

**LAPORAN TUGAS BESAR 1**  
**IF2211 STRATEGI ALGORITMA**  
**PEMANFAATAN ALGORITMA *GREEDY* DALAM PEMBUATAN**  
**BOT PERMAINAN ROBOCODE TANK ROYALE**



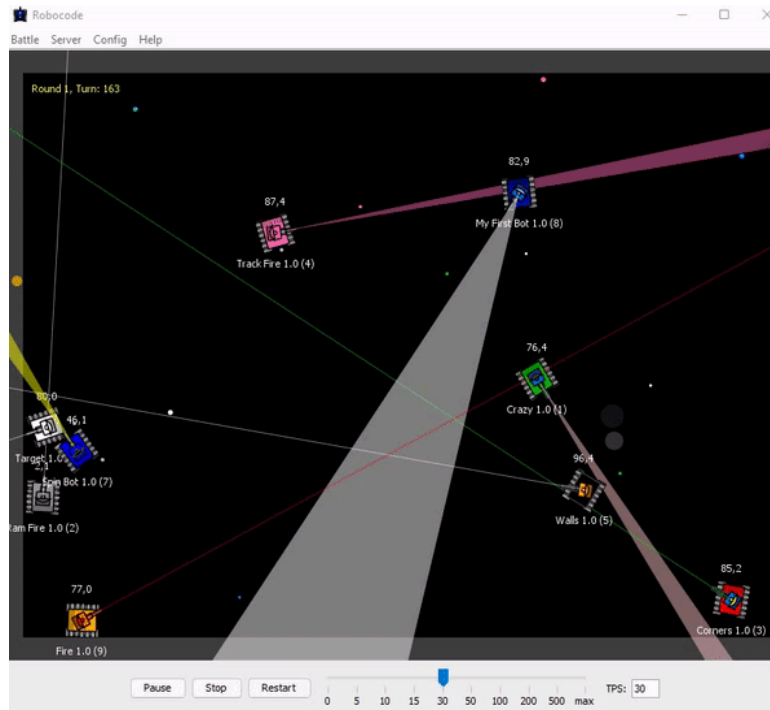
Disusun Oleh:  
Kelompok 17 - Popo Shiroyo

Wardatul Khoiroh	13523001
Indah Novita Tangdililing	13523047
Kefas Kurnia Jonathan	13523113

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

# BAB I

## DESKRIPSI MASALAH



Gambar 1 - Robocode Tank Royale

Robocode adalah permainan pemrograman yang bertujuan untuk membuat kode bot dalam bentuk tank virtual untuk berkompetisi melawan bot lain di arena. Pertempuran Robocode berlangsung hingga bot-bot bertarung hanya tersisa satu seperti permainan Battle Royale, karena itulah permainan ini dinamakan Tank Royale. Robocode Tank Royale adalah evolusi/versi 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, kami diminta untuk membuat sebuah bot yang nantinya akan dipertandingkan. Tentunya 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.

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.

## 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.

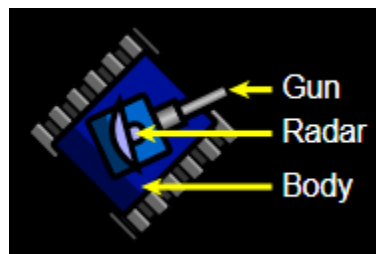
#### 6. Tabrakan

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

Secara umum, tubuh tank terdiri dari 3 bagian:



Gambar 2 - Bagian Tubuh Tank dalam Robocode

*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

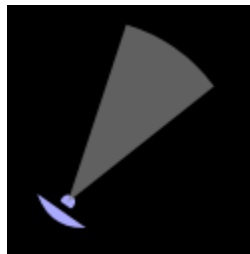
Seperti yang disebutkan sebelumnya, bagian tubuh, turret (meriam), dan radar dapat berputar secara independen satu sama lain. Jika turret atau radar tidak diputar, maka keduanya akan mengarah ke arah yang sama dengan tubuh bot. Setiap bagian tubuh memiliki kecepatan putar yang berbeda. Radar adalah bagian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

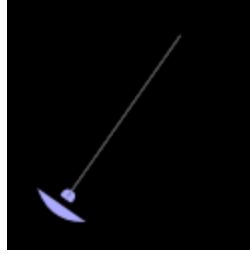
Aspek penting dalam Robocode adalah memindai bot musuh menggunakan radar. Radar dapat mendeteksi bot dalam jangkauan hingga 1200 piksel. Musuh yang berada lebih dari 1200 piksel dari bot tidak dapat terdeteksi atau dipindai oleh radar.

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.



Gambar 3 - Contoh pemindaian pada Robocode

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.



Gambar 4 - Contoh pemindaian pada Robocode yang tidak bergerak

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

## 11. Skor

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

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

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

## BAB II

### LANDASAN TEORI

#### 2.1. Algoritma *Greedy*

Algoritma *greedy* adalah metode sederhana untuk memecahkan persoalan mencari solusi optimal. Terdapat dua macam persoalan optimasi yaitu maksimasi dan minimasi. Kata *Greedy* memiliki arti rakus atau tamak atau loba. Algoritma *greedy* ini memilih keputusan yang terbaik pada setiap langkahnya dimana tidak bisa melihat apa yang terjadi kedepannya. Namun, algoritma ini tidak dapat mundur ke langkah sebelumnya. Keputusan yang dipilih pada tiap langkah disebut optimum lokal. Sedangkan solusi dari permasalahan yang dicari disebut optimum global. Enam elemen dari algoritma *greedy* yaitu:

1. Himpunan kandidat (C). Adalah kandidat yang akan dipilih pada setiap langkah seperti simpul di dalam graf atau koin.
2. Himpunan solusi (S). Berisi kandidat yang sudah dipilih pada langkah sebelumnya.
3. Fungsi solusi. Berfungsi menentukan apakah himpunan kandidat yang dipilih (himpunan solusi) sudah memberikan solusi dari permasalahannya.
4. Fungsi seleksi (selection function). Berfungsi memilih kandidat berdasarkan strategi *greedy* yang bersifat heuristik.
5. Fungsi kelayakan (feasible). Berfungsi memeriksa apakah kandidat yang dipilih dapat dimasukkan ke dalam himpunan solusi atau tidak.
6. Fungsi obyektif. Fungsi untuk memaksimumkan atau meminimumkan.

Perlu diperhatikan bahwa optimum global belum tentu memberikan solusi terbaik, bisa jadi hanya merupakan solusi sub-optimum atau pseudo-optimum. Hal ini dikarenakan algoritma *greedy* tidak dapat memilih semua kemungkinan solusi yang ada sebagaimana metode *exhaustive search*. Sekalipun dapat menghasilkan solusi optimal, perlu dibuktikan keoptimalannya secara matematis. Namun hal ini menjadi tantangan tersendiri dibandingkan memperlihatkan bahwa algoritma *greedy* tidak optimal yaitu dengan *counterexample*. Berikut beberapa contoh persoalan yang dapat diselesaikan dengan algoritma *greedy*:

1. Persoalan penukaran uang (coin exchange problem).
2. Persoalan memilih aktivitas (activity selection problem).
3. Minimisasi waktu di dalam sistem.
4. Persoalan *knapsack* (*knapsack problem*).
5. Penjadwalan job dengan tenggat waktu (*job scheduling with deadlines*).
6. Pohon merentang minimum (*minimum spanning tree*).
7. Lintasan terpendek (*shortest path*).
8. Kode Huffman.
9. Pecahan Mesir.

Berikut beberapa kelebihan dari algoritma *greedy*:

1. Mudah untuk diimplementasikan karena tidak memerlukan struktur data yang kompleks.
2. Pada beberapa persoalan, waktu eksekusi lebih cepat dibandingkan algoritma lain.
3. Cocok untuk permasalahan real-time yang membutuhkan keputusan cepat.
4. Tidak membutuhkan memori tambahan yang besar dibandingkan yang lain.

