28	
4.1.3 Alternatif Solusi 3 – Penghindaran Tembakan (Nelson).....	29
4.1.4 Alternatif Solusi 4 – Pencarian Posisi Optimal (Bryan).....	31
4.2 Deskripsi Struktur Data, Fungsi, dan Prosedur.....	34
4.2.1 Struktur Data.....	34
4.2.2 Fungsi dan Prosedur.....	35
4.3 Pengujian Bot Utama dan Bot Asisten.....	38
4.3.1 Pengujian 1.....	38
4.3.2 Pengujian 2.....	39
4.3.3 Pengujian 3.....	40
4.4 Analisis Hasil Pengujian.....	41
<b>Bab V Penutup.....</b>	<b>43</b>
5.1 Kesimpulan.....	43
5.2 Saran.....	44
<b>Daftar Pustaka.....</b>	<b>46</b>

## **Bab I**

### **Deskripsi Tugas**

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

Pada Tugas Besar pertama Strategi Algoritma ini, mahasiswa diminta untuk membuat sebuah bot yang nantinya akan dipertandingkan satu sama lain. Tentunya mahasiswa harus menggunakan strategi greedy dalam membuat bot ini. Komponen-komponen dari permainan ini antara lain:

#### **1. Rounds dan Turns**

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

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

Pada setiap turn, sebuah bot dapat:

- Menggerakkan bot, memindai musuh, dan menembakkan senjata.

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

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

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

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

## **2. Batas Waktu Giliran**

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

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

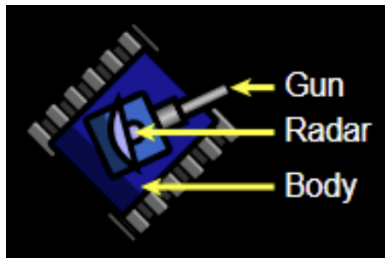
#### **6. Tabrakan**

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

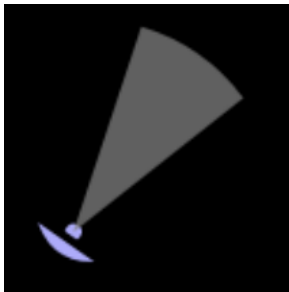
saat ini. Semakin cepat bot bergerak, semakin lambat kemampuannya untuk berbelok.

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

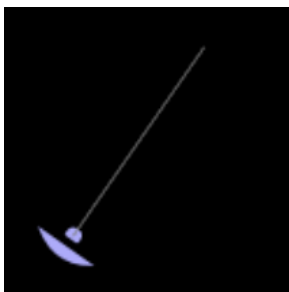
## 10. Pemindaian

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

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

Untuk informasi lebih lengkap, silahkan buka dokumentasi Tank Royale pada link [berikut](#).

## **Bab II**

### **Landasan Teori**

#### **2.1 Algoritma Greedy**

##### **2.1.1 Teori Umum**

Algoritma greedy adalah salah satu pendekatan dalam pemrograman yang digunakan untuk menyelesaikan masalah optimisasi. Prinsip dasar dari algoritma ini adalah memilih solusi terbaik yang ada pada setiap langkah, dengan harapan pilihan-pilihan lokal tersebut akan menghasilkan solusi yang optimal secara keseluruhan. Intinya, algoritma greedy selalu memilih opsi yang terlihat terbaik pada saat itu tanpa mempertimbangkan konsekuensi jangka panjang dari keputusan tersebut.

Pada setiap tahap, algoritma greedy mengambil keputusan berdasarkan informasi yang ada saat itu, dan biasanya fokus pada pilihan yang akan memberikan keuntungan paling besar atau mengurangi biaya secara maksimal. Meskipun algoritma ini sangat efisien dan seringkali memberikan solusi dengan cepat, bukan berarti hasilnya selalu optimal untuk semua masalah. Ada beberapa kasus di mana algoritma greedy gagal memberikan solusi terbaik, terutama jika masalah tersebut tidak memenuhi syarat tertentu, seperti tidak memiliki sifat greedy optimal.

Salah satu contoh klasik dari algoritma greedy adalah masalah pemberian kembalian uang atau coin change. Misalnya, jika ingin memberikan kembalian dengan jumlah uang yang minimal, menggunakan koin-koin dengan denominasi tertentu. Dalam kasus ini, algoritma greedy akan memilih koin dengan nilai terbesar terlebih dahulu hingga jumlah yang diinginkan tercapai. Meskipun pendekatan ini cepat, ada kasus di mana solusi terbaik tidak tercapai hanya dengan memilih koin terbesar terlebih dahulu.

Berikut adalah langkah-langkah umum dalam penerapan algoritma greedy:

1. Tentukan langkah-langkah atau keputusan lokal yang akan diambil pada setiap tahap.
2. Pilih solusi lokal yang terbaik pada setiap langkah.
3. Ulangi langkah tersebut sampai mencapai solusi akhir.
4. Susun solusi akhir berdasarkan keputusan-keputusan lokal yang telah diambil.

### **2.1.2 Kelebihan dan Kekurangan**

#### **Kelebihan:**

##### **1. Efisiensi Waktu**

Algoritma greedy seringkali lebih cepat dibandingkan algoritma lain yang lebih kompleks, seperti dynamic programming atau backtracking. Hal ini karena greedy hanya fokus pada keputusan lokal yang sederhana.

##### **2. Kesederhanaan**

Algoritma ini mudah dipahami dan diterapkan karena tidak memerlukan proses yang rumit. Cukup memilih solusi terbaik pada setiap langkah tanpa perlu memikirkan seluruh gambaran besar.

##### **3. Solusi yang Cukup Baik**

Meskipun tidak selalu optimal, untuk banyak kasus, algoritma greedy memberikan solusi yang cukup baik dan memadai untuk diterapkan dalam praktik.

#### **Kekurangan:**

##### **1. Tidak Selalu Optimal**

Karena hanya mempertimbangkan keputusan lokal, algoritma greedy tidak selalu menghasilkan solusi yang optimal untuk setiap masalah. Khususnya pada masalah yang melibatkan ketergantungan jangka panjang atau tidak memenuhi kriteria greedy optimal.

## **2. Terbatas pada Keputusan Lokal**

Algoritma ini fokus pada keputusan yang dapat diambil saat itu juga, tanpa mempertimbangkan keseluruhan konteks atau gambaran besar dari masalah. Hal ini bisa membuat algoritma ini salah arah jika solusi global memerlukan pertimbangan yang lebih mendalam.

Dengan demikian, meskipun algoritma greedy sangat bermanfaat untuk masalah yang dapat diselesaikan dengan cara ini, namun perlu dipahami kapan pendekatan ini efektif dan kapan pendekatan lain mungkin diperlukan untuk mendapatkan solusi yang lebih optimal.

## **2.2 Program, Bot, dan Algoritma**

### **2.2.1 Cara Kerja Program**

Dalam permainan Robocode Tank Royale, setiap bot yang dibuat bertarung dengan bot lainnya berdasarkan algoritma atau instruksi yang diberikan oleh pembuatnya. Setiap bot akan melakukan aksi sesuai dengan kode program yang diterapkan. Salah satu strategi yang sering digunakan dalam pembuatan bot adalah algoritma greedy. Dalam konteks ini, bot akan memprioritaskan tindakan yang memberikan keuntungan langsung atau menghindari kerugian secara lokal pada setiap giliran pertempuran.

### **2.2.2 Perlakuan Aksi Bot**

Di dalam Robocode Tank Royale, setiap bot memiliki beberapa komponen yang dapat bergerak secara independen, seperti tubuh (body), senjata (gun), dan radar. Setiap komponen ini bisa diprogram untuk bergerak, memindai, dan menembak sesuai dengan situasi yang ada. Algoritma greedy dapat diterapkan untuk membantu bot dalam membuat keputusan yang optimal berdasarkan kondisi yang ada saat itu.

Berikut adalah proses dasar bot dalam satu giliran (turn):

#### **1. Pemindaian (Scan)**

Bot menggunakan radar untuk memindai sekelilingnya dan

mendeteksi keberadaan bot musuh. Dengan algoritma greedy, bot akan fokus pada musuh yang paling dekat, karena musuh yang dekat lebih mudah dijangkau dan lebih cepat bisa dihindari atau diserang.

