LAPORAN TUGAS BESAR 1
IF2211 – Strategi Algoritma
Robocode Tank Royale

Disusun oleh :

M Hazim Ramadhan P	13523009
Adhimas Aryo Bimo	135523052
Darrel Adinarya Sunanda	13523061

Dosen Pengajar :
Dr. Nur Ulfa Maulidevi, S.T, M.Sc.

PROGRAM STUDI TEKNIK INFORMATIKA
SEKOLAH TEKNIK ELEKTRO DAN INFORMATIKA
INSTITUT TEKNOLOGI BANDUNG

2025

Kata Pengantar

Segala puji dan syukur kami panjatkan ke hadirat Tuhan Yang Maha Esa atas limpahan rahmat dan karunia-Nya, sehingga kami dapat menyelesaikan Laporan Tugas Besar 1 untuk mata kuliah Strategi Algoritma yang berjudul *Robocode Tank Royale* tepat pada waktunya. Laporan ini disusun sebagai salah satu bentuk tugas IF2211 yang bertujuan untuk mengaplikasikan teori yang telah dipelajari selama perkuliahan dalam menyelesaikan permasalahan menggunakan metode-metode dalam algoritma *greedy*.

Kami menyampaikan rasa terima kasih yang sebesar-besarnya kepada Dosen pengajar, Ibu Dr. Nur Ulfa Maulidevi, S.T, M.Sc., atas bimbingan dan arahannya selama proses perkuliahan. Ucapan terima kasih juga kami sampaikan kepada para asisten dosen serta asisten laboratorium yang telah memberikan penjelasan tambahan dan evaluasi terhadap tugas-tugas kami. Tak lupa, kami mengucapkan terima kasih kepada Muhammad Hazim Ramadhan Prajoda, Adhimas Aryo Bimo, dan Darrel Adinarya Sunanda atas kerja sama dan kontribusi yang luar biasa dalam penyelesaian tugas ini.

Akhir kata, kami berharap laporan ini dapat memberikan manfaat bagi kami serta para pembaca sekalian. Kami menyadari bahwa laporan ini masih memiliki kekurangan. Oleh karena itu, kami dengan senang hati menerima kritik dan saran yang membangun demi perbaikan di masa mendatang.

Bandung, 24 Maret 2024

Daftar Isi

Table of Contents

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.

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.

I.I Spesifikasi Wajib

- Buatlah 4 bot (1 utama dan 3 alternatif) dalam bahasa C# (.net) yang mengimplementasikan algoritma Greedy pada bot permainan Robocode Tank Royale dengan tujuan memenangkan permainan
- Tugas dikerjakan berkelompok dengan anggota minimal 2 orang dan maksimal 3 orang, boleh lintas kelas dan lintas kampus.
- Strategi greedy yang diimplementasikan setiap kelompok harus dikaitkan dengan fungsi objektif dari permainan ini, yaitu memperoleh skor setinggi mungkin pada akhir 7 pertempuran. Hal ini dapat dilakukan dengan mengoptimalkan komponen skor yang telah dijelaskan di atas.
- Strategi greedy yang diimplementasikan harus berbeda untuk setiap bot yang diimplementasikan dan setiap strategi greedy harus menggunakan heuristic yang berbeda.
- **Bot yang dibuat TIDAK BOLEH sama dengan SAMPEL yang diberikan sebagai CONTOH. Baik dari starter pack maupun dari repository engine asli.**
- Buatlah strategi greedy terbaik, karena setiap bot utama dari masing-masing kelompok akan diadu dalam kompetisi Tubes 1.
- Strategi greedy yang kelompok anda buat harus dijelaskan dan ditulis secara eksplisit pada laporan, karena akan diperiksa saat demo apakah strategi yang dituliskan sesuai dengan yang diimplementasikan.
- Setiap kelompok dapat menggunakan kreativitas yang bermacam macam dalam menyusun strategi greedy untuk memenangkan

permainan. Implementasi pemain harus dapat dijalankan pada game engine yang telah disebutkan diatas serta dapat dikompetisikan dengan bot dari kelompok lain.

- Program harus mengandung komentar yang jelas, dan untuk setiap strategi greedy yang disebutkan, harus dilengkapi dengan kode sumber yang dibuat. Artinya semua strategi harus diimplementasikan
- Mahasiswa disarankan membaca dokumentasi dari game engine. Perlu diperhatikan bahwa game engine yang digunakan untuk tubes ini SUDAH DIMODIFIKASI. Untuk Perubahan dapat dilihat pada bagian starter pack

I.II Spesifikasi Bonus

- (maks 10) Membuat video tentang aplikasi greedy pada bot serta simulasinya pada game kemudian mengunggahnya di Youtube. Video dibuat harus memiliki audio dan menampilkan wajah dari setiap anggota kelompok. Untuk contoh video tubes stima tahun-tahun sebelumnya dapat dilihat di Youtube dengan kata kunci “Tubes Stima”, “strategi algoritma”, “Tugas besar stima”, dll. Semakin menarik video, maka semakin banyak poin yang diberikan.
- (maks 10) Beberapa kelompok pemenang lomba kompetisi akan mendapatkan nilai tambahan berdasarkan posisi yang diraih.