Berikut beberapa kekurangan dari algoritma *greedy*:

1. Tidak dapat digunakan untuk semua jenis permasalahan terutama yang membutuhkan evaluasi menyeluruh terhadap semua kemungkinan solusi.
2. Rentan terhadap jebakan lokal.
3. Tidak selalu memberikan solusi optimal.
4. Pada beberapa permasalahan, hasil hanya mendekati solusi optimal.

## **2.2. Bahasa Pemrograman C#**

C# adalah bahasa pemrograman modern, berorientasi objek, dan berbasis .NET Framework yang dikembangkan oleh Microsoft. .NET Framework adalah perantara yang membantu aplikasi dengan bahasa pemrograman yang didukung dapat berkomunikasi dengan sistem operasi yang digunakan oleh komputer mayoritas. Bahasa pemrograman C# banyak digunakan untuk membangun berbagai macam aplikasi seperti aplikasi web, desktop, aplikasi zune, aplikasi permainan, dan jenis aplikasi lainnya. Berikut beberapa sintaks dasar dari C#:

1. Fungsi `console.WriteLine()` untuk menampilkan informasi ke layar.
2. Fungsi `console.ReadLine()` untuk meminta input user bertipe string.
3. Perintah `if()` untuk memberikan syarat pada satu atau sekumpulan statement.
4. *Loop statement*.

C# memiliki lima struktur dasar yaitu:

1. *Resource* atau *library*.
2. *Namespace*.
3. Nama *Class*.
4. Deklarasi *Method*.
5. *Method* atau *Command*.

Terdapat ketentuan dalam pendefinisian variabel yaitu awalan variabel tidak boleh angka, variabel tidak boleh mengandung karakter khusus seperti \$, # %, +, -, dan sebagainya, serta tidak boleh ada variabel yang memiliki spasi. Contoh variabel yang benar yaitu `nama_`, `aku1`, `status_palsu`.



## 2.3. Cara Kerja Program Secara Umum

Bot pada Robocode Tank Royale adalah entitas yang bergerak, mendeteksi musuh, dan menembak. Program bot ini menggunakan model berbasis event (event-driven model), dimana bot merespons berbagai kejadian yang terjadi di arena robocode. Kejadian di arena seperti mendeteksi lawan, menabrak lawan, menembak lawan, menabrak dinding, dan lainnya.

### 2.3.1. Bagaimana Bot Melakukan Aksinya

Bot dalam Robocode Tank Royale memiliki beberapa aksi utama yang dilakukan dalam pertandingan yaitu:

1. Pergerakan Bot: bot dapat bergerak maju (`Forward(distance)`), mundur (`Backward(distance)`), belok kanan (`TurnRight(angle)`), dan belok kiri (`TurnLeft(angle)`).
2. Pemindaian Lawan: bot memiliki radar untuk mendeteksi keberadaan lawan di arena sehingga saat menemukan lawan, `OnScannedBot()` akan dipicu. Informasi yang didapat berupa jarak bot lawan (`distance`), sudut relatif terhadap bot kita (`bearing`), posisi X dan Y, serta kecepatan dan arah gerak lawan.
3. Menembak Lawan: bot dapat menembak dengan metode `Fire(power)` dengan power adalah kekuatan tembakan, semakin besar nilai power maka semakin besar pula damage yang diberikan namun semakin banyak pula energi bot yang dihabiskan.
4. Menghindari Bahaya: bot dapat menghindari tembakan lawan dan tabrakan dengan dinding atau bot lain agar tidak kehilangan energi.

### 2.3.2 Mengimplementasikan Algoritma Greedy ke Dalam Bot

1. Menembak dengan Agresif: setiap kali lawan terdeteksi, bot langsung menembak dengan kekuatan maksimal apabila jarak lawan cukup dekat.
2. Bergerak dengan Pola Spiral atau Zig-Zag: pergerakan random untuk menghindari tembakan lawan.
3. Menghindari Tabrakan dengan Dinding dan Bot Lain: saat bot menabrak dinding atau lawan, bot segera berbelok agar menghindari kehilangan energi akibat benturan.

### 2.3.3 Bagaimana Menjalankan Bot

Untuk menjalankan bot pada gim *Robocode* sangatlah mudah. Untuk memahami bagaimana cara menjalankan bot dapat diperhatikan sebagai berikut :

1. Jalankan GUI Robocode dengan menuliskan:

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

2. Pada menu *Config* pilih *Boot Root Directories*, kemudian klik *add* dan masukkan folder yang berisi bot-bot yang ada.

3. Pada menu *Battle* pilih *Start Battle*, maka list bot-bot yang ada akan terdaftar pada *Bot Directories (local only)*.
4. Pilih Bot mana saja yang ingin di-*boot* ke dalam permainan. Bot yang sukses di-*boot* akan muncul pada *Joined Bots (local/remote)*.
5. Dari bot yang sudah ada pada *Joined Bots*, klik *add* untuk bot-bot yang akan dipertandingkan.
6. Jika sudah, klik *Start Battle* untuk memulai pertandingan. Selamat menyaksikan kemenangan PopoBot!

## BAB III

### APLIKASI STRATEGI *GREEDY*

#### 3.1 *Mapping* Persoalan

Komponen	Penjelasan
Himpunan Kandidat	Dalam permasalahan ini, himpunan kandidat terdiri atas semua kemungkinan variasi aksi atau <i>command</i> yang dapat dilakukan oleh bot Robocode. <i>Command</i> aksi tersebut yaitu berupa kombinasi dari perintah FORWARD, BACK, TURN RIGHT, TURN LEFT, dan FIRE.
Himpunan Solusi	Himpunan solusi adalah subset dari himpunan kandidat, yaitu kombinasi aksi yang valid dan dapat dijalankan secara efektif dalam pertandingan. Himpunan ini mencakup strategi yang memungkinkan bot bertahan hidup lama, melakukan tabrakan serangan, atau menembak lawan dengan akurasi.
Fungsi Solusi	Menentukan bagaimana kombinasi aksi dalam himpunan solusi diterapkan dalam permainan. Ini mencakup algoritma yang mengontrol pergerakan, penghindaran, penargetan musuh, serta pengambilan keputusan menembak berdasarkan data dari radar dan status robot.
Fungsi Seleksi	Memilih solusi terbaik dari himpunan solusi berdasarkan performa bot buatan dalam pertempuran. Seleksi dilakukan berdasarkan jumlah kemenangan, akurasi tembakan, jumlah damage yang diberikan, atau jumlah damage yang diterima saat melawan bot-bot sampel dalam <i>trial and error</i> .
Fungsi Kelayakan	Menentukan apakah suatu strategi atau kombinasi aksi dalam himpunan solusi layak digunakan dalam pertandingan. Suatu solusi dianggap layak jika dapat digunakan hingga akhir ronde dan tidak mengalami error atau ketidaksesuaian gerakan dengan kode.