## **2. Gerakan (Move)**

Berdasarkan hasil pemindaian, bot akan memilih gerakan yang dapat menghindari ancaman atau mendekati musuh secara langsung untuk menyerang. Dalam algoritma greedy, bot memilih gerakan yang paling menguntungkan dalam jangka pendek, seperti bergerak ke posisi yang menghindari serangan musuh atau mendekat untuk menyerang.

## **3. Menembak (Shoot)**

Setelah mendeteksi musuh, bot akan menembak peluru ke arah musuh. Dalam hal ini, algoritma greedy akan memilih peluru yang cukup kuat untuk memberikan damage besar, tetapi juga mempertimbangkan kecepatan peluru agar bisa tepat sasaran sebelum musuh menghindar.

## **4. Menghindari (Dodge)**

Jika bot mendeteksi bahwa ia akan diserang, bot akan berusaha menghindar. Algoritma greedy akan memilih pola gerakan yang memberikan peluang terbaik untuk menghindari tembakan musuh, seperti bergerak secara acak atau mengikuti pola yang sulit ditebak oleh musuh.

### **2.2.3 Implementasi Algoritma Greedy pada Bot**

Implementasi algoritma greedy dalam bot Robocode bisa dilakukan dengan cara membuat bot selalu membuat keputusan berdasarkan kondisi yang ada saat itu. Beberapa langkah umum dalam implementasi algoritma greedy pada bot adalah sebagai berikut:

#### **1. Pemetaan Keputusan**

Tentukan parameter yang perlu dievaluasi di setiap giliran, seperti

posisi musuh, energi yang tersisa, atau suhu senjata. Misalnya, jika bot sangat dekat dengan musuh, maka bot mungkin memilih untuk menembak dengan peluru yang lebih kuat meskipun peluru tersebut lebih lambat.

## **2. Strategi Menghindar**

Bot bisa mengevaluasi ancaman yang ada dan menghindari gerakan yang berisiko tinggi, seperti menghantam dinding atau berpapasan dengan musuh yang lebih kuat. Algoritma greedy akan memilih langkah yang meminimalkan risiko pada setiap giliran.

## **3. Pemilihan Tindakan Terbaik**

Bot akan memilih langkah terbaik berdasarkan perhitungan saat itu. Misalnya, bot akan memilih untuk menembak dengan peluru berat jika musuh sudah sangat dekat atau menggunakan peluru ringan jika musuh masih berada pada jarak jauh.

## **4. Optimasi Waktu**

Mengingat batas waktu pada setiap giliran, bot harus membuat keputusan dengan cepat. Algoritma greedy akan sangat bergantung pada keputusan yang dapat diambil dalam waktu singkat, tanpa membutuhkan perhitungan yang rumit.

### **2.2.4 Eksekusi Bot dan Program**

Untuk menjalankan bot di arena Robocode, berikut adalah langkah-langkah yang perlu dilakukan:

#### **1. Penulisan Kode Bot**

Bot ditulis dalam bahasa Java menggunakan API Robocode, yang mengatur perilaku bot, seperti gerakan, pemindaian, dan penembakan peluru.

#### **2. Menggunakan Algoritma Greedy**

Saat menulis kode, penerapan algoritma greedy dilakukan pada setiap aksi bot. Misalnya, bot akan memprioritaskan musuh yang

paling dekat dan memilih peluru yang lebih cepat jika musuh berada dalam jarak dekat. Algoritma greedy ditulis dalam bahasa C#.

### **3. Pengujian dan Perbaikan**

Setelah bot dibuat, bot akan diuji dalam arena Robocode untuk melawan bot lain. Berdasarkan hasil pertarungan, bot akan diperbaiki dengan mengubah strategi agar semakin optimal dalam menghadapi berbagai situasi.

### **4. Optimasi Berkelanjutan**

Setelah pengujian, bot dapat dioptimalkan lebih lanjut dengan memperbaiki cara bot memilih musuh atau cara bot menghindari serangan. Algoritma greedy dapat terus disempurnakan agar bot selalu membuat keputusan terbaik di setiap giliran.

## Bab III

### Metodologi Aplikasi Strategi Greedy

#### 3.1 Pemetaan Masalah menjadi Elemen Algoritma Greedy

Pada permainan Robocode Tank Royale, pembuatan bot yang efektif memerlukan pemahaman tentang bagaimana menerjemahkan elemen-elemen algoritma greedy ke dalam konteks pertempuran antar bot. Algoritma greedy bekerja dengan memilih tindakan terbaik secara lokal di setiap giliran untuk mencapai tujuan yang lebih besar, yaitu memenangkan pertarungan. Elemen-elemen dalam algoritma greedy, seperti **himpunan kandidat**, **himpunan solusi**, **fungsi solusi**, **fungsi seleksi**, **fungsi kelayakan**, dan **fungsi objektif**, dapat diterapkan untuk membuat keputusan terbaik bagi bot dalam setiap langkah pertempuran. Berikut adalah penjelasan bagaimana elemen-elemen tersebut diterapkan dalam permainan Robocode Tank Royale.

##### 3.1.1 Himpunan Kandidat

Himpunan kandidat adalah berbagai pilihan atau tindakan yang dapat dipilih pada setiap giliran dalam permainan. Dalam konteks Robocode Tank Royale, himpunan kandidat ini mencakup berbagai aksi yang bisa dilakukan oleh bot pada satu giliran, seperti:

###### 1. Pergerakan (Move)

Bot dapat bergerak maju, mundur, atau berbelok untuk menghindari musuh atau memposisikan diri untuk menyerang.

###### 2. Pemindaian (Scan)

Bot memindai area sekitar untuk mendeteksi keberadaan musuh.

###### 3. Penembakan (Shoot)

Bot dapat menembakkan peluru dengan berbagai kekuatan, tergantung pada jarak dan situasi.



#### **4. Menghindar (Dodge)**

Bot dapat bergerak acak atau mengikuti pola tertentu untuk menghindari serangan musuh.

#### **5. Ramming**

Bot dapat mencoba menabrak musuh jika posisinya menguntungkan.

Setiap tindakan ini merupakan bagian dari himpunan kandidat yang akan dievaluasi untuk memilih langkah terbaik pada giliran tersebut.

### **3.1.2 Himpunan Solusi**

Himpunan solusi adalah sekumpulan tindakan yang dipilih oleh bot untuk mencapai tujuan tertentu dalam pertempuran. Solusi ini terbentuk dari keputusan-keputusan yang diambil di setiap giliran, yang pada akhirnya menentukan hasil akhir yang optimal, seperti kelangsungan hidup atau peningkatan skor.

Misalnya, jika bot memutuskan untuk mundur (untuk menghindari musuh), memindai musuh, lalu menembakkan peluru, maka kombinasi langkah tersebut menjadi salah satu solusi yang diterapkan dalam permainan. Himpunan solusi bisa berupa berbagai kombinasi gerakan, tembakan, atau penghindaran yang dipilih untuk memaksimalkan peluang kemenangan dan kelangsungan hidup bot.

### **3.1.3 Fungsi Solusi**

Fungsi solusi adalah aturan atau metode yang digunakan untuk memastikan bahwa tindakan yang diambil dapat diimplementasikan dengan baik dalam kondisi permainan saat itu. Fungsi ini mengevaluasi apakah keputusan yang dibuat sesuai dengan aturan permainan dan memberikan dampak positif bagi bot dan musuh.

Contohnya, jika bot memilih untuk bergerak maju dan menembak, fungsi solusi akan memeriksa apakah ada musuh di depan yang bisa

diserang, apakah suhu meriam cukup rendah untuk menembak, dan apakah bot memiliki cukup energi untuk menembak. Jika bot memilih untuk bergerak mundur dan menghindar, fungsi solusi memastikan bahwa bot menghindari posisi berbahaya, seperti menabrak dinding atau bertemu dengan musuh yang lebih kuat.

#### **3.1.4 Fungsi Seleksi**

Fungsi seleksi adalah mekanisme untuk memilih solusi terbaik dari himpunan kandidat berdasarkan kriteria yang relevan. Dalam algoritma greedy, pada setiap langkah atau giliran, fungsi seleksi memilih opsi yang paling menguntungkan secara lokal.