Bab II. Landasan Teori

II.I Algoritma Greedy

Algoritma *greedy* atau algoritma serakah merupakan pendekatan penyelesaian masalah dalam ilmu komputer dan matematika diskrit yang didasarkan pada prinsip memilih solusi optimal lokal pada setiap langkah dengan harapan solusi tersebut mengarah pada solusi optimal global. Pada setiap iterasi, algoritma greedy akan memilih opsi yang tampak terbaik saat itu, tanpa mempertimbangkan konsekuensi jangka panjang dari keputusan tersebut. Contoh klasik dari penerapan algoritma greedy meliputi masalah *coin change*, *activity selection*, dan *minimum spanning tree* (seperti pada algoritma Kruskal dan Prim).

Kelebihan utama dari algoritma greedy terletak pada kesederhanaannya serta efisiensi waktu komputasi, karena tidak memerlukan eksplorasi seluruh ruang solusi seperti pendekatan *brute-force* atau *backtracking*. Algoritma ini juga mudah diimplementasikan dan cocok untuk masalah-masalah yang memenuhi sifat *greedy-choice property* dan *optimal substructure*, yaitu ketika solusi optimal dari submasalah dapat disusun menjadi solusi optimal keseluruhan.

Namun, algoritma greedy memiliki kekurangan signifikan, yaitu tidak selalu menghasilkan solusi optimal untuk semua jenis masalah. Karena hanya berfokus pada keuntungan sesaat di setiap langkah, algoritma ini bisa saja melewatkan solusi global yang lebih baik. Oleh karena itu, penting untuk melakukan analisis terhadap karakteristik masalah terlebih dahulu untuk memastikan bahwa pendekatan greedy dapat diterapkan secara efektif. Dalam kasus di mana sifat optimal substruktur tidak terpenuhi, pendekatan lain seperti *dynamic programming* mungkin lebih tepat digunakan.

Algoritma ini memiliki beberapa komponen dalam mengeksekusi tindakannya :

1. **Himpunan Kandidat (C)**
   Himpunan yang berisi seluruh kandidat yang akan dipertimbangkan pada setiap langkah algoritma.
   Contoh: simpul atau sisi dalam graf, pekerjaan (*job*), tugas (*task*), koin, benda, karakter, dan sebagainya.
2. **Himpunan Solusi (S)**
   Himpunan yang berisi kandidat-kandidat yang telah dipilih dan dianggap bagian dari solusi sementara.
3. **Fungsi Solusi**
   Fungsi yang menentukan apakah himpunan kandidat yang telah dipilih sudah membentuk solusi akhir yang valid atau lengkap.
4. **Fungsi Seleksi (Selection Function)**
   Fungsi yang memilih kandidat dari himpunan kandidat berdasarkan strategi greedy tertentu. Strategi greedy ini bersifat heuristik, yaitu berdasarkan pendekatan yang tampak paling menjanjikan pada saat itu.

5. **Fungsi Kelayakan (Feasibility Function)**
   Fungsi yang memeriksa apakah kandidat yang dipilih masih dapat dimasukkan ke dalam himpunan solusi tanpa melanggar batasan masalah.
6. **Fungsi Objektif (Objective Function)**
   Fungsi yang menggambarkan tujuan dari masalah, yaitu untuk memaksimumkan atau meminimumkan nilai tertentu dari solusi yang dibentuk.

II.II Cara Kerja Program dan Aksi Bot

Robocode Tank Royale merupakan game simulasi pertempuran tank yang ditujukan untuk mengembangkan dan menguji algoritma dalam bentuk *bot*. Saat game engine berjalan, arena pertempuran akan diinisialisasi, dan setiap *bot* akan dimasukkan ke dalam arena tersebut secara acak baik dari sisi posisi maupun arah hadap awalnya. Engine kemudian akan mengatur jalannya pertandingan, seperti pergerakan, penembakan, tabrakan, dan pengurangan energi berdasarkan interaksi antar bot.

Setiap bot bekerja secara independen dan dikontrol oleh algoritma yang ditulis oleh pengguna. Bot akan menerima *event* dari game engine seperti musuh terlihat (*OnScannedBot*), terkena tembakan (*OnHitByBullet*), bertabrakan dengan dinding atau bot lain, dan sebagainya. Berdasarkan event tersebut, fungsi Run() dan berbagai fungsi *event handler* lainnya akan menentukan gerakan, rotasi, dan aksi dari bot.

Game engine akan memanggil logika dari setiap bot dalam siklus yang berlangsung sangat cepat (biasanya setiap beberapa milidetik), dan semua aksi bot akan dieksekusi secara paralel oleh engine. Oleh karena itu, efisiensi algoritma bot menjadi penting agar bot dapat merespons secara cepat dan akurat terhadap perubahan di medan pertempuran.