Komponen	Penjelasan
Fungsi Objektif	<p>Menentukan tujuan utama yang ingin dicapai oleh bot dalam pertandingan. Tujuan ini dapat bervariasi tergantung pada strategi yang diterapkan. Beberapa kemungkinan fungsi objektif yang dapat digunakan adalah:</p> <ul style="list-style-type: none"> <li>- Bot memenangkan permainan dengan berusaha untuk bertahan hidup lebih lama dan menghindari serangan lawan menggunakan strategi evasive movement.</li> <li>- Bot memenangkan permainan dengan berusaha menembak akurat bot lawan dengan strategi penembakan yang optimal.</li> <li>- Bot memenangkan permainan dengan berusaha menyerang bot lawan dari dekat dan menabrakkan diri..</li> </ul>

## 3.2 Alternatif Solusi

### 3.2.1 Alternatif Solusi 1 : Popo-Bot

Pada Popo-Bot ini, menggunakan pendekatan greedy terhadap pergerakan untuk menghindari tembakan musuh serta memaksimalkan skor tembakan. Strategi ini bertumpu pada **gerakan spiral** yang semakin melebar hingga batas tertentu (limit). Heuristik yang digunakan adalah kombinasi antara greedy terhadap gerakan dan greedy terhadap serangan. Greedy terhadap serangan yang dimaksud yaitu saat melihat musuh, bot akan langsung menembak tanpa mempertimbangkan jarak atau energi lawan (sehingga mempercepat waktu dalam berfikir). Berikut langkah-langkah greedynya:

- Bot akan bergerak terus menerus dalam pola spiral dengan jarak yang semakin melebar sehingga dapat menghindari tembakan lawan. Jarak gerakan dan sudut pergerakannya bertambah secara bertahap sehingga bot tetap bergerak namun tidak mudah diprediksi.
- Jarak pergerakan spiral diberi maksimal yaitu 500 agar bot tetap stabil pergerakannya dan kembali ke masa kejayaannya. Saat jarak terlalu lebar, bot akan bergerak spiral terlalu jauh sehingga hanya terlihat seperti memutar arena. Hal ini tidak efektif untuk bot karena dapat sering tertembak oleh bot lain. Oleh karena itu, perlu reset kembali ke awal setelah jarak mencapai maksimum.
- Pada tiap langkah pengambilan keputusan, bot ini mempertimbangkan musuh dengan menscan musuh. Apabila bot melihat musuh, bot akan langsung menembak dengan kekuatan maksimal tanpa mempertimbangkan jarak karena bot mengasumsikan bahwa menembak selalu menguntungkan ketiga target terlihat. Bot tidak mempertimbangkan jarak karena untuk meningkatkan efisiensi waktu berfikir sehingga dapat memanfaatkan

waktu tiap round sebaik mungkin, tidak hanya diam untuk berfikir berapa jaraknya dengan musuh.

- d. Pada tiap langkah pengambilan keputusan, bot juga akan menghindari saat bertabrakan dengan musuh yaitu memutar arah dan maju sehingga mencegah penahanan posisi dan terkena tembakan.

### 3.2.2 Alternatif Solusi 2 : Shi-Bot

Pada Shi-Bot ini, menggunakan pendekatan greedy terhadap pergerakan acak dengan **pola zig-zag** sehingga sulit diprediksi oleh bot lain. Hal ini dapat membantu terhindar dari tembakan musuh dan tetap bergerak. Heuristik yang digunakan yaitu greedy terhadap pergerakan acak dengan pola zig-zag dan greedy terhadap serangan dengan saat melihat musuh bot langsung menembak tanpa mempertimbangkan aspek lain. Berikut langkah-langkah greedynya:

- a. Bot akan bergerak maju atau mundur sejauh 1000 unit sehingga memungkinkan bot menjelajahi arena dan menghindari tembakan musuh. Setelah itu, bot akan berbelok dalam sudut zig-zag sebesar 145 derajat. Pergantian arah ini membuat risiko bot terkena tembakan berkurang (karena saat bot lain melihat bot ini akan ditembak secara lurus sehingga perlu berputar arah).
- b. Bot akan terus dalam pola zig-zag hingga mencapai batas maksimal yaitu 3 kali. Lalu, bot akan mengubah arah maju/mundur agar pergerakannya terus terjadi namun lebih acak dan tidak hanya maju dalam pola berulang. Contoh: awalnya bergerak maju berubah menjadi mundur.
- c. Setiap pengambilan keputusan dalam tiap langkah, bot melakukan belokan berlawanan dengan arah belok yang sebelumnya (bergantian). Hal ini bertujuan membuat gerakan lebih tidak beraturan sehingga musuh sulit mengunci tembakannya.
- d. Pada tiap langkah pengambilan keputusan, bot akan menembak dengan kekuatan penuh setiap kali mendeteksi musuh karena tiap kesempatan ini dianggap menguntungkan.
- e. Pada tiap langkah pengambilan keputusan, bot juga akan langsung berbelok 30 derajat dan maju 50 unit saat menabrak tembok sehingga bot tidak terjebak pada posisi sudut arena (risiko tertembak lawan lainnya besar). Selain itu, bot juga akan berbelok saat bertabrakan dengan musuh sehingga bot lawan tidak dapat menahannya di posisi tetap.

### 3.2.3 Alternatif Solusi 3 : Ro-Bot

Pada Ro-Bot ini, menggunakan pendekatan greedy terhadap jarak dengan musuh sehingga dapat menghindari pertempuran jarak dekat yang berbahaya. Heuristik yang digunakan yaitu greedy terhadap keselamatan dengan bot selalu **menghindari musuh** jika terlalu dekat (agar terhindar dari tabrakan maupun tembakannya) dan greedy terhadap serangan dengan menembak setiap kali melihat musuh namun tetap menjaga jarak. Berikut langkah-langkah greedynya:

- a. Bot akan terus bergerak maju sejauh 400 unit sehingga bot tidak diam di satu tempat dan mengurangi kemungkinan terkena tembakan lawan.
- b. Pada tiap pengambilan keputusan, bot akan menembak langsung dengan kekuatan penuh saat melihat musuh dengan fokus utama bertahan.
- c. Pada tiap langkah pengambilan keputusan, apabila jarak ke musuh lebih kecil dari 400 unit, bot akan segera berpindah ke posisi baru untuk menjauh.
- d. Jika musuh terlalu dekat, bot akan memilih sudut acak antara 30 hingga 150 derajat untuk berbelok sehingga mencegah bot berbelok dalam pola yang dapat diprediksi oleh lawan. Lalu, bot akan berbelok ke kanan dan maju untuk keluar dari zona bahaya dan terhindar dari tembakan musuh.
- e. Pada tiap pengambilan keputusan, apabila bot menabrak dinding, bot akan langsung berbalik arah dan maju sehingga mencegah bot terjebak di sudut arena.
- f. Pada tiap pengambilan keputusan, apabila bot bertabrakan dengan musuh, bot akan langsung mundur untuk menjauh dan berbelok dengan sudut acak sehingga arah kabur bot tidak mudah ditebak oleh musuh serta bergerak maju 100 unit agar tidak bertemu musuh lagi.

#### 3.2.4 Alternatif Solusi 4 : Yo-Bot

Pada Yo-Bot ini, menggunakan pendekatan greedy terhadap perburuan musuh dengan tujuan untuk selalu mendekati musuh dan menembak dari jarak dekat. Heuristik yang digunakan yaitu greedy terhadap **perburuan musuh** dengan bot selalu berusaha mendekati musuh untuk menembak dari jarak dekat dan greedy terhadap serangan dengan bot menembak tiap kali melihat musuh tanpa mempertimbangkan strategi bertahan. Berikut langkah-langkah greedynya:

- a. Pada setiap pengambilan keputusan, bot mengejar dan menyerang musuh secara agresif namun bot selalu bergerak dalam pola memutar dan tanpa menabrak (menjaga jarak).
- b. Bot selalu berputar ke kiri dengan sudut 20 derajat agar terus bergerak dan tidak menjadi sasaran diam serta bergerak melingkar membuat kesulitan membidik posisi bot.
- c. Pada tiap pengambilan keputusan, bot akan menembak langsung dengan kekuatan penuh tanpa mempertimbangkan jumlah energi tersisa.
- d. Pada tiap langkah pengambilan keputusan, bot menghitung sudut yang mengarah ke bot musuh dan mengubah arah putaran sehingga selalu menghadap ke arah musuh secepat mungkin (mengejar). Namun, jika jarak lebih dari 50 unit dengan bot musuh, bot akan maju mendekati musuh bukan menabrak untuk menjaga jarak aman tidak diserang balik.
- e. Pada tiap pengambilan keputusan, apabila bot menabrak dinding, bot akan langsung berbalik arah dan maju sehingga mencegah bot terjebak di sudut arena.