Dalam Robocode Tank Royale, fungsi seleksi akan memilih tindakan yang memberikan keuntungan langsung bagi bot. Beberapa kriteria untuk pemilihan aksi antara lain:

##### **1. Energi yang tersisa**

Jika energi bot sudah sedikit, fungsi seleksi akan memilih untuk menghindar atau menjauh dari musuh.

##### **2. Posisi bot**

Jika musuh terlalu dekat, seleksi bisa memilih untuk menyerang dengan menembak atau melakukan ramming.

##### **3. Status suhu meriam**

Jika suhu meriam terlalu tinggi, fungsi seleksi akan memilih untuk tidak menembak atau memilih tembakan yang lebih ringan.

Fungsi seleksi akan selalu memilih langkah yang memaksimalkan peluang untuk bertahan hidup atau memberikan kerusakan yang besar pada musuh.

### 3.1.5 Fungsi Kelayakan

Fungsi kelayakan memastikan bahwa tindakan yang dipilih sesuai dengan aturan permainan dan tidak akan menyebabkan kerugian besar. Fungsi ini memastikan bahwa setiap langkah yang diambil adalah sah dan tidak melanggar batasan waktu, energi, atau peraturan lainnya.

Misalnya, jika bot memilih untuk menembak tapi suhu meriam terlalu tinggi, maka solusi tersebut tidak dapat diterima. Bot harus mencari alternatif lain. Selain itu, fungsi kelayakan juga mempertimbangkan batas waktu giliran. Jika waktu habis tanpa adanya perintah, bot akan melewatkan giliran, yang akan merugikan. Fungsi kelayakan memastikan bahwa semua keputusan dapat diselesaikan dalam batas waktu yang tersedia.

### 3.1.6 Fungsi Objektif

Fungsi objektif digunakan untuk mengukur sejauh mana suatu solusi memenuhi tujuan yang ditetapkan. Dalam permainan Robocode Tank Royale, tujuan utamanya adalah untuk memaksimalkan skor dan bertahan hidup hingga ronde terakhir.

Beberapa komponen skor yang diperhitungkan dalam fungsi objektif antara lain:

#### 1. Bullet Damage

Kerusakan yang diberikan oleh peluru bot terhadap musuh.

#### 2. Bullet Damage Bonus

Poin tambahan yang didapat saat bot berhasil menghancurkan musuh dengan peluru.

#### 3. Survival Score

Poin yang diperoleh ketika bot bertahan hidup setelah musuh-musuh lainnya hancur.

#### 4. Ram Damage

Poin yang didapat ketika bot berhasil menabrak musuh.

### **5. Ram Damage Bonus**

Poin tambahan saat musuh hancur akibat ditabrak.

### **6. Last Survival Bonus**

Poin untuk bot yang bertahan hingga ronde terakhir.

Fungsi objektif ini membantu bot untuk mengevaluasi apakah tindakan atau kombinasi tindakan yang diambil akan mengarah pada hasil yang lebih baik, baik dari segi skor maupun kelangsungan hidup. Setiap langkah yang diambil berdasarkan algoritma greedy bertujuan untuk memaksimalkan skor dan meningkatkan peluang untuk bertahan hidup.

## **3.2 Eksplorasi Alternatif Solusi Greedy**

Pada permainan Robocode Tank Royale, kita bisa mengeksplorasi beberapa alternatif solusi menggunakan algoritma greedy. Setiap solusi ini mengandalkan heuristik yang berbeda untuk menentukan langkah terbaik bagi bot. Berikut adalah empat alternatif solusi greedy yang bisa diterapkan dalam permainan ini, masing-masing dengan pendekatan yang berbeda sesuai dengan kondisi dan strategi yang diinginkan.

### **3.2.1 Alternatif Solusi 1 – Pencarian Posisi Optimal (BJB)**

#### **1. Heuristik**

Bot membagi arena menjadi beberapa zona dan fokus untuk tetap berada di zona yang aman. Jika bot terjebak atau didekati musuh, bot akan memilih manuver tertentu untuk keluar dari zona berisiko.

#### **2. Strategi**

Bot mengontrol posisinya dengan menghindari dinding dan musuh, menggunakan manuver penghindaran jika diperlukan, serta memilih posisi terbaik berdasarkan pembagian zona di arena. Bot akan berpindah ke zona yang lebih aman jika diperlukan.

#### **3. Kelebihan**

- a. Menggunakan strategi zonasi memungkinkan bot untuk mengontrol pergerakannya dengan lebih terstruktur.
- b. Memperbesar peluang bot untuk menghindari situasi berisiko dengan berpindah ke zona yang lebih aman.

#### **4. Kekurangan**

- a. Membutuhkan lebih banyak perhitungan untuk menentukan zona yang aman dan posisi strategis.
- b. Jika musuh dapat mengejar atau memprediksi pergerakan bot, strategi ini mungkin kurang efektif dalam beberapa situasi.

### **3.2.2 Alternatif Solusi 2 – Gerakan Sudut dan Penembakan Berdasarkan Jarak (Kenan)**

#### **1. Heuristik**

Bot menggunakan gerakan ke setiap sudut untuk menghindari prediksi musuh dan menembak berdasarkan jarak dengan musuh. Bot juga bisa mengunjungi titik sudut arena untuk menghindari musuh atau memperbarui posisinya.

#### **2. Strategi**

Bot bergerak ke setiap sudut untuk mengacaukan prediksi musuh dan membuat dirinya lebih sulit untuk dibaca. Selain itu, bot akan menembak berdasarkan jarak dengan musuh. Jika bertemu musuh atau terkena serangan, bot bisa mundur atau bergerak ke sudut arena untuk repositioning.

#### **3. Kelebihan**

- a. Gerakan ke setiap sudut membuat bot sulit diprediksi oleh musuh, meningkatkan ketahanan bot terhadap serangan.
- b. Menembak berdasarkan jarak memungkinkan bot untuk menyesuaikan kekuatan tembakan secara efektif sesuai dengan posisi musuh.

#### **4. Kekurangan**

- a. Gerakan ke setiap sudut tidak selalu mengarah ke posisi strategis dan bisa membuang-buang waktu tanpa keuntungan nyata.
- b. Jika gerakan ke setiap sudut terlalu sering dilakukan, bot bisa kehilangan peluang untuk menyerang musuh atau mengatur strategi yang lebih terstruktur.

### **3.2.3 Alternatif Solusi 3 – Penghindaran Tembakan (Nelson)**

#### **1. Heuristik**

Bot mendeteksi tembakan musuh dan langsung beradaptasi dengan melakukan manuver penghindaran (dodge) untuk menghindari kerusakan lebih lanjut.

#### **2. Strategi**

Ketika bot mendeteksi tembakan musuh, bot segera memasuki mode penghindaran, mengubah arah secara cepat untuk menghindari tembakan lebih lanjut. Selain itu, bot berusaha untuk menghindari area yang berisiko terkena serangan musuh.

#### **3. Kelebihan**

- a. Mengurangi kerusakan yang diterima, meningkatkan kelangsungan hidup bot.
- b. Menjadikan bot lebih lincah dan sulit diserang karena terus bergerak menghindari tembakan.

#### **4. Kekurangan**

- a. Penghindaran yang berlebihan dapat mengurangi kesempatan untuk menyerang musuh, yang bisa menghambat perolehan skor.

- 5. Mode dodge membutuhkan perhitungan yang tepat, yang dapat membuat logika lebih kompleks dan memperlambat pengambilan keputusan.

### **3.2.4 Alternatif Solusi 4 – Pencarian Posisi Optimal (Bryan)**

#### **1. Heuristik**

Bot mengutamakan pemilihan posisi yang lebih aman, dengan menjaga jarak dari batas arena dan bergerak menuju pusat arena jika terlalu dekat dengan dinding.

#### **2. Strategi**

Bot akan selalu menghindari dinding dan bergerak menuju pusat arena jika posisinya sudah terlalu dekat dengan batas arena. Fokus utama dari strategi ini adalah menjaga posisi bot agar tidak terjebak di pinggir arena yang bisa membatasi gerakan.

#### **3. Kelebihan**

- a. Menghindari bot terjebak di pinggir arena, memberi kebebasan gerak yang lebih baik.
- b. Memberikan kesempatan untuk menjaga posisi yang lebih aman dan menghindari pertemuan langsung dengan musuh.