II.III Cara Menjalankan Game Engine

Untuk menjalankan Game Engine, perlu untuk menyiapkan Starter Pack yang tersedia pada tautan ini. Lalu melakukan download jar dari “robocode-tankroyale-gui-0.30.0.jar”, yang merupakan game engine hasil modifikasi asisten. Terdapat berbagai template bot yang dapat menjadi inspirasi dan bisa didapatkan dengan Download dan Ekstrak “TemplateBot.zip” sebagai template bagi bot C#.

jar dapat dijalankan dengan klik 2 kali pada file tersebut atau, dengan command berikut:

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

II.IV Cara Membuat Bot dan Menjalankan Bot

Untuk membuat Bot baru, Anda bisa membuat folder dengan nama sesuai Bot anda lalu ikuti langkah berikut:

Membuat file json dengan keterangan seperti berikut :

```
{
  "name": "«nama kelompok»",
  "version": "1.0",
  "authors": [ "«anggota 1»", "«anggota 2»", "«anggota 3»"],
  "description": "Deskripsi strategi greedy yang digunakan",
  "homepage": "«…»",
  "countryCodes": [ "«enter your country code, e.g. id»" ],
  "platform": "«enter programming platform, e.g. Java or .Net»",
  "programmingLang": "C#"
}
```

Lalu buat source code dalam C# untuk bot Anda (”Misal BotTemplate.cs”) dan buat source code dengan kerangka seperti berikut:

BotTemplate.cs

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

public class BotTemplate : Bot
{
    // The main method starts our bot
    static void Main(string[] args)
    {
        new BotTemplate().Start();
    }

    // Constructor, which loads the bot config file
    BotTemplate() : base(BotInfo.FromFile("BotTemplate.json")) { }

    // Called when a new round is started -> initialize and do some
movement
    public override void Run()
    {

        BodyColor = Color.FromArgb(0x00, 0x00, 0x00);
        TurretColor = Color.FromArgb(0x00, 0x00, 0x00);
        RadarColor = Color.FromArgb(0x00, 0x00, 0x00);
        BulletColor = Color.FromArgb(0x00, 0x00, 0x00);
        ScanColor = Color.FromArgb(0x00, 0x00, 0x00);
        TracksColor = Color.FromArgb(0x00, 0x00, 0x00);
        GunColor = Color.FromArgb(0x00, 0x00, 0x00);

        // Repeat while the bot is running
        while (IsRunning)
        {
            // Write your bot logic
        }
    }
}
```

BotTemplate Ganti dengan nama bot anda

Lalu buat fle .csproj , .cmd, dan .sh agar bot dapat dimuat pada permainan:

BotTemplate.csproj

```
<Project Sdk="Microsoft.NET.Sdk">
    <PropertyGroup>
        <RootNamespace>BotTemplate</RootNamespace>
        <OutputType>Exe</OutputType>
```

```
        <TargetFramework>net6.0</TargetFramework>
        <LangVersion>10.0</LangVersion>
    </PropertyGroup>
    <ItemGroup>
        <PackageReference Include="Robocode.TankRoyale.BotApi"
Version="0.30.0"/>
    </ItemGroup>
</Project>
```

Ganti dengan nama bot anda

BotTemplate.cmd

```
@echo off
REM TemplateBot.cmd - Run the bot in development or release mode
REM Set MODE=dev for development (default, always rebuilds)
REM Set MODE=release for release (only runs if bin exists)