### 3.3 Analisis Efektivitas dan Efisiensi

Pada Popo-Bot, strategi utama yang digunakan adalah pola gerakan spiral yang semakin melebar untuk menghindari tembakan lawan serta pendekatan greedy terhadap serangan dengan

menembak setiap kali musuh terdeteksi tanpa mempertimbangkan jarak. Keunggulan utama strategi ini adalah bot selalu bergerak sehingga sulit diprediksi serta memiliki waktu reaksi yang cepat dalam menembak. Namun, bot tidak mempertimbangkan konsumsi energi, yang dapat menyebabkan kehabisan energi lebih cepat. Fungsi `Run()` pada Popo-Bot mengandung *if-conditional* yang mengecek dan mengembalikan *increment* perubahan gerakan saat mencapai angka tertentu, karenanya kompleksitas waktu `Run()`  $O(n)$ . Fungsi lainnya hanya dijalankan pada *trigger* sehingga kompleksitasnya  $O(1)$ . Dari segi efisiensi, kompleksitas waktunya adalah  $O(n)$  per langkah.

Strategi utama Shi-Bot adalah bergerak dalam pola zig-zag dengan arah yang berubah setelah beberapa langkah, membuatnya sulit diprediksi oleh musuh. Bot ini menggunakan pendekatan greedy dengan segera menembak begitu mendeteksi lawan tanpa mempertimbangkan jarak atau kondisi lainnya. Keunggulan dari strategi ini adalah kesulitannya untuk ditembak karena pergerakan yang tidak linier, namun kelemahannya adalah tidak mempertimbangkan posisi lawan secara optimal, yang dapat menyebabkan bot berpindah ke arah yang lebih berisiko. Fungsi `Run()` berjalan dalam loop yang terus menerus mengevaluasi perubahan arah berdasarkan hitungan langkah, sehingga kompleksitas waktunya adalah  $O(n)$ . Fungsi lain seperti `OnScannedBot()`, `OnHitWall()`, dan `OnHitBot()` hanya dipanggil saat trigger tertentu terjadi, sehingga kompleksitasnya  $O(1)$ . Dengan demikian, kompleksitas total bot ini tetap  $O(n)$  per siklus permainan.

Ro-Bot menerapkan strategi defensif, dengan selalu bergerak maju untuk menghindari tembakan lawan dan mengubah arah jika musuh terlalu dekat. Implementasi greedy dilakukan bot ini dengan selalu mengambil posisi yang “optimal” pada setiap kesempatan. Bot ini juga tetap menyerang dengan pendekatan greedy, di mana ia akan menembak setiap kali mendeteksi musuh tanpa mempertimbangkan efisiensi energi atau jarak optimal. Keunggulan strategi ini adalah meminimalisir kemungkinan terkena tembakan, terutama saat berhadapan dengan bot yang diam atau memiliki pola gerakan tetap. Namun, kelemahannya adalah bot cenderung kurang agresif dan dapat kesulitan menghadapi lawan yang memiliki akurasi tinggi atau bisa mengejar dengan cepat. Fungsi `Run()` berjalan dalam loop yang terus melakukan pergerakan maju, sehingga kompleksitas waktunya  $O(n)$ . Fungsi `OnScannedBot()`, `OnHitWall()`, dan `OnHitBot()` hanya dipanggil saat event tertentu, yang berarti kompleksitasnya  $O(1)$  per pemanggilan. Secara keseluruhan, kompleksitas total bot ini tetap  $O(n)$  per siklus permainan.

Terakhir, Yo-Bot menggunakan strategi agresif dengan terus mencari musuh dan mengejarnya setelah terdeteksi, sambil tetap menembak setiap kali melihat lawan. Pendekatan ini cocok untuk menghadapi bot yang cenderung pasif atau defensif, karena Yo-Bot akan terus mendekati dan memberikan tekanan. Keunggulan utama dari strategi ini adalah bot selalu aktif dalam pertempuran dan dapat mengurangi ruang gerak lawan. Namun, kelemahannya adalah jika bot terlalu agresif tanpa mempertimbangkan energi dan posisi optimal, ia bisa mudah terkena serangan balik atau terjebak dalam pertempuran jarak dekat yang tidak menguntungkan. Fungsi

Run() yang berisi loop pergerakan putaran memiliki kompleksitas  $O(n)$ , sedangkan fungsi OnScannedBot(), OnHitWall(), dan TurnToFaceTarget() hanya dijalankan saat event tertentu, sehingga kompleksitasnya  $O(1)$  per pemanggilan. Secara keseluruhan, kompleksitas bot ini tetap  $O(n)$  per siklus permainan.

### 3.4 Strategi Greedy yang Dipilih

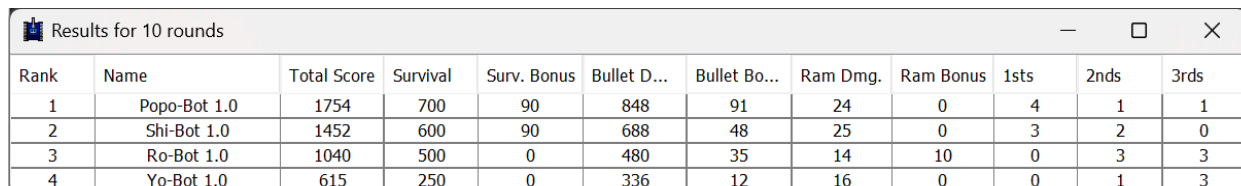
Berdasarkan analisis efisiensi dan efektivitas yang sudah dipaparkan sebelumnya, kami memutuskan memilih strategi greedy sebagai berikut:

1. Greedy berdasarkan serangan
2. Greedy berdasarkan gerakan
3. Greedy berdasarkan poin

Ketiga strategi ini saling melengkapi, dan ketika digabungkan, dapat mengatasi kelemahan masing-masing serta meningkatkan peluang kemenangan dibandingkan strategi greedy lainnya. Namun, karena posisi awal bot dalam arena bersifat acak dan tidak dapat diprediksi, hasil pertandingan tetap bergantung pada kondisi awal yang diberikan.

Untuk mengoptimalkan performa bot, kami menetapkan urutan prioritas dalam penerapan strategi. Prioritas pertama adalah greedy dalam pergerakan, yang memastikan bot selalu bergerak agar sulit menjadi target serangan lawan. Prioritas kedua adalah greedy dalam serangan, di mana bot akan segera menembak setiap kali menemukan lawan untuk meningkatkan peluang kemenangan. Prioritas ketiga adalah greedy dalam perolehan poin, yang berfokus pada upaya bot untuk memaksimalkan skor, baik dengan bertahan hingga akhir pertandingan maupun dengan kembali menerapkan strategi pergerakan dan serangan secara efektif.

Dalam implementasinya, kami merancang empat bot dengan berbagai alternatif strategi, baik yang menggunakan salah satu aspek greedy secara spesifik maupun yang menggabungkan ketiganya. Setiap bot memiliki pendekatan heuristik yang berbeda, sehingga dapat beradaptasi dengan berbagai skenario pertempuran di arena. Dengan strategi ini, kami berharap bot dapat menunjukkan performa yang kompetitif dan memiliki peluang menang yang lebih besar.