#### **4. Kekurangan**

- a. Fokus pada gerakan evasif mengurangi waktu yang tersedia untuk menyerang musuh.
- b. Bot mungkin terlalu sering mundur tanpa ada serangan agresif, mengurangi potensi untuk meraih kemenangan.

## **3.3 Analisis Efisiensi dan Efektivitas**

### **3.3.1 Alternatif Solusi 1 – Zoning dan Manuver (BJB)**

#### **1. Efisiensi**

Strategi zoning lebih efisien dalam hal pengelolaan posisi karena bot bisa mengatur pergerakannya dengan lebih terstruktur. Namun, perhitungan zona yang aman dan posisi yang tepat dapat memperlambat pengambilan keputusan.

## **2. Efektivitas**

Efektivitas strategi ini sangat tinggi, karena bot selalu berada di posisi yang aman dan terorganisir. Manuver penghindaran yang terstruktur meningkatkan peluang bot untuk bertahan hidup sambil tetap bisa menyerang musuh dengan efektif. Strategi ini lebih cocok untuk pertarungan jangka panjang, meskipun memerlukan perhitungan yang lebih rumit.

### **3.3.2 Alternatif Solusi 2 – Gerakan Sudut dan Penembakan Berdasarkan Jarak (Kenan)**

#### **1. Efisiensi**

Gerakan ke setiap sudut cukup efisien karena bot tidak perlu melakukan banyak perhitungan strategis. Namun, gerakan ke setiap sudut bisa menghabiskan waktu dan membuat bot kehilangan peluang untuk menyerang secara efektif.

#### **2. Efektivitas**

Dalam hal efektivitas, gerakan ke setiap sudut bisa membingungkan musuh dan memberi bot kesempatan untuk menyerang secara efektif berdasarkan jarak. Namun, gerakan ke setiap sudut juga bisa membuat bot kehilangan kontrol terhadap posisi strategis, yang dapat merugikan dalam beberapa situasi.

### **3.3.3 Alternatif Solusi 3 – Penghindaran Tembakan (Nelson)**

#### **1. Efisiensi**

Algoritma penghindaran tembakan lebih kompleks karena bot harus mendeteksi tembakan musuh dan menghitung manuver penghindaran. Ini sedikit menambah beban komputasi dan bisa memperlambat pengambilan keputusan.

#### **2. Efektivitas**

Penghindaran tembakan sangat efektif dalam meningkatkan



kelangsungan hidup bot. Namun, penghindaran yang terlalu sering dilakukan bisa mengurangi agresivitas bot dan menghambat pencapaian skor. Bot harus menemukan keseimbangan antara bertahan hidup dan menyerang untuk meningkatkan efektivitasnya.

#### **3.3.4 Alternatif Solusi 4 – Pencarian Posisi Optimal (Bryan)**

##### **1. Efisiensi**

Solusi ini cukup efisien karena fokus utamanya adalah pada pergerakan yang sederhana, yaitu menghindari dinding dan bergerak ke pusat. Bot tidak perlu melakukan banyak perhitungan rumit, sehingga pengambilan keputusan bisa lebih cepat.

##### **2. Efektivitas**

Meskipun efisien, strategi ini kurang efektif dalam meningkatkan skor karena bot lebih banyak menghindari daripada menyerang. Bot yang terlalu banyak bergerak menjauh dari musuh berisiko kesulitan untuk meraih kemenangan.

### **3.4 Pemilihan Strategi Greedy Utama**

Dari keempat alternatif solusi greedy yang diuji, strategi **Zoning dan Manuver (Bot BJB)** dipilih sebagai strategi utama. Ada beberapa alasan yang mendasari pemilihan strategi ini, yaitu:

#### **1. Penghindaran Dinding yang Efektif**

Strategi ini membantu bot untuk selalu menghindari dinding atau batas arena, yang merupakan faktor penting dalam permainan Robocode Tank Royale. Dengan memastikan bahwa bot selalu berada di dalam zona aman, risiko terjebak di sudut arena atau dalam situasi berbahaya dapat diminimalkan. Hal ini memungkinkan bot untuk memiliki lebih banyak ruang untuk bergerak dan bertahan hidup.

#### **2. Fleksibilitas dalam Penentuan Zona**

Bot ini membagi arena menjadi beberapa zona dan menggunakan strategi

untuk beradaptasi dengan zona yang lebih aman atau lebih menguntungkan. Ketika bot berada di zona yang kurang aman, seperti dekat dinding atau posisi berisiko, bot bisa secara otomatis bergerak menuju zona yang lebih aman atau strategis. Ini memberi fleksibilitas lebih dalam pergerakan dan meningkatkan peluang bertahan hidup dalam pertempuran.

### **3. Kemampuan Menghindari Tembakan Musuh**

Dengan menggunakan strategi penghindaran yang lebih dinamis, seperti berpindah zona atau menjauh dari tembakan musuh, Ecus Bot dapat tetap berada dalam posisi yang lebih baik. Jika bot mendeteksi serangan atau tembakan, bot akan bergerak untuk menghindar dan mencari posisi aman yang lebih menguntungkan.

### **4. Strategi Manuver yang Dapat Beradaptasi**

Keunggulan dari strategi Zoning dan Manuver adalah kemampuan bot untuk melakukan manuver acak atau berpindah-pindah zona guna menghindari serangan musuh. Ini membuatnya lebih sulit diprediksi oleh musuh, memberikan keunggulan dalam situasi pertempuran yang dinamis dan tak terduga.

### **5. Optimasi Penggunaan Energi**

Dengan selalu memastikan bahwa bot berada dalam posisi strategis dan menghindari situasi berisiko tinggi (seperti terjebak di dinding atau terlalu dekat dengan musuh), bot dapat mengoptimalkan penggunaan energi. Ini juga membantu bot untuk menghindari pemborosan energi yang tidak perlu selama pertempuran, sehingga bot tetap siap menghadapi serangan yang lebih besar.

## Bab IV

### Implementasi dan Pengujian

#### 4.1 Pseudocode Implementasi Solusi

##### 4.1.1 Alternatif Solusi 1 – Zoning dan Manuver (BJB)

```
# Inisialisasi variabel
Initialize movement flags (currentZone, moveClockwise)
Initialize arena boundaries and zone definitions (MIN_Y_UPPER,
MIN_X_LEFT, etc.)

# Fungsi utama untuk menjalankan bot
function Run():
    SetBotColors() # Set warna bot
    InitializeMovement() # Inisialisasi gerakan awal

    while (bot is running):
        CheckIfStuck() # Cek apakah bot terjebak
        CheckPosition() # Cek posisi dan zona

        if (needToReposition):
            MoveToAllowedZone() # Pindah ke zona yang aman

            RandomizedMovement() # Gerakan acak jika perlu
            ScanningMode() # Mode pemindaian

# Fungsi untuk memeriksa apakah bot terjebak
function CheckIfStuck():
```

```

distanceMoved = CalculateDistanceMoved()
if (distanceMoved < 2):
    stuckCounter += 1
    if (stuckCounter >= STUCK_THRESHOLD):
        EmergencyUnstuck() # Lakukan manuver darurat

# Fungsi untuk melakukan manuver darurat jika bot terjebak
function EmergencyUnstuck():
    EscapeFromWallsOrCorners() # Jika terjebak di sudut atau dekat dinding

# Fungsi untuk bergerak acak
function RandomizedMovement():
    if (IsNearWall()):
        AvoidWall() # Hindari dinding
    else:
        MoveForwardRandomly() # Gerakan maju acak

# Fungsi untuk menghindari dinding
function AvoidWall():
    TurnRight(RandomBetween(120, 240)) # Putar acak untuk menghindari
    dinding
    SetForward(100) # Bergerak maju

# Fungsi untuk memeriksa apakah bot terjebak di dinding
function IsNearWall():
    return (X < WALL_SAFETY_DISTANCE or X > ArenaWidth -
    WALL_SAFETY_DISTANCE)