set MODE=dev

if "%MODE%"=="dev" (
    REM Development mode: always clean, build, and run
    rmdir /s /q bin obj >nul 2>&1
    dotnet build >nul
    dotnet run --no-build >nul
) else if "%MODE%"=="release" (
    REM Release mode: no rebuild if bin exists
    if exist bin\ (
        dotnet run --no-build >nul
    ) else (
        dotnet build >nul
        dotnet run --no-build >nul
    )
) else (
    echo Error: Invalid MODE value. Use "dev" or "release".
)
```

BotTemplate.sh

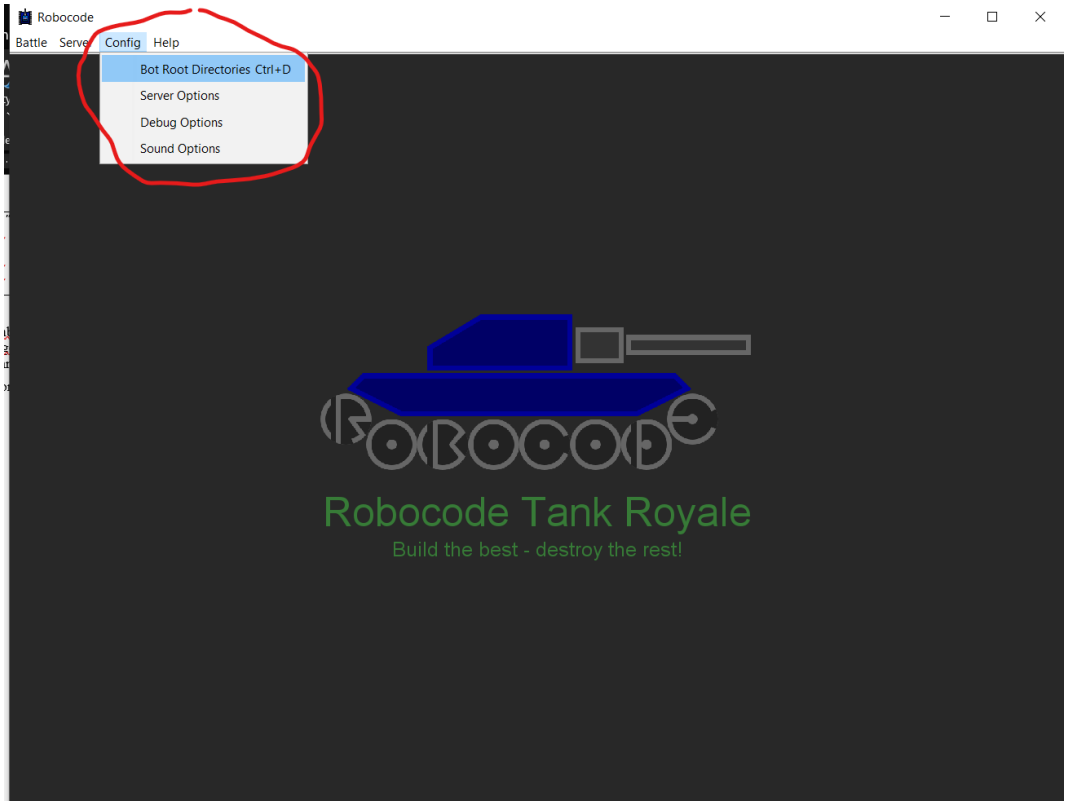
```
#!/bin/sh
# This script runs the bot in development mode by default.
# Development Mode (default): Always cleans, rebuilds, and runs.
# Release Mode (commented out): Runs without rebuilding.