Rank	Name	Total Score	Survival	Surv. Bonus	Bullet D...	Bullet Bo...	Ram Dmg.	Ram Bonus	1sts	2nds	3rds
1	Popo-Bot 1.0	1754	700	90	848	91	24	0	4	1	1
2	Shi-Bot 1.0	1452	600	90	688	48	25	0	3	2	0
3	Ro-Bot 1.0	1040	500	0	480	35	14	10	0	3	3
4	Yo-Bot 1.0	615	250	0	336	12	16	0	0	1	3

Gambar 5 - Hasil pertandingan bot

Dari hasil *trial* yang dilakukan dengan mempertandingkan keempat bot melawan satu sama lain serta bot sampel, Popo-Bot dipilih sebagai bot terbaik karena relatif menang dalam pertempuran melawan bot lainnya. Keunggulan utama Popo-Bot terletak pada strategi pergerakan spiral yang terus berubah dan semakin melebar, membuatnya sulit diprediksi serta lebih sulit terkena tembakan lawan. Selain itu, pendekatan greedy dalam menembak memastikan



bahwa setiap musuh yang terdeteksi langsung ditembak tanpa penundaan, memberikan tekanan konstan kepada lawan. Dibandingkan dengan bot lain, Popo-Bot memiliki keseimbangan dan kombinasi antara mobilitas tinggi dan agresivitas, yang memungkinkan bot ini untuk bertahan lebih lama sekaligus memberikan serangan yang efektif.

## BAB IV

# IMPLEMENTASI DAN PENGUJIAN

### 4.1 Implementasi 4 Alternatif Solusi (Pseudocode)

#### 4.1.1 Pseudocode untuk Main Bot (Popo-Bot)

```
PseudoCode Popo-Bot

BEGIN PopoBot

    SET spiralDistance TO 100
    SET angleIncrement TO 10

    FUNCTION Main(args)
        CREATE INSTANCE OF PopoBot
        START INSTANCE

    FUNCTION PopoBot()
        LOAD BOT CONFIGURATION FROM "PopoBot.json"

    FUNCTION Run()
        SET BodyColor TO Blue
        SET TurretColor TO Yellow
        SET RadarColor TO Blue
        SET ScanColor TO Yellow

        SET GunTurnRate TO 15

        /* Algoritma Greedy: Selalu bergerak agar meminimalisir ditembak musuh. */
        /* Bergerak secara spiral, dengan lama-lama semakin jauh dan anglenya menaik,
        tetapi di cap ke 500. */
        WHILE Bot is Running DO
            IF spiralDistance > 500 THEN
                SET spiralDistance TO 100
                SET angleIncrement TO 10
            END IF

            MOVE FORWARD BY spiralDistance
            TURN RIGHT BY angleIncrement

            INCREMENT spiralDistance BY 5
            INCREMENT angleIncrement BY 0.5
        END WHILE
    END FUNCTION

    /* Algoritma Greedy: Setiap melihat musuh, DORRRR! */
    FUNCTION OnScannedBot(evt)
        FIRE POWER 3
    END FUNCTION

    FUNCTION OnHitBot(e)
        IF Bot is Rammed THEN
            TURN RIGHT BY 30
            MOVE FORWARD BY 50
        END IF
    END FUNCTION

END PopoBot
```

#### 4.1.2 Pseudocode untuk Alternative Bot 1 (Shiro-Bot / ZigiZagaBot)

```
PseudoCode ZigiZaga-Bot (Shi-Bot)

BEGIN ZigiZagaBot

    SET moveDistance TO 1000    // Jarak pergerakan
    SET turnAngle TO 145        // Sudut belokan
    SET isMovingForward TO TRUE
    SET moveCount TO 0
    SET MAX_MOVES TO 3          // Berapa kali belok sebelum mengubah arah

    FUNCTION Main(args)
        CREATE INSTANCE OF ZigiZagaBot
        START INSTANCE

    FUNCTION ZigiZagaBot()
        LOAD BOT CONFIGURATION FROM "ZigiZagaBot.json"

    FUNCTION Run()
        SET BodyColor TO Red
        SET TurretColor TO Blue
        SET RadarColor TO Red
        SET ScanColor TO Blue

        SET GunTurnRate TO 15

        // Algoritma Greedy: Selalu bergerak, dengan arah yang tidak bisa ditebak
        WHILE Bot is Running DO
            // Ubah arah setelah MAX_MOVES gerakan
            IF moveCount >= MAX_MOVES THEN
                TOGGLE isMovingForward
                SET moveCount TO 0
            END IF

            // Gerak maju atau mundur
            IF isMovingForward THEN
                MOVE FORWARD BY moveDistance
            ELSE
                MOVE BACKWARD BY moveDistance
            END IF

            // Zigzag dengan belokan kanan/kiri bergantian
            IF moveCount MOD 2 == 0 THEN
                TURN RIGHT BY turnAngle
            ELSE
                TURN LEFT BY turnAngle
            END IF

            INCREMENT moveCount BY 1
        END WHILE
    END FUNCTION

    // Algoritma Greedy: Ketika melihat musuh langsung menembak
    FUNCTION OnScannedBot(evt)
        FIRE POWER 3
    END FUNCTION

    FUNCTION OnHitWall(e)
        TURN RIGHT BY 30
        MOVE FORWARD BY 50
        SET moveCount TO 0
    END FUNCTION

    FUNCTION OnHitBot(e)
        TURN RIGHT BY 30
        MOVE FORWARD BY 50
    END FUNCTION

END ZigiZagaBot
```

### 4.1.3 Pseudocode untuk Alternative Bot 2 (Ro-Bot / KaburBot)

```
PseudoCode Kabur-Bot (Ro-Bot)

BEGIN KaburBot

    SET lastEnemyDistance TO MAX_VALUE // Menyimpan jarak terakhir dengan musuh
    SET SAFE_DISTANCE TO 400 // Jarak aman dari musuh
    INITIALIZE random NUMBER GENERATOR

    FUNCTION Main(args)
        CREATE INSTANCE OF KaburBot
        START INSTANCE

    FUNCTION KaburBot()
        LOAD BOT CONFIGURATION FROM "KaburBot.json"

    FUNCTION Run()
        SET BodyColor TO Green
        SET TurretColor TO DarkGreen
        SET RadarColor TO LightGreen
        SET ScanColor TO Yellow

        SET GunTurnRate TO 20

        /* Algoritma Greedy: Bot selalu bergerak maju sejauh 400 untuk menghindari
        tembakan musuh*/
        WHILE Bot is Running DO
            SET BodyColor TO Green // Debugging indicator
            MOVE FORWARD BY 400
        END WHILE
    END FUNCTION

    /* Algoritma Greedy: Ketika melihat musuh yang dekat,
    berpindah arah agar tidak sejajar dengan musuh dan tidak terkena tembakan */
    FUNCTION OnScannedBot(e)
        FIRE POWER 3 // Selalu menembak saat melihat musuh

        SET lastEnemyDistance TO DISTANCE TO (e.X, e.Y)

        // Jika musuh terlalu dekat, menghindar dengan berpindah arah
        IF lastEnemyDistance < SAFE_DISTANCE THEN
            SET BodyColor TO Yellow // Debugging indicator
            SET randomAngle TO RANDOM VALUE BETWEEN 30 AND 150
            TURN RIGHT BY randomAngle
            MOVE FORWARD BY 100
        END IF
    END FUNCTION

    FUNCTION OnHitWall(e)
        TURN RIGHT BY 180 // Jika menabrak dinding, putar balik
        MOVE FORWARD BY 100
    END FUNCTION

    FUNCTION OnHitBot(e)
        MOVE BACKWARD BY 50
        TURN RIGHT BY RANDOM VALUE BETWEEN 90 AND 180
        MOVE FORWARD BY 100
    END FUNCTION

END KaburBot
```