```

#### 4.1.2 Alternatif Solusi 2 – Gerakan Sudut dan Penembakan Berdasarkan Jarak (Kenan)

```
# Inisialisasi variabel
Initialize movement flags (cornerIndex, stopWhenSeeEnemy)
Initialize corner positions (Corners array)

# Fungsi utama untuk menjalankan bot
function Run():
    # Set warna bot
    SetBodyColor(DarkGreen)
    SetTurretColor(Yellow)
    SetRadarColor(Yellow)

    GoToCorner() # Pindah ke sudut pertama

    while (bot is running):
        iterations = CalculateTurnIterations() # Hitung jumlah iterasi putaran
        for i = 0 to iterations:
            if (i == iterations / 2):
                AimGunToCenter() # Mengarahkan senjata ke tengah arena
            else:
                TurnGunRight(3) # Putar senjata sedikit-sedikit
            SwitchTurnDirection() # Ubah arah tembakan setelah setengah iterasi

# Fungsi untuk pergi ke sudut tertentu
function GoToCorner():
    targetCorner = Corners[cornerIndex] # Tentukan sudut yang akan dituju
    MoveToTarget(targetCorner) # Bergerak ke sudut tersebut
```

```

# Fungsi untuk menghitung jumlah iterasi putaran berdasarkan jarak
function CalculateTurnIterations():
    distX = abs(X - CenterX) / CenterX # Hitung jarak relatif ke pusat
    distY = abs(Y - CenterY) / CenterY
    cornerFactor = sqrt(distX * distX + distY * distY) / sqrt(2)
    iterations = max(30, min(60, 60 - int(cornerFactor * 30))) # Tentukan
    jumlah iterasi
    return iterations

# Fungsi untuk mengarahkan senjata ke pusat
function AimGunToCenter():
    bearing = GunBearingTo(CenterX, CenterY)
    TurnGunLeft(bearing) # Putar senjata ke arah pusat arena

```

#### 4.1.3 Alternatif Solusi 3 – Penghindaran Tembakan (Nelson)

```

# Inisialisasi variabel
Initialize movement flags (movingForward, randomMoveCounter,
dodgeMode)
Initialize scan variables (turnsSinceScan, SCAN_THRESHOLD)

# Fungsi utama untuk menjalankan bot
function Run():
    # Set warna bot
    SetBodyColor(Lime)
    SetGunColor(Green)
    SetRadarColor(DarkCyan)

```

```

SetBulletColor(Yellow)
SetScanColor(LightPink)

while (bot is running):
    DoFallbackRadarIfLost()
    StayAwayFromWalls() # Menghindari dinding
    RandomMoveIfNeeded() # Gerakan acak setelah interval

    if (dodgeMode):
        PerformDodgeManeuver() # Manuver menghindari tembakan

    Go() # Bergerak sesuai perintah
    IncrementTurnsSinceScan() # Hitung berapa banyak giliran sejak
    pemindaian

# Fungsi untuk menangani tembakan musuh
function OnHitByBullet():
    dodgeMode = true # Aktifkan mode dodge
    dodgeTurnsLeft = 15 # Tentukan durasi mode dodge

# Fungsi untuk melakukan manuver penghindaran tembakan
function PerformDodgeManeuver():
    if (dodgeTurnsLeft > 0):
        TurnRate = RandomTurn() # Acak arah perputaran
        dodgeTurnsLeft -= 1 # Kurangi sisa manuver
    else:
        dodgeMode = false # Matikan mode dodge

```

```

# Fungsi untuk menghindari dinding
function StayAwayFromWalls():
    if (X < MARGIN or X > ArenaWidth - MARGIN or Y < MARGIN or Y >
ArenaHeight - MARGIN):
        TurnRate = RandomBetween(-10, 10) # Putar sedikit untuk menghindari
dinding

# Fungsi untuk gerakan acak pada interval tertentu
function RandomMoveIfNeeded():
    randomMoveCounter += 1
    if (randomMoveCounter > RANDOM_MOVE_INTERVAL):
        DoRandomMove() # Gerakan acak
        randomMoveCounter = 0 # Reset counter

# Fungsi untuk gerakan acak
function DoRandomMove():
    angle = RandomBetween(0, 45) # Arah gerakan acak
    distance = RandomBetween(100, 300) # Jarak acak
    TurnRate = RandomBetween(-MaxTurnRate, MaxTurnRate) # Tentukan
kecepatan putaran
    SetForward(distance) # Bergerak maju

```

#### 4.1.4 Alternatif Solusi 4 – Pencarian Posisi Optimal (Bryan)

```

# Inisialisasi posisi dan variabel
Initialize position variables (X, Y)
Initialize movement flags (movingForward, random)
Initialize arena boundaries (MaxX, MaxY, MinX, MinY, CenterX, CenterY)

```



```

# Fungsi utama untuk menjalankan bot
function Run():
    # Set warna bot
    SetBodyColor(Pink)
    SetTurretColor(Red)
    SetRadarColor(Red)

    while (bot is running):
        # Cek posisi dan lakukan pergerakan
        CheckAndMove()

        # Putar turret secara acak untuk mengacak tembakan
        SetTurnRight(RandomBetween(-180, 180))
        WaitFor(TurnComplete)

# Fungsi untuk memeriksa dan melakukan pergerakan
function CheckAndMove():
    if (bot is near arena boundaries):
        MoveToCenter() # Pindah ke pusat arena
    else:
        MoveRandomly() # Pindah secara acak

# Fungsi untuk memindahkan bot ke pusat arena
function MoveToCenter():
    TurnToFaceTarget(CenterX, CenterY) # Menghadap ke pusat
    MoveForward(DistanceTo(CenterX, CenterY)) # Bergerak maju menuju
pusat

```

```

WaitFor(MoveComplete)

# Fungsi untuk melakukan pergerakan acak
function MoveRandomly():
    distance = RandomBetween(100, 200) # Jarak acak antara 100 dan 200

    if (movingForward):
        if (X + distance is near boundary):
            ReverseDirection() # Jika dekat dinding, mundur
        else:
            SetForward(distance) # Jika aman, maju
    else:
        if (X - distance is near boundary):
            ReverseDirection() # Jika dekat dinding, mundur
        else:
            SetBack(distance) # Jika aman, mundur

# Putar turret untuk mengacak arah tembakan
SetTurnRight(RandomBetween(-45, 45))

# Fungsi untuk membalikkan arah gerakan
function ReverseDirection():
    if (movingForward):
        SetBack(400) # Mundur 400 unit
        movingForward = false
    else:
        SetForward(400) # Maju 400 unit
        movingForward = true

```

## 4.2 Deskripsi Struktur Data, Fungsi, dan Prosedur

### 4.2.1 Struktur Data

Untuk mengimplementasikan strategi Zoning dan Manuver pada Ecus Bot, beberapa struktur data digunakan untuk mengelola keadaan dan pergerakan bot dengan efisien. Struktur data ini sangat penting untuk memastikan bahwa bot dapat beroperasi dengan baik dalam berbagai kondisi.

#### 1. Zona Area

Arena dibagi menjadi empat zona berdasarkan koordinat (x, y). Setiap zona memiliki batasan yang membatasi pergerakan bot. Zona ini digunakan untuk menentukan apakah bot berada di area yang aman atau perlu berpindah ke zona lain. Terdapat empat zona utama:

- a. Zona atas (MIN\_Y\_UPPER, MAX\_Y\_UPPER)
- b. Zona kanan (MIN\_X\_RIGHT, MAX\_X\_RIGHT)
- c. Zona bawah (MIN\_Y\_LOWER, MAX\_Y\_LOWER)
- d. Zona kiri (MIN\_X\_LEFT, MAX\_X\_LEFT)

#### 2. Posisi Bot

Variabel posisi (X, Y) digunakan untuk melacak lokasi bot saat ini di arena. Ini berguna untuk menentukan apakah bot berada dalam batas zona aman atau perlu bergerak menuju zona yang lebih strategis.

#### 3. Zona yang Dituju

Variabel `currentZone` digunakan untuk menyimpan zona tempat bot saat ini berada. Jika bot perlu berpindah ke zona lain, nilai `currentZone` akan diperbarui untuk menunjukkan zona yang lebih aman.

#### 4. Flag Status

##### a. `needToReposition`

Menandakan apakah bot perlu berpindah ke zona lain karena terlalu dekat dengan dinding atau musuh.

b. **movingForward**

Menunjukkan arah pergerakan bot. Jika bot bergerak mundur, maka flag ini akan diubah menjadi false.

c. **stuckCounter**

Digunakan untuk mendeteksi apakah bot terjebak di posisi tertentu (misalnya di sudut). Jika bot tidak bergerak dalam beberapa giliran berturut-turut, bot akan dianggap terjebak dan akan melakukan manuver untuk keluar dari situasi tersebut.