# Development mode: always rebuild
rm -rf bin obj
dotnet build
dotnet run --no-build

# Uncomment below for release mode (runs without rebuilding)
# if [ -d "bin" ]; then
#   dotnet run --no-build
# else
#   dotnet build
#   dotnet run --no-build
# fi
```

Setelah membuat bot, anda bisa menjalankan bot dengan menjalankan game baik dengan mengeklik 2 kali pada file .jar yang di download atau dengan source command yang ada di **II.III.**

Sesuaikan config ketika game berjalan dengan directory bot yang Anda buat berada. Lalu, jalankan permainan dengan Battle dan muat bot yang ingin ditandingkan.



II.V Algoritma Greedy dalam Robocode Tank Royoale

Dalam permainan ini, untuk mencapai kemenangan diperoleh dengan mendapatkan skor setinggi mungkin. Skor tersebut dapa didapatkan dengan berbagai cara sebagai berikut:

- **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.

Dari berbagai cara tersebut, Bot dapat memaksimalkan poin dengan memilih kondisi yang optimal dalam permainan untuk mendapatkan poin sebanyak mungkin

Beberapa strategi *Greedy* yang bisa diterapkan:

- **Menyerang bot dengan HP rendah** untuk meningkatkan kemungkinan mendapatkan **Bullet Damage Bonus** atau **Ram Damage Bonus**.
- **Menghindari pertempuran langsung ketika musuh banyak** untuk meningkatkan **Survival Score** dan kesempatan meraih **Last Survival Bonus**.
- **Memanfaatkan ram attack (tabrakan) secara efektif** untuk mendapatkan **2x damage points** serta bonus tambahan jika musuh terbunuh.
- **Menjaga jarak dan memilih tembakan yang lebih akurat** agar setiap peluru memberikan damage yang maksimal dan meningkatkan peluang kill.
- **Menghindari peluru** agar tank bisa selamat sampai akhir dan energi yang digunakan cukup untuk melakukan aksi lain

Dengan pendekatan ini, bot dapat secara *greedy* memilih langkah terbaik dalam setiap situasi untuk memperoleh skor setinggi mungkin dan meningkatkan peluang menang dalam pertandingan.

Bab III. Strategi Greedy

III.I Pemetaan Umum Algoritma Greedy

Dalam Robocode Tank Royale, terdapat *Event* yang dapat dijadikan acuan dalam pengambilan keputusan optimal secara lokal yang ada di setiap langkah permainan. *Event* tersebut dapat dijadikan kandidat dalam melakukan mapping untuk menerapkan algoritma *Greedy*. Terdapat komponen-komponen yang dapat digunakan untuk menentukan *Event*, seperti musuh yang terdeteksi pada radar , posisi tank baik pribadi atau musuh, energi tank, dan kondisi permainan. Komponen tersebut dapat menjadi **Himpunan Kandidat** untuk menentukkan *Event* yang akan dipilih sebagai **Himpunan Solusi.** Dari himpunan kandidat ini, tank dapat menentukkan berbagai himpunan solusi seperti, arah tembakkan, arah pergerakan untuk menghindar, dan lain sebagainya.

Fungsi Solusi digunakan untuk mengeksekusi keputusan dari himpunan solusi seperti, mengunci pergerakan musuh yang dekat dengan radar, menembak jika posisi memungkinkan dan berbagai cara lain untuk eksekusi keputusan dari himpunan solusi. Proses memilih fungsi solusi tergantung pada **Fungsi Seleksi** dalam memilih himpunan kandidat. Bisa saja hanya memilih tank dengan energi paling sedikit untuk dikunci atau memilih untuk kabur karena posisi musuh terlalu dekat dengan posisi tank pribadi. Fungsi seleksi akan menentukkan elemen terbaik yang akan dimasukkan ke dalam himpunan solusi. Agar solusi tetap valid, maka diperlukan **Fungsi Kelayakan.** Fungsi kelayakan akan memastikan bahwa tindakan yang dipilih pada fungsi seleksi merupakan tindakan yang dapat dilakukan oleh tank. Misal, jika energi cukup dan posisi tank musuh dekat dengan derajat arah senjata, maka dapat dilakukan serangan untuk tank musuh.

Terakhir, **Fungsi Objektif** menentukkan tujuan utama dari strategi ini. Dalam permainan Robocode Tank Royale, poin akhir merupakan tujuan yang perlu dicapai dan untuk mencapainya dapat dilakukan dengan menembak musuh dan mengenainya atau bertahan selama mungkin dalam peta. Hal ini dapat dilakukan dengan memaksimalkan akurasi tembakkan, atau meminimkan penggunaan energi.

Untuk dapat lebih ringkas dan teknis dalam mengaitkan persoalan dalam di dalam Robocode Tank Royale terdapat beberapa properti yang dapat digunakan dalam permainan dalam tabel berikut:

No	Persoalan	Kegunaan	Fungsi/Proses
1.	Enemy (Musuh)	Target utama untuk mendapatkan poin dan bertahan dalam permainan	OnScannedBot()
2.	Bot Movement (Gerakan Bot)	Bot dapat berpindah posisi untuk melakukan strategi tertentu	Forward(), Backward(), SetTurnLeft(), SetTurnRight(), dsb...

3.	Firing (Menembak)	Menyerang musuh untuk mendapatkan poin dan energi jika terkena	Fire(), SetFire()
4.	Radar	Mengetahui posisi dan kondisi musuh melalui radar	OnScannedBot(), TurnRadarRight(), TurnRadarLeft(), ReScan(), dsb...
5.	On Hit by Bullet (Ditembak Musuh)	Tank dapat melakukan berbagai tindakan ketika mendapatkan tembakkan dari musuh	OnHitByBullet()
6.	On Hit Bot (Menyerang Musuh)	Ketika tank berhasil menyerang musuh, tank dapat melakukan berbagai tindakan untuk memaksimalkan strategi	OnHitBot()

III.II Eksplorasi Alternatif Solusi Greedy

Kelompok kami telah merancang empat alternatif solusi yang mengadopsi pendekatan algoritma greedy. Berikut adalah penjelasan rinci mengenai masing-masing algoritma greedy tersebut:

III.II.I. FlankerD - Strategi Flanking Agresif dengan Penembakan Prediktif

1) Mapping Elemen Greedy

- → **Himpunan kandidat**: Semua posisi potensial relatif terhadap musuh (depan, belakang, samping), semua kekuatan tembakan yang mungkin, dan semua arah pergerakan yang mungkin.
- → **Himpunan solusi**: Posisi flanking di belakang musuh, kekuatan tembakan optimal berdasarkan jarak, dan strategi menghindari serangan.
- → **Fungsi solusi**: Memeriksa apakah bot berada di posisi flanking yang optimal (di belakang musuh) dan dapat menembak dengan akurat.
- → **Fungsi seleksi**: Memilih posisi di belakang musuh sebagai titik target pergerakan dan menyesuaikan kekuatan tembakan berdasarkan jarak.
- → **Fungsi kelayakan**: Memastikan posisi target berada dalam batas arena, memeriksa jarak minimum dari dinding, dan memverifikasi bahwa sudut tembakan cukup presisi

→ **Fungsi objektif**: Memaksimalkan damage ke musuh dengan penembakan presisi sambil meminimalkan kemungkinan terkena serangan balik.

2) Analisis Efisiensi Solusi

FlankerD menggunakan pendekatan greedy yang efisien dengan selalu memilih posisi di belakang musuh sebagai posisi optimal. Kompleksitas waktu algoritma ini adalah O(1) untuk pengambilan keputusan posisi dan O(n) untuk prediksi tembakan, dengan n adalah jumlah iterasi penyempurnaan (bergantung pada jarak ke target).

Bot menggunakan memori minimal karena hanya menyimpan data posisi dan arah musuh saat ini tanpa perlu menyimpan historical data. Prediksi tembakan menggunakan metode iteratif sederhana yang meningkatkan akurasi tanpa menambah kompleksitas secara signifikan.

3) Analisis Efektivitas Solusi

FlankerD mengimplementasikan strategi greedy by position dengan selalu mengincar posisi di belakang musuh. Strategi ini sangat efektif karena memanfaatkan kelemahan musuh yang sulit menembak ke belakang dan memberikan keuntungan posisi untuk serangan presisi.

Strategi ini efektif jika:

- Musuh bergerak dengan pola yang dapat diprediksi, memungkinkan bot untuk konsisten berada di posisi belakang.
- Arena memiliki ruang yang cukup untuk manuver flanking tanpa terkena dinding.
- Musuh tidak sering mengubah arah atau memiliki kemampuan radar terbatas.
- Pertempuran terjadi dalam format 1v1 di mana fokus pada satu target optimal.

Strategi ini tidak efektif jika:

- Musuh menggunakan strategi berputar cepat yang menggagalkan upaya flanking.
- Terdapat banyak bot musuh yang menyerang dari berbagai arah secara bersamaan.
- Arena terlalu sempit sehingga manuver flanking terbatas oleh dinding.
- Musuh memiliki strategi khusus untuk mendeteksi dan menghadapi serangan dari belakang.

III.II.II. SuperSub - Memastikan Kehancuran semua Bot ketika Bot Tersisa Sedikit