#### 4.1.4 Pseudocode untuk Alternative Bot 3 (Yo-Bot / KejarBot)

```
PseudoCode Kejar-Bot (Yo-Bot)

BEGIN KejarBot

    SET turnDirection TO 1 // Variabel untuk menentukan arah belok kiri atau kanan

    FUNCTION Main(args)
        CREATE INSTANCE OF KejarBot
        START INSTANCE

    FUNCTION KejarBot()
        LOAD BOT CONFIGURATION FROM "KejarBot.json"

    FUNCTION Run()
        SET BodyColor TO Black
        SET TurretColor TO DarkRed
        SET RadarColor TO Orange
        SET ScanColor TO Yellow

        // Bot akan terus berputar untuk mencari musuh
        WHILE Bot is Running DO
            TURN LEFT BY (20 * turnDirection)
        END WHILE
    END FUNCTION

    /* Algoritma Greedy: Setiap melihat musuh, langsung menembak lalu mengejar,
    tetapi tidak sampai menabrak musuh */
    FUNCTION OnScannedBot(e)
        FIRE POWER 3 // Tembak saat melihat musuh

        CALL TurnToFaceTarget(e.X, e.Y) // Hadapkan bot ke arah musuh
        SET distance TO DISTANCE TO (e.X, e.Y)

        // Jika jarak ke musuh lebih dari 50, dekati musuh
        IF distance > 50 THEN
            MOVE FORWARD BY (distance - 50)
        END IF
    END FUNCTION

    /* Menghadapkan bot ke arah musuh */
    FUNCTION TurnToFaceTarget(x, y)
        SET bearing TO BEARING TO (x, y)

        // Tentukan arah belok berdasarkan posisi musuh
        IF bearing >= 0 THEN
            SET turnDirection TO 1
        ELSE
            SET turnDirection TO -1
        END IF

        TURN LEFT BY bearing
    END FUNCTION

    /* Jika menabrak dinding, berbalik arah */
    FUNCTION OnHitWall(e)
        TURN RIGHT BY 180
        MOVE FORWARD BY 100
    END FUNCTION

END KejarBot
```

## 4.2 Penjelasan Lengkap Popo-Bot

Berdasarkan analisis yang kelompok kami lakukan, kami memutuskan bahwa bot dengan solusi greedy terbaik yang dipilih yaitu bot dengan nama Popo-Bot. Berikut algoritma C# dari bot ini:

```
using Robocode.TankRoyale.BotApi;
using Robocode.TankRoyale.BotApi.Events;
using System.Drawing;
public class PopoBot : Bot
{
    private double spiralDistance = 100;
    private double angleIncrement = 10;
    static void Main(string[] args)
    {
        new PopoBot().Start();
    }
    PopoBot() : base(BotInfo.FromFile("PopoBot.json")) { }
    public override void Run()
    {
        BodyColor = Color.Blue;
        TurretColor = Color.Yellow;
        RadarColor = Color.Blue;
        ScanColor = Color.Yellow;
        GunTurnRate = 15;
        while (IsRunning)
        {
            if(spiralDistance > 500){
                spiralDistance = 100;
                angleIncrement = 10;
            }
            Forward(spiralDistance);
            TurnRight(angleIncrement);
            spiralDistance += 5;
            angleIncrement += 0.5;
        }
    }
    public override void OnScannedBot(ScannedBotEvent evt)
    {
        Fire(3);
    }
    public override void OnHitBot(HitBotEvent e)
    {
        if (e.IsRammed)
        {
            TurnRight(30);
            Forward(50);
        }
    }
}
```

```
}  
}
```

Struktur data yang kami gunakan dalam pembuatan Popo-Bot ini secara garis besar terbagi menjadi 5, yaitu sebagai berikut:

a. PopoBot.cmd

File ini berisi script batch Windows yang berfungsi menjalankan bot dalam mode pengembangan atau rilis pada proyek Robocode Tank Royale dengan .NET. Saat dijalankan dalam mode dev, bot selalu dikompilasi ulang dan dijalankan dari awal sehingga cocok untuk debugging. Sedangkan mode release hanya melakukan kompilasi ulang jika belum pernah dibangun sebelumnya sehingga cocok untuk menjalankan bot dengan cepat tanpa membuang waktu untuk rebuild.

b. PopoBot.cs

File ini berisi kode utama dalam bahasa C# yang berisi logika dan algoritma dari Popo-Bot. Pada file ini terdapat beberapa variabel yang digunakan yaitu:

1. spiralDistance : jarak yang ditempuh bot dalam pola spiral dimulai dari 100 bertambah hingga maksimal 500.
2. angleIncrement : besar sudut belokan untuk menciptakan pola spiral.
3. GunTurnRate : kecepatan pergerakan turret senjata bot.

Berikut beberapa metodenya:

1. Fungsi Main() : fungsi utama program untuk memulai bot dan menjalankan bot.
2. Konstruktor PopoBot() : memuat konfigurasi bot dari file JSON dan mengisi class Bot yang merupakan bagian dari API.
3. Prosedur Run() : prosedur ini digunakan untuk mengatur warna bot yaitu badan biru, turret kuning, radar biru, dan scan kuning. Selain itu, digunakan juga untuk mengatur kecepatan putaran turret dan menjalankan pergerakan bot dalam pola spiral dengan selalu bergerak maju, berbelok kanan, jarak tempuh bertambah 5 unit setiap iterasi, sudut belok bertambah 0.5 setiap iterasi, dan jika jarak spiral sudah melebihi 500 akan di reset kembali ke 100. Pergerakan ini agar bot selalu bergerak sehingga meminimalisir terkena tembakan musuh.
4. Prosedur OnScannedBot(ScannedBotEvent evt) : prosedur ini digunakan untuk menembak peluru dengan kekuatan maksimum saat bot mendeteksi musuh menggunakan radar. Penggunaan Fire(3) agar tembakan yang terkena musuh sangatlah sakit namun bila tidak mengenai, energi bot kita yang berkurang.
1. Prosedur OnHitBot(HitBotEvent e) : prosedur ini digunakan untuk penanganan pergerakan bot saat menabrak bot musuh sehingga perlu berbelok kanan 30 derajat dan mundur sedikit ke belakang untuk

menghindari macet atau stuck. Hal ini membuat bot tidak terkena tembakan karena arah berikutnya berbeda dengan saat bertabrakan.

c. PopoBot.csproj

File ini berisi konfigurasi proyek .NET yang digunakan oleh MSBuild untuk mengelola dependensi, properti proyek, dan pengaturan build. Berikut beberapa tag dan library yang ada:

1. <RootNameSpace> : menentukan namespace default kode bot.
2. <OutputType> : menentukan tipe keluaran proyek yaitu Exe (aplikasi yang bisa dijalankan).
3. <TargetFramework> : menentukan framework .NET yang digunakan.
4. <LangVersion> : menentukan versi bahasa C# yang digunakan.
5. Robocode.TankRoyale.BotApi : library utama untuk ebrinteraksi dnegan Robocode Tank Royale API sehingga memungkinkan bot bergerak, menembak, dan merespons sekitar.

d. PopoBot.json

File ini berisi konfigurasi dalam format JSON yang menyimpan informasi tentang bot berupa nama bot, penulis bot, versi bot, dan parameter eksekusi bot.