#### 4.2.2 Fungsi dan Prosedur

Berikut adalah beberapa fungsi dan prosedur utama yang digunakan untuk mengimplementasikan strategi Zoning dan Manuver pada Bot BJB:

1. **CheckIfStuck()**

Fungsi ini memeriksa apakah bot terjebak dengan mengukur jarak yang ditempuh bot antara dua giliran berturut-turut. Jika jarak yang ditempuh sangat kecil, bot dianggap terjebak dan fungsi `EmergencyUnstuck()` dipanggil untuk mengarahkan bot ke posisi yang lebih aman.

```
def CheckIfStuck():
    distance_moved = calculate_distance_moved(X,
lastX, Y, lastY)
    if distance_moved < 2.0:
        stuck_counter += 1
        if stuck_counter >= STUCK_THRESHOLD:
            EmergencyUnstuck() # Mengambil
tindakan darurat jika terjebak
```

2. **EmergencyUnstuck()**

Fungsi ini digunakan untuk mengambil tindakan darurat jika bot terjebak. Bot akan diarahkan untuk menghindari dinding atau sudut

arena dengan cara yang lebih aman, seperti bergerak ke arah pusat arena.

```
def EmergencyUnstuck():  
    escape_from_walls_or_corners()
```

### 3. MoveToAllowedZone()

Fungsi ini memastikan bot bergerak ke zona yang lebih aman jika berada terlalu dekat dengan dinding atau berada di zona berisiko tinggi. Bot akan bergerak menuju zona tengah yang lebih aman.

```
def MoveToAllowedZone():  
    move_to_zone(currentZone)
```

### 4. RandomizedMovement()

Fungsi ini memberikan pergerakan acak pada bot. Jika bot berada di zona yang aman, bot akan bergerak acak dengan arah dan jarak yang ditentukan secara acak. Jika bot berada dekat dengan dinding, fungsi ini akan memicu manuver penghindaran.

```
def RandomizedMovement():  
    if IsNearWall():  
        AvoidWall() # Menghindari dinding dengan  
cara yang aman  
    else:  
        random_move()
```

### 5. IsNearWall()

Fungsi ini memeriksa apakah bot berada dekat dengan dinding arena. Jika bot terlalu dekat, maka bot akan berusaha menghindari dinding dengan bergerak ke arah yang lebih aman.

```
def IsNearWall():
    if X < WALL_SAFETY_DISTANCE or X > ArenaWidth -
WALL_SAFETY_DISTANCE:
        return True
    elif Y < WALL_SAFETY_DISTANCE or Y >
ArenaHeight - WALL_SAFETY_DISTANCE:
        return True
    return False
```

## 6. ScanningMode()

Fungsi ini digunakan untuk memindai musuh di sekitar bot. Bot akan terus-menerus memindai area sekitar dan menyesuaikan arah senjata jika musuh terdeteksi.

```
def ScanningMode():
    if turn_to_scan_condition():
        scan_for_enemy()
```

## 7. AvoidWall()

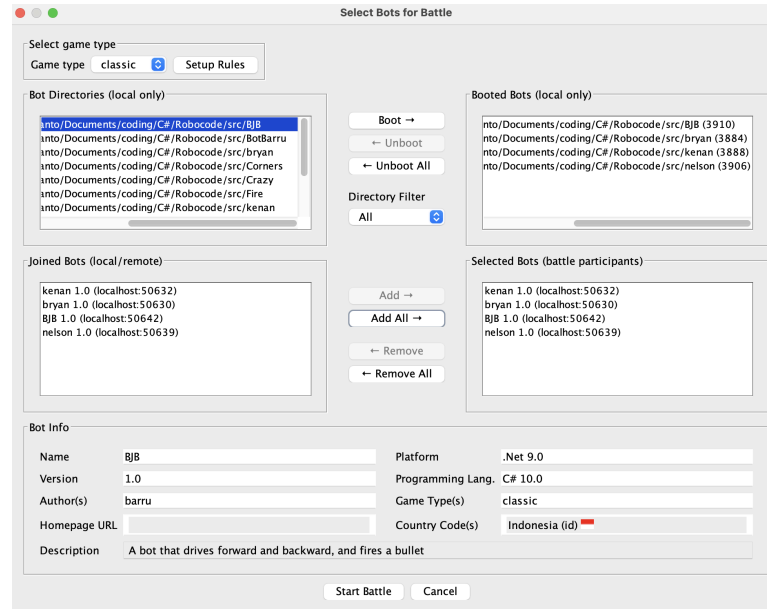
Fungsi ini sangat penting untuk memastikan bot selalu berada di zona yang aman dan tidak terjebak dekat dinding. Jika bot terdeteksi dekat dengan dinding, bot akan mengubah arah pergerakan atau melarikan diri ke zona yang lebih aman.

```
def AvoidWall():
    if IsNearWall():
        random_turn_and_move()
```

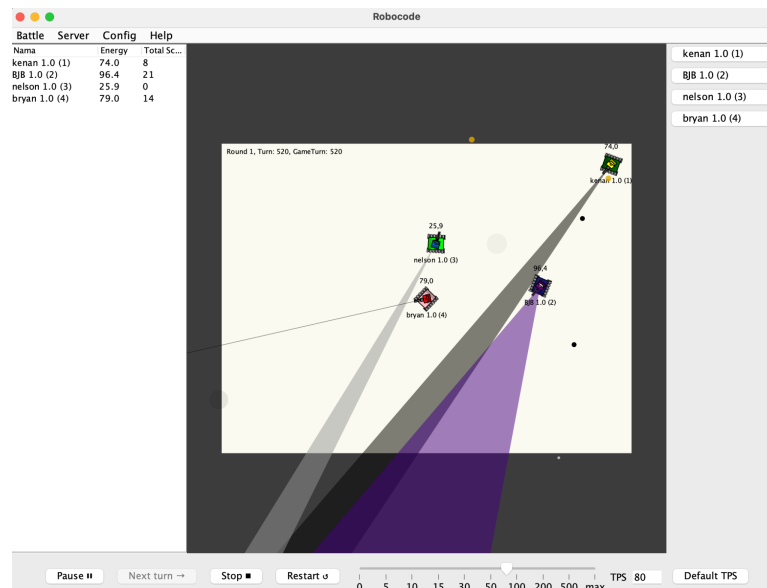
## 4.3 Pengujian Bot Utama dan Bot Asisten