1) Mapping Elemen Greedy

- → **Himpunan kandidat**: Berputar, menembak, mengubah arah, Jumlah musuh.
- → **Himpunan solusi**: Bergerak secara melingkar, Kondisi bot musuh tersisa sedikit, Menembak bot yang masuk ke radar
- → **Fungsi solusi**: Menembak bot, Mengubah arah rotasi
- → **Fungsi seleksi**: Mengecek apakah bot tersisa < 3, mengecek kondisi ketertembakkan bot.
- → **Fungsi kelayakan**: Menembak jika sudut < 10 derajat dari orientasi senjata, Berhenti mengejar jika jarak musuh < 150.
- → **Fungsi objektif**: Mengunci dan mendekati musuh tersisa (Memastikkan bot tersisa satu satunya), Menembak direntang sudut dekat, Bergant arah putar menghindari peluru musuh

2) Analisis Efisiensi Solusi

Aksi utama dari bot adalah berputar berlawanan arah jarum jam. Bot akan berputar dan mengubah ara/h putarnya sesaat jika tertembak oleh Musuh. Bot akan menembak jika ada musuh yang terdeteksi oleh radar dan sudut musuh yang terdeteksi kurang dari 10 derajat. Proses ketika permainan berlangsung tidak memiliki banyak pertimbangan sehingga bot lebih cepat untuk melakukan aksi baik mengubah arah putar atau menembak. Jika musuh yang tersisa sudah di bawah ambang batas (kurang dari 3), bot akan mengunci satu musuh dan mengincarnya hingga sedekat mungkin sembari menembak ke arah musuh tersebut. Dalam hal ini, bot akan memiliki peluang yang besar untuk menjadi satu-satunya yang tersisa pada permainan.

3) Analisis Efektivitas Solusi

SuperSub akan sangat diunggulkan pada kondisi ketika musuh yang tersisa sangat sedikit, tetapi cara ini akan berpeluang sangat baik jika Bot dapat *survive* dalam game dan memiliki energi yang cukup.

Strategi ini efektif jika :

- Musuh yang ada mayoritas merupakan musuh yang *greedy* pada poin tembakkan. Dengan banyak musuh yang menembak, kemungkinan musuh akan memiliki energi yang sedikit ketika akhir permainan.
- Ketika tidak ada musuh yang dapat mengunci tank. Semakin lama tank ini bertahan semakin besar kemungkinan *SuperSub* menang.

Strategi ini tidak efektif jika :

- Terdapat musuh yang dapat mengunci musuh dengan baik dari awal permainan.
- Posisi tank ketika awal permainan terlalu dekat dengan tank lain.

III.II.III. IkanLele - Mengunci Radar dan Arah Senjata dengan Posisi Musuh

1) **Mapping Elemen Greedy**

→ **Himpunan kandidat**: Musuh yang terdeteksi oleh radar, kondisi ketika tertembak, kondisi ketika merusak musuh (mengenai musuh dalam menembak/menabrak).
→ **Himpunan solusi**: Kombinasi terbaik dari target musuh yang terdeteksi dengan sudut dengan senjata, Menarget musuh yang menabrak tank.
→ **Fungsi solusi**: Mengunci target dan menembak dengan FirePower 3
→ **Fungsi seleksi**: Memastikan urgensi target dengan mengutamakan tank yang ditabrak.
→ **Fungsi kelayakan**: Memastikan bahwa musuh < 10 derajat dari orientasi senjata
→ **Fungsi objektif**: Memaksimalkan akurasi dengan mengunci dan menembak atau mengubah orientasi gerakan.

2) **Analisis Efisiensi Solusi**

Aksi utama dari *IkanLele* adalah berputar sembari mengunci musuh. Orientasi radar dan senjata akan mengunci pada musuh yang pertama kali terdeteksi di radar. Jika terdapat musuh yang menabrak *IkanLele*, tank ini akan menargetkan musuh yang menyerangnya dengan harapan tank ini dapat *survive* dari serangan musuh yang paling dekat. Tidak banyak pilihan dalam *IkanLele* sehingga eksekusi aksi akan lebih cepat. Kondisi mengunci radar dan senjata akan memaksimalkan potensi terkenanya tembakkan ke Musuh sehingga poin tembakkan akan maksimal.

3) **Analisis Efektivitas Solusi**

Tank ini termasuk ke dalam *greedy* dengan menembak. Tembakkan yang akan dieksekusi adalah tembakkan yang sudah dimaksimalkan. Tetapi, kemungkinan untuk tidak kena masih ada dan bot tidak mendapatkan aksi tambahan jika peluru tidak terkena musuh.

Strategi ini efektif jika:
- Mayoritas bot adalah bot yang gerakannya statis dan mudah diprediksi.

Strategi ini tidak efektif jika:

- Gerakan mayoritas bot cukup cepat dan acak.

III.II.IV. RimRunnerz - Strategi Orbit Berbasis Jarak Terdekat

1) **Mapping Elemen Greedy**
   - → **Himpunan kandidat**: Semua musuh yang terdeteksi oleh radar, semua arah orbit yang mungkin (searah/berlawanan jarum jam), semua daya tembak (0.5, 1.0, 2.0, 3.0), semua jarak orbit yang mungkin.
   - → **Himpunan solusi**: Aksi target dan orbit yang terpilih, termasuk jarak orbit, arah orbit, dan kekuatan tembakan yang optimal.
   - → **Fungsi solusi**: Memeriksa apakah aksi orbit dan penembakan dapat dilakukan terhadap musuh tertentu.
   - → **Fungsi seleksi**: Pilih musuh terdekat sebagai target untuk diorbit, kemudian pilih aksi yang memaksimalkan damage dengan mempertahankan jarak orbit optimal (ORBIT_DISTANCE = 150).
   - → **Fungsi kelayakan**: Memeriksa apakah musuh berada dalam jangkauan (MAX_RANGE = 250), target berada dalam batas arena, dan senjata dapat diarahkan dengan presisi ($\leq 3°$).
   - → **Fungsi objektif**: Memaksimalkan *damage* ke musuh terdekat sambil mempertahankan kelangsungan hidup melalui pergerakan orbit.

2) **Analisis Efisiensi Solusi**

   Rimrunnerz memilih musuh terdekat dengan kompleksitas O(n) di mana n adalah jumlah musuh. Proses pemilihan target cepat namun tidak selalu menghasilkan strategi optimal.

   Rimrunnerz hanya menyimpan informasi musuh terbaru (10 putaran terakhir), sehingga efisien dalam penggunaan memori. Pergerakan orbital dan tembakan prediktif memerlukan kalkulasi matematis yang cukup sederhana, menjadikan algoritma ini efisien secara komputasi.

3) **Analisis Efektivitas Solusi**

   Strategi orbit RimRunnerz bekerja dengan cara mengelilingi musuh terdekat pada jarak optimal sambil terus menembak. Bot secara aktif beradaptasi terhadap kondisi pertempuran dengan mengubah kekuatan tembakan dan jarak orbit berdasarkan energi yang tersisa dan jarak ke target. Ketika tidak ada musuh yang terdeteksi, bot bergerak secara acak untuk mencari musuh.

   Sistem orbit menggunakan pendekatan "berputar seperti bulan mengelilingi planet", memungkinkan bot tetap bergerak (sulit ditembak) sambil mempertahankan pandangan pada target. Bot juga menggunakan daya tembak tinggi saat energi berlimpah dan beralih ke mode bertahan hidup saat energi menipis.

Strategi ini efektif jika:
- Arena memiliki sedikit musuh yang tersebar (terutama untuk duel 1v1).
- Target berada pada jarak menengah yang memungkinkan orbit optimal.
- Bot memiliki cukup ruang untuk melakukan pergerakan orbital.

Strategi ini tidak efektif jika:

- Banyak musuh dalam jarak dekat secara bersamaan.
- Medan perang sangat terbatas dengan banyak dinding atau rintangan.
- Musuh menggunakan taktik khusus anti-orbit atau penembakan prediktif yang canggih.

III.III Strategi Greedy yang Dipilih

Strategi yang pada akhirnya dipilih oleh kelompok kami adalah FlankerD yang mengimplementasikan strategi greedy by position. Strategi greedy by position adalah strategi greedy yang mengutamakan penempatan posisi relatif optimal terhadap musuh. Strategi ini dinilai paling efektif untuk permainan Robocode Tank Royale karena memberikan keuntungan taktis yang signifikan dalam pertarungan tank.

FlankerD mengutamakan posisi di belakang musuh, yang merupakan area "buta" yang membuat musuh sulit membalas serangan. Strategi ini memungkinkan bot untuk memaksimalkan damage dan meminimalkan kerusakan yang diterima, sehingga meningkatkan peluang untuk bertahan lebih lama dan memenangkan pertandingan.

Pengimplementasian strategi greedy by position memungkinkan bot untuk beradaptasi dengan cepat terhadap pergerakan musuh. Bot selalu berusaha mempertahankan posisi optimalnya meskipun musuh bergerak atau mengubah arah. Hal tersebut memungkinkan bot untuk mengambil keputusan yang efektif di berbagai situasi pertempuran, terutama dengan sistem pertarungan Robocode dimana penempatan yang baik sering kali menjadi kunci kemenangan.

Bab IV. Implementasi dan Pengujian

IV.I Struktur Data Program