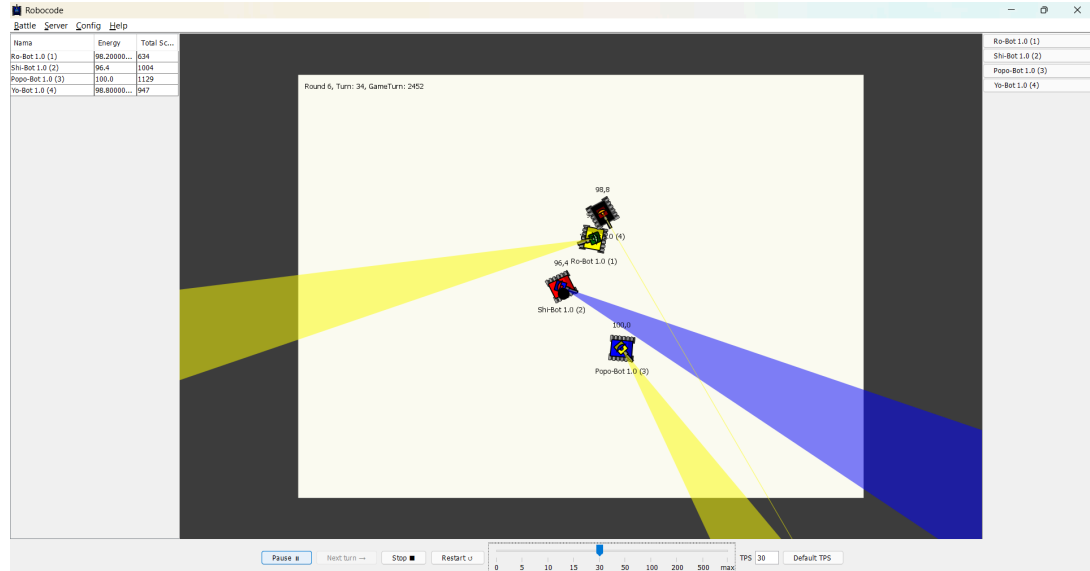
e. PopoBot.sh

File ini berisi script untuk Linux/MacOS yang berfungsi seperti PopoBot.cmd namun ditulis dalam bash script (.sh) sehingga dapat menjalankan bot di lingkungan berbasis UNIX.

### 4.3 Pengujian dan Analisisnya

Untuk menguji apakah algoritma Greedy yang diterapkan pada bot utama berhasil atau tidak, maka dapat dilakukan pengujian dengan menandingkan bot utama dengan ketiga bot alternatif lainnya. Dengan demikian, dapat diamati perilaku bot apakah sesuai dengan yang diharapkan dan dapat memenangkan pertandingan atau tidak. Pengujian dilakukan sebanyak 6 kali, dengan 3 kali ditandingkan secara 1 vs 1 vs 1 vs 1 melawan tiga bot alternatif, dan 3 kali ditandingkan melawan sample bot yang ada pada *starter pack*.





Gambar 6 - Contoh pengujian menandingkan bot dalam permainan 1 vs 1 vs 1 vs 1

### 4.3.1 Pengujian Bot Utama Melawan Bot Alternatif

Pengujian Pertama:

Rank	Name	Total Score	Survival	Surv. Bonus	Bullet Dmg.	Bullet Bonus	Ram Dmg.	Ram Bonus	1sts	2nds	3rds
1	Popo-Bot 1.0	2288	900	120	1120	103	44	0	4	3	1
2	Shi-Bot 1.0	2288	1050	120	992	93	32	0	4	4	2
3	Ro-Bot 1.0	1324	750	30	512	0	32	0	1	3	4
4	Yo-Bot 1.0	895	300	30	480	48	36	0	1	0	3

Gambar 7 - Hasil Pengujian 1 Bot Utama Melawan Bot Alternatif

Berdasarkan hasil pengujian pertama, dapat dilihat bot utama (Popo-Bot) berada di peringkat pertama, dengan total skor sama dengan bot alternatif 1 (Shi-Bot). Hal yang menarik adalah pada pertandingan ini, *survival score* Popo-Bot dikalahkan oleh Shi-Bot, atau bot yang selalu bergerak tanpa bisa ditebak. Perbedaan skor yang sangat tipis ini terjadi bisa disebabkan karena posisi awal setiap bot di setiap ronde yang berbeda-beda. Posisi bot di awal ronde dapat menentukan keberlangsungan permainan. Akan tetapi, sesuai dengan algoritma *greedy* yang diharapkan, Popo-Bot memiliki *bullet damage* terbesar, dan *survival* kedua, berbeda tipis dengan Shi-Bot.

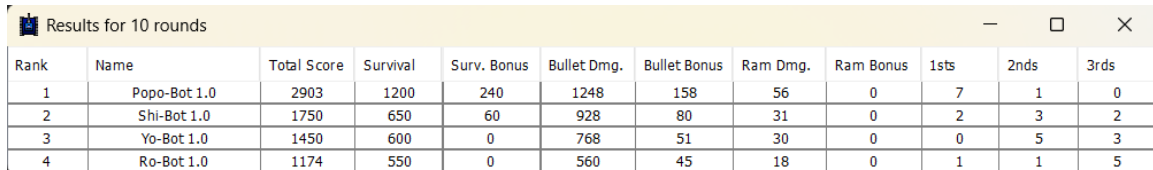
Pengujian Kedua:

Rank	Name	Total Score	Survival	Surv. Bonus	Bullet Dmg.	Bullet Bonus	Ram Dmg.	Ram Bonus	1sts	2nds	3rds
1	Popo-Bot 1.0	2804	1300	240	1088	142	34	0	7	1	2
2	Yo-Bot 1.0	1543	600	0	832	90	20	0	0	6	0
3	Ro-Bot 1.0	1427	600	30	720	55	22	0	2	2	3
4	Shi-Bot 1.0	1412	500	30	800	16	50	15	1	1	5

Gambar 8 - Hasil Pengujian 2 Bot Utama Melawan Bot Alternatif

Pada pengujian kali ini, didapatkan bahwa Popo-Bot berhasil memenangkan pertandingan secara telak. Hal ini menunjukkan bahwa strategi algoritma *greedy* yang direncanakan berhasil, dapat dilihat dari *bullet damage* yang paling besar, serta *survival* paling besar juga.

Pengujian Ketiga:



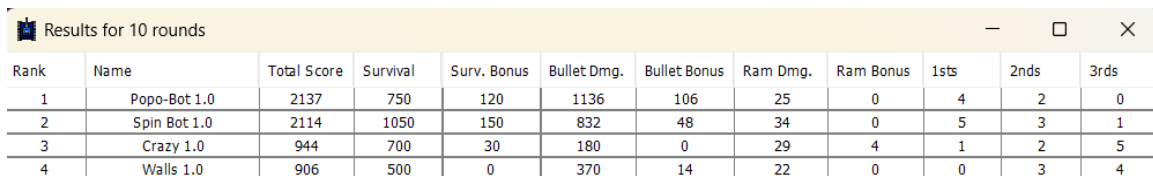
Rank	Name	Total Score	Survival	Surv. Bonus	Bullet Dmg.	Bullet Bonus	Ram Dmg.	Ram Bonus	1sts	2nds	3rds
1	Popo-Bot 1.0	2903	1200	240	1248	158	56	0	7	1	0
2	Shi-Bot 1.0	1750	650	60	928	80	31	0	2	3	2
3	Yo-Bot 1.0	1450	600	0	768	51	30	0	0	5	3
4	Ro-Bot 1.0	1174	550	0	560	45	18	0	1	1	5

Gambar 9 - Hasil Pengujian 3 Bot Utama Melawan Bot Alternatif

Pengujian terakhir semakin menguatkan bahwa strategi algoritma *greedy* milik Popo-Bot teruji berhasil. Meskipun strategi *greedy*-nya dapat dikatakan cukup simpel, tapi terbukti mampu menjawab persoalan peraihan poin terbanyak pada Robocode. Popo-Bot setiap *turn* selalu bergerak, sehingga lebih sedikit terkena tembakan musuh, serta selalu menembak ketika melihat musuh agar poin yang didapatkan dari tembakan besar.

#### 4.3.1 Pengujian Bot Utama Melawan Bot Sampel dari *Starter Pack*

Pengujian Pertama:



Rank	Name	Total Score	Survival	Surv. Bonus	Bullet Dmg.	Bullet Bonus	Ram Dmg.	Ram Bonus	1sts	2nds	3rds
1	Popo-Bot 1.0	2137	750	120	1136	106	25	0	4	2	0
2	Spin Bot 1.0	2114	1050	150	832	48	34	0	5	3	1
3	Crazy 1.0	944	700	30	180	0	29	4	1	2	5
4	Walls 1.0	906	500	0	370	14	22	0	0	3	4

Gambar 10 - Hasil Pengujian 1 Bot Utama Melawan Bot Sampel pada *Starter Pack*

Berdasarkan hasil pengujian bersama, dapat dilihat bahwa Popo-Bot berhasil memenangkan pertandingan, meskipun perbedaan *total score* dengan Spin Bot yang menduduki peringkat dua sangat sedikit. Pada pertandingan ini, terlihat bahwa *survival* Popo-Bot tidak yang pertama, melainkan dikalahkan oleh Spin Bot. Ini menunjukkan ada kalanya strategi algoritma *greedy* yang diimplementasikan tidak berhasil untuk memaksimalkan *survival* point. Hal ini terjadi salah satunya karena penempatan posisi bot di setiap awal ronde yang diacak. Pada pertandingan ini, Popo-Bot meskipun selalu bergerak, berada di kawasan tembak bot lainnya sehingga tetap terkena tembakan. Akan tetapi, untuk *bullet damage* Popo-Bot masih memimpin seperti sebelum-sebelumnya.

Pengujian Kedua:

Results for 10 rounds											
Rank	Name	Total Score	Survival	Surv. Bonus	Bullet Dmg.	Bullet Bonus	Ram Dmg.	Ram Bonus	1sts	2nds	3rds
1	Popo-Bot 1.0	1818	750	60	896	84	28	0	5	2	1
2	Spin Bot 1.0	1260	600	60	512	30	58	0	2	3	3
3	Walls 1.0	1202	450	90	550	46	37	29	2	1	1
4	Crazy 1.0	782	600	30	128	0	23	0	0	3	4

Gambar 11 - Hasil Pengujian 2 Bot Utama Melawan Bot Sampel pada *Starter Pack*

Percobaan kedua, kembali menunjukkan bahwa strategi algoritma milik Popo-Bot memberikan hasil yang maksimal. Terlihat bahwa Popo-Bot memenangkan pertandingan secara telak, mengalahkan bot-bot sampel lainnya dari segi *survivability* dan *bullet damage*, sesuai dengan yang diharapkan dari metode *greedy* yang diimplementasikan.

### Pengujian Ketiga:

Results for 10 rounds											
Rank	Name	Total Score	Survival	Surv. Bonus	Bullet Dmg.	Bullet Bonus	Ram Dmg.	Ram Bonus	1sts	2nds	3rds
1	Spin Bot 1.0	1894	900	90	784	50	70	0	3	3	4
2	Popo-Bot 1.0	1605	500	30	976	58	41	0	4	0	2
3	Crazy 1.0	1472	1100	120	192	6	53	0	1	7	2
4	Walls 1.0	952	500	60	340	8	34	10	2	0	2

Gambar 12 - Hasil Pengujian 3 Bot Utama Melawan Bot Sampel pada *Starter Pack*

Percobaan ketiga, akhirnya Popo-Bot kalah untuk pertama kalinya. Pada pertandingan kali ini, *survival score* milik Popo-Bot berada pada peringkat ketiga, bahkan yang tertinggi adalah Crazy, dimana sebelum-sebelumnya tidak terlalu menonjol. Berdasarkan percobaan ini, kami melihat tetap ada satu faktor lain yang mempengaruhi berjalannya pertandingan, yaitu keberuntungan. Akan tetapi, melihat *bullet damage* tetap yang terbesar, membuktikan bahwa strategi algoritma *greedy* untuk meningkatkan skor berdasarkan tembakan berhasil.

## **BAB V**

### **KESIMPULAN DAN SARAN**

#### **5.1 Kesimpulan**

Algoritma greedy adalah pendekatan penyelesaian masalah dengan mengambil keputusan lokal tanpa mempertimbangkan langkah kedepannya. Pemanfaatan algoritma greedy dalam pengembangan bot Robocode Tank Royale terbukti sebagai pendekatan yang efektif dalam meningkatkan kemenangan. Hal ini dapat dibuktikan dengan menangnya Popo-Bot dibandingkan dengan CrazyBot. Dengan menerapkan strategi greedy terhadap pergerakan, greedy terhadap serangan, dan greedy terhadap perolehan poin, bot dapat bertahan lebih lama di arena, menyerang musuh secara agresif, dan mengoptimalkan perolehan skor.

Keempat bot telah dirancang dengan pendekatan heuristik yang berbeda sehingga memungkinkan analisis lebih dalam terhadap efektivitas strategi masing-masing. Popo-Bot menggunakan kombinasi dari ketiga strategi dengan gerakan pola spiral. Shi-Bot menggunakan heuristik dengan pergerakan pola zig-zag. Ro-Bot menggunakan strategi greedy menghindari musuh. Terakhir, Yo-Bot menggunakan strategi greedy mengejar bot musuh dan menembaknya secara agresif.

Bot yang mengkombinasikan ketiga aspek greedy memiliki keunggulan dibandingkan bot yang hanya fokus pada satu aspek saja. Namun, hasil akhir pertandingan tetap dipengaruhi oleh faktor eksternal, seperti posisi awal di arena dan strategi bot lawan. Secara keseluruhan, algoritma greedy dapat menjadi dasar yang kuat dalam pengembangan bot untuk Robocode. Meskipun bersifat lokal optima dan tidak selalu menjamin solusi terbaik dalam jangka panjang, strategi ini memberikan respons cepat terhadap kondisi pertempuran, yang sering kali lebih menguntungkan dalam permainan kompetitif seperti Robocode Tank Royale.

#### **5.2 Saran**

Apa yang kita tuangkan pada tugas besar kali ini pasti tidak luput dari kesalahan. Saran untuk kedepannya, sebaiknya lebih mengeksplor strategi greedy yang lain sehingga memiliki banyak opsi dalam pembuatan botnya. Selain itu, lebih banyak melakukan uji coba mengenai kekuatan dan kelemahan bot serta materi algoritma greedy. Beberapa aspek dalam implementasi dapat diperbaiki untuk meningkatkan efisiensi pemrosesan, seperti pengelolaan memori, optimasi perhitungan sudut, dan strategi penghindaran yang lebih cerdas agar bot lebih responsif terhadap ancaman. Dengan beberapa saran tersebut, diharapkan dapat menciptakan pengetahuan lebih dan bot yang lebih baik.

## LAMPIRAN

- Tautan Repository GitHub : [https://github.com/indahtangdililing/Tubes1\\_PopoShiroyo.git](https://github.com/indahtangdililing/Tubes1_PopoShiroyo.git)
- Tautan Video Youtube : <https://youtu.be/u7ujq8ps0f8>

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

## DAFTAR PUSTAKA

Munir, Rinaldi. 2025. “Algoritma Brute Force (Bagian 2)”.

[https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2024-2025/03-Algoritma-Brute-Force-\(2025\)-Bag2.pdf](https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2024-2025/03-Algoritma-Brute-Force-(2025)-Bag2.pdf).

Munir, Rinaldi. 2025. “Algoritma Greedy (Bagian 1)”.

[https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2024-2025/04-Algoritma-Greedy-\(2025\)-Bag1.pdf](https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2024-2025/04-Algoritma-Greedy-(2025)-Bag1.pdf).

Munir, Rinaldi. 2025. “Algoritma Greedy (Bagian 2)”.

[https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2024-2025/05-Algoritma-Greedy-\(2025\)-Bag2.pdf](https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2024-2025/05-Algoritma-Greedy-(2025)-Bag2.pdf).

Munir, Rinaldi. 2025. “Algoritma Greedy (Bagian 3)”.

[https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2024-2025/06-Algoritma-Greedy-\(2025\)-Bag3.pdf](https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2024-2025/06-Algoritma-Greedy-(2025)-Bag3.pdf).