### 4.3.1 Pengujian 1

#### 1. Booting



#### 2. Playing

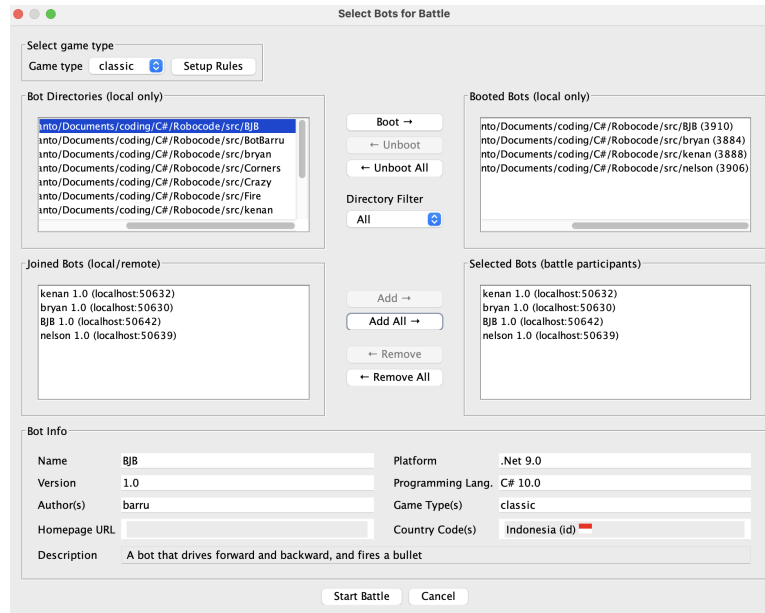


### 3. Leaderboard

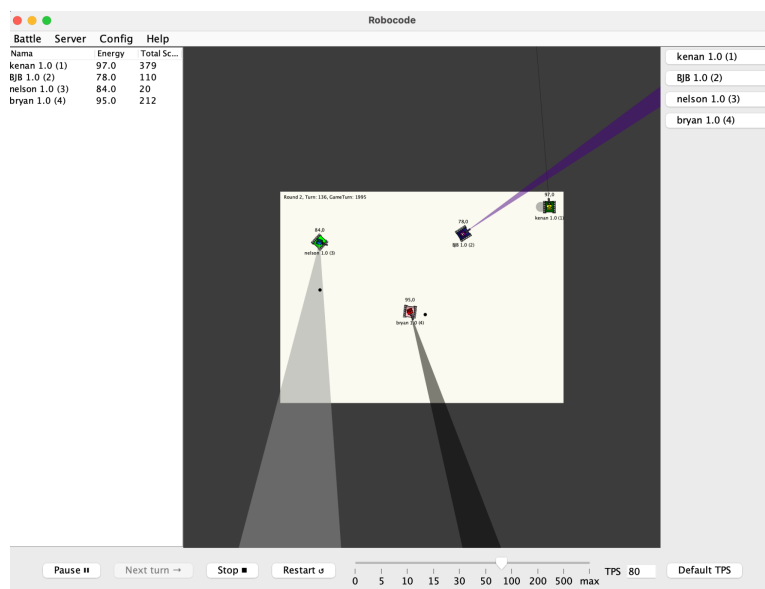
Results for 10 rounds											
Rank	Name	Total Score	Survival	Surv. Bonus	Bullet Dmg.	Bullet Bon...	Ram Dmg.	Ram Bonus	1sts	2nds	3rds
1	BJB 1.0	563	300	30	208	21	4	0	2	0	1
2	bryan 1.0	509	250	30	204	16	8	0	1	0	2
3	kenan 1.0	404	200	0	188	0	16	0	0	3	0
4	nelson 1.0	91	0	0	78	0	13	0	0	0	0

#### 4.3.2 Pengujian2

### 1. Booting



### 2. Playing





### 3. Leaderboard

Results for 10 rounds											
Rank	Name	Total Score	Survival	Surv. Bonus	Bullet Dmg.	Bullet Bon...	Ram Dmg.	Ram Bonus	1sts	2nds	3rds
1	bryan 1.0	634	250	0	334	43	6	0	2	0	1
2	BJB 1.0	599	300	30	246	15	7	0	0	2	1
3	kenan 1.0	540	250	30	236	20	4	0	1	1	1
4	nelson 1.0	75	50	0	20	0	5	0	0	0	0

### 4.3.3 Pengujian 3

#### 1. Booting

Select Bots for Battle

Select game type  
Game type: classic

Bot Directories (local only)

into/Documents/coding/C#/Robocode/src/BJB  
into/Documents/coding/C#/Robocode/src/BotBaru  
into/Documents/coding/C#/Robocode/src/bryan  
into/Documents/coding/C#/Robocode/src/Corners  
into/Documents/coding/C#/Robocode/src/Crazy  
into/Documents/coding/C#/Robocode/src/Fire  
into/Documents/coding/C#/Robocode/src/kenan

Booted Bots (local only)

into/Documents/coding/C#/Robocode/src/BJB (3910)  
into/Documents/coding/C#/Robocode/src/bryan (3884)  
into/Documents/coding/C#/Robocode/src/kenan (3888)  
into/Documents/coding/C#/Robocode/src/nelson (3906)


Joined Bots (local/remote)

kenan 1.0 (localhost:50632)  
bryan 1.0 (localhost:50630)  
BJB 1.0 (localhost:50642)  
nelson 1.0 (localhost:50639)

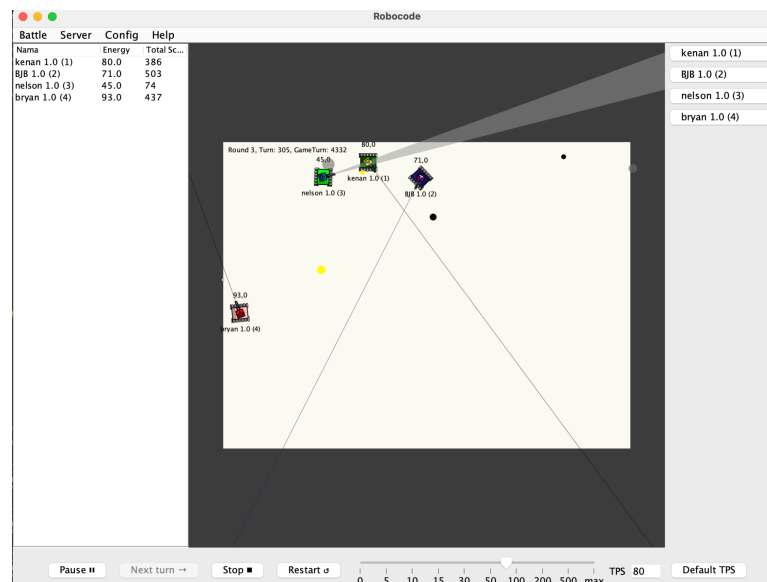
Selected Bots (battle participants)

kenan 1.0 (localhost:50632)  
bryan 1.0 (localhost:50630)  
BJB 1.0 (localhost:50642)  
nelson 1.0 (localhost:50639)

Bot Info

Name: BJB Platform: .Net 9.0  
Version: 1.0 Programming Lang: C# 10.0  
Author(s): barru Game Type(s): classic  
Homepage URL: Country Code(s): Indonesia (id)   
Description: A bot that drives forward and backward, and fires a bullet

#### 2. Playing



### 3. Leaderboard

Results for 10 rounds											
Rank	Name	Total Score	Survival	Surv. Bonus	Bullet Dmg.	Bullet Bon...	Ram Dmg.	Ram Bonus	1sts	2nds	3rds
1	BJB 1.0	587	300	30	238	14	5	0	2	0	1
2	kenan 1.0	584	300	0	254	23	6	0	0	3	0
3	bryan 1.0	422	200	30	172	13	7	0	1	0	1
4	nelson 1.0	182	50	0	94	0	38	0	0	0	1

#### 4.4 Analisis Hasil Pengujian

Param	Pengujian 1				Pengujian 2				Pengujian 3			
	BJB	Kenan	Bryan	Nelson	BJB	Kenan	Bryan	Nelson	BJB	Kenan	Bryan	Nelson
<b>Total Score</b>	<b>563</b>	404	509	91	599	540	<b>634</b>	75	<b>587</b>	584	422	182
<b>Survival</b>	300	200	250	0	300	250	250	50	300	300	200	50
<b>Surv. Bonus</b>	30	0	30	0	30	30	0	0	30	0	30	0
<b>Bullet Dmg.</b>	208	188	204	78	246	236	334	20	238	254	172	94
<b>Bullet Bonus</b>	21	0	16	0	15	20	43	0	14	23	13	0
<b>Ram Dmg.</b>	4	16	8	13	7	4	6	5	5	6	7	38
<b>Ram Bonus</b>	0	0	0	0	0	0	0	0	0	0	0	0

Dari hasil pengujian yang dilakukan, kemenangan masing-masing bot dipengaruhi oleh strategi unik yang mereka terapkan dalam arena pertempuran.

Bot “BJB” memiliki strategi zoning dan manuver yang sangat efektif. Ia berpindah-pindah secara acak tetapi tetap memahami zonanya dengan baik, sehingga selalu mengarahkan senjatanya ke area tengah arena untuk meningkatkan

akurasi tembakan. Selain itu, BJB **sulit dilacak** dan ditarget oleh lawan karena pergerakannya yang tidak terprediksi. Keunggulan lain dari BJB adalah **efisiensinya dalam melakukan scanning**. BJB mengeliminasi pemindaian yang tidak perlu, seperti mendeteksi ke arah tembok atau area kosong, sehingga lebih cepat menemukan target yang benar-benar berada dalam zona serangannya. Dengan demikian, ia bisa lebih cepat dan akurat dalam mendeteksi serta menyerang musuh dibandingkan dengan bot lain yang masih membuang waktu untuk melakukan scan ke area yang tidak relevan.