```
TUBES1_RAYAPBESI
├── docs
├── src
│   └── alternative-bots
│       ├── FlankerD
│       │   ├── bin
│       │   ├── obj
│       │   ├── FlankerD.cmd
│       │   ├── FlankerD.cs
│       │   ├── FlankerD.csproj
│       │   ├── FlankerD.json
│       │   └──  FlankerD.sh
│       │
│       ├── IkanLele
│       │   ├── bin
│       │   ├── obj
│       │   ├── IkanLele.cmd
│       │   ├── IkanLele.cs
│       │   ├── IkanLele.csproj
│       │   ├── IkanLele.json
│       │   └── IkanLele.sh
│       │
│       ├── RimRunnerZ
│       │   ├── bin
│       │   ├── obj
│       │   ├── RimRunnerZ.cmd
│       │   ├── RimRunnerZ.cs
│       │   ├── RimRunnerZ.csproj
│       │   ├── RimRunnerZ.json
│       │   └── RimRunnerZ.sh
│       │
│       └── SuperSub
│           ├── bin
│           ├── obj
│           ├── SuperSub.cmd
│           ├── SuperSub.cs
│           ├── SuperSub.csproj
│           ├── SuperSub.json
│           └── SuperSub.sh
│
├── .gitignore
├── LICENSE
└── README.md
```

IV.II Implementasi dari Bot FlankerD

```
Function Run():
    BodyColor ← Purple

    While IsRunning Do
        TurnLeft(BearingTo(ArenaWidth * 0.5, ArenaHeight *
0.5))
        TurnRadarLeft(360)
```

```
    End While
End Function

Function OnScannedBot(event e):
    flankDistance ← 100

    SetTurnRadarLeft(RadarBearingTo(e.X, e.Y) * 1.5)

    If e.Speed < 0 Then
        targetDirection ← (e.Direction + 180) % 360
    Else
        targetDirection ← e.Direction
    End If
    targetDirection ← targetDirection * (π / 180)

    xFlank ← e.X - cos(targetDirection) * flankDistance
    yFlank ← e.Y - sin(targetDirection) * flankDistance

    flankBearing ← BearingTo(xFlank, yFlank)
    SetTurnLeft(flankBearing)

    SetForward(Min(DistanceTo(xFlank, yFlank), DistanceTo(e.X,
e.Y) - 70))

    If RadarDirection - e.Direction ≤ 90 OR RadarDirection -
e.Direction ≥ 270 Then
        If DistanceTo(e.X, e.Y) ≤ 1 * flankDistance Then
            SetTurnLeft(CalcBearing(RadarDirection + 120))
        Else If DistanceTo(e.X, e.Y) ≤ 2 * flankDistance Then
            SetTurnLeft(CalcBearing(RadarDirection + 90))
        Else
            SetTurnLeft(CalcBearing(RadarDirection + 45))
        End If
    End If

    If X < 150 OR X > ArenaWidth - 150 OR Y < 150 OR Y >
ArenaHeight - 150 Then
        SetTurnLeft(BearingTo(ArenaWidth * 0.5, ArenaHeight *
0.5))
    End If

    firepower ← 3 * flankDistance / DistanceTo(e.X, e.Y)

    LeadFire(e, firepower)
End Function

Function LeadFire(event e, firepower):
    targetDistance ← DistanceTo(e.X, e.Y)
    targetSpeed ← |e.Speed|

    If e.Speed < 0 Then
        targetDirection ← (e.Direction + 180) % 360
    Else
        targetDirection ← e.Direction
    End If
    targetDirection ← targetDirection * (π / 180)
```

```
    bulletSpeed ← CalcBulletSpeed(firepower)
    deltaTime ← targetDistance / bulletSpeed

    xLead ← e.X + cos(targetDirection) * targetSpeed *
deltaTime
    yLead ← e.Y + sin(