Bot BJB sempat kalah dan Bot Bryan menang saat pertandingan 2 karena bot BJB masih memiliki kekurangan yaitu bot tersebut terkadang **masih sering tertabrak dinding dan metode pengatasannya minim**, sedangkan Bot Bryan kemampuannya menghindari serangan dan tabrakan. Ia memiliki pola gerakan yang **menjaga jaraknya dari dinding dan selalu kembali ke tengah arena setelah mencapai batas aman**. Strategi ini membuatnya lebih sulit untuk terkena serangan langsung. Selain itu, dalam kondisi pertarungan dengan jumlah bot yang lebih sedikit, Bot Bryan mampu **melacak bot yang pernah terdeteksi oleh sensor atau yang pernah menabraknya**. Kemampuannya dalam fokus pada target tertentu membuatnya lebih efektif dalam pertarungan dengan sedikit lawan, karena ia dapat secara konsisten menargetkan musuh yang telah dipindainya. Namun masih kalah dikarenakan scanning oleh Bot Bryan **tidak memiliki efisiensi** seperti Bot BJB.

Kesimpulan dari hasil pengujian ini, BJB cenderung unggul karena **keakuratan dalam membidik, efisiensi dalam scanning, dan pola pergerakan yang sulit diprediksi**. Di sisi lain, Bot Bryan sempat menang dikarenakan **lebih unggul dalam bertahan dan membidik terus bot yang di scan** (sehingga baik dalam pertarungan dengan sedikit lawan).

## **Bab V**

### **Penutup**

#### **5.1 Kesimpulan**

Berdasarkan seluruh pengujian yang dilakukan terhadap berbagai bot yang menggunakan algoritma greedy dalam permainan Robocode Tank Royale, ada beberapa poin penting yang dapat disimpulkan:

##### **1. Strategi Greedy yang Berbeda Memengaruhi Kinerja Bot**

Setiap bot yang diuji mengadopsi strategi greedy yang berbeda, dan hal ini berpengaruh langsung pada hasil dan kinerja mereka di arena pertempuran. Bot "BJB" menonjol dalam hal kecepatan, efisiensi, dan ketepatan dalam mendeteksi serta menyerang musuh. BJB mengoptimalkan pergerakan acaknya untuk menyulitkan musuh dalam menargetkan dirinya, sekaligus mengurangi waktu yang terbuang dalam pemindaian.

Sementara itu, Bot "Bryan" lebih unggul dalam bertahan dan menghindari tabrakan, serta mampu menjaga jarak yang lebih baik dengan musuh. Namun, efisiensi dalam pemindaian pada Bryan masih kalah dibandingkan dengan BJB, yang mempengaruhi kecepatan dan ketepatannya dalam menyerang.

##### **2. Kelemahan dan Perbaikan yang Diperlukan**

Bot BJB, meskipun strateginya efektif, memiliki kekurangan dalam penghindaran tabrakan dengan dinding. Bot ini perlu ditingkatkan dalam menghindari dinding dan memanfaatkan ruang arena secara lebih optimal. Di sisi lain, Bot Bryan lebih unggul dalam bertahan dan fokus pada musuh, namun efisiensi dalam pemindaian dan ketepatan dalam penargetan perlu diperbaiki untuk meningkatkan performa.

##### **3. Efisiensi Scanning sebagai Faktor Penentu**

Efisiensi dalam scanning dan deteksi musuh sangat mempengaruhi kinerja bot. BJB memiliki pemindaian yang efisien, mampu fokus pada area

pertempuran yang relevan, sementara Bryan menghabiskan terlalu banyak waktu untuk memindai area yang tidak relevan.

#### **4. Pola Pergerakan yang Adaptif**

BJB memiliki keunggulan dengan pola pergerakan acak yang sulit diprediksi oleh musuh, memberikan keuntungan dalam menghindari serangan dan memusatkan tembakan ke area yang strategis. Sebaliknya, Bryan lebih terfokus pada penghindaran dan menjaga jarak dengan musuh serta dinding.

#### **5. Survival dan Skor Akhir**

Secara keseluruhan, BJB memiliki skor akhir yang lebih tinggi dibandingkan bot lainnya. Keberhasilan BJB terutama terlihat dari kemampuannya bertahan hidup lebih lama dan mendapatkan bonus dari tembakan yang efektif. Sementara itu, meskipun Bot Bryan menunjukkan kekuatan dalam bertahan, ia masih kalah dalam aspek kecepatan deteksi dan tembakan.

## **5.2 Saran**

Berdasarkan kesimpulan yang telah disampaikan, ada beberapa saran yang dapat diterapkan untuk meningkatkan kinerja bot dalam Robocode Tank Royale:

### **1. Perbaikan Penghindaran Tabrakan pada BJB**

Untuk meningkatkan performa BJB, perlu ditambahkan algoritma penghindaran tabrakan yang lebih cerdas, terutama dalam menghindari dinding. Menambahkan mekanisme deteksi dini terhadap batas arena dan mengubah arah secara otomatis sebelum bot terlalu dekat dengan dinding akan membuat BJB lebih efektif dalam memanfaatkan ruang arena.

### **2. Peningkatan Efisiensi Scanning pada Bot Bryan**

Untuk Bot Bryan, penting untuk meningkatkan efisiensi dalam pemindaian musuh. Bryan bisa diperkenalkan dengan algoritma scanning yang lebih terfokus pada area pertempuran yang lebih relevan dan menghindari waktu yang terbuang pada area yang tidak penting. Fokus pada pemindaian musuh

yang telah terdeteksi sebelumnya atau berdasarkan prediksi arah musuh juga bisa meningkatkan responsivitas bot.

### **3. Pengembangan Algoritma Serangan yang Cerdas**

Kedua bot dapat meningkatkan algoritma tembakannya dengan memanfaatkan analisis jarak dan kecepatan musuh. Penambahan algoritma tembakan prediktif yang memperhitungkan pergerakan musuh akan meningkatkan akurasi tembakan, memberi bot keunggulan dalam serangan.

### **4. Pengujian dan Iterasi Lebih Lanjut**

Pengujian lebih lanjut perlu dilakukan dengan variasi skenario yang lebih luas dan dengan banyak lawan untuk memastikan strategi yang diterapkan dapat bekerja dengan baik dalam berbagai kondisi pertempuran. Uji coba dengan bot-bot lain yang menggunakan strategi berbeda juga akan memberikan wawasan tentang keunggulan relatif dari bot yang dikembangkan. Mencoba pendekatan hybrid yang menggabungkan keunggulan BJB dan Bryan dapat menciptakan bot yang lebih fleksibel dalam menghadapi berbagai situasi.

### **5. Adaptasi terhadap Kondisi Arena yang Berubah**

Bot perlu dilengkapi dengan algoritma yang dapat menyesuaikan strategi dengan kondisi arena yang berubah-ubah. Hal ini penting, terutama saat ada perubahan cepat dalam posisi musuh atau jika arena menjadi lebih sempit, yang mempengaruhi pergerakan dan strategi bot.

### **6. Pemrograman Paralel atau Peningkatan Kecepatan Respons**

Dengan perkembangan teknologi, pemrograman paralel bisa digunakan untuk meningkatkan kecepatan respons bot dalam setiap giliran. Pemrosesan paralel akan mempercepat pengolahan informasi dan memungkinkan bot untuk merespons perubahan keadaan dalam arena dengan lebih cepat.

## Lampiran

Link Github Repository : <https://s.hmif.dev/GithubBJB>  
Link Video Youtube : <https://s.hmif.dev/YoutubeBJB>

## Daftar Pustaka

Robocode Tank Royale. (n.d.). *GitHub Repository*. Retrieved from <https://github.com/robocode-dev/tank-royale>

Robocode Tank Royale Docs. (n.d.). *Complete Documentation of Tank Royale*. Retrieved from <https://robocode-dev.github.io/tank-royale/>

C# .Net. (n.d.). *Official .NET Website*. Retrieved from <https://dotnet.microsoft.com/en-us/>

Learn to code C#. (n.d.). *C# Tutorials from Microsoft*. Retrieved from <https://dotnet.microsoft.com/en-us/learn/code>

C# Language Documentation. (n.d.). *C# Documentation from Microsoft*. Retrieved from <https://learn.microsoft.com/en-us/dotnet/csharp/>

What is an API? (n.d.). *AWS Documentation on API*. Retrieved from <https://aws.amazon.com/what-is/api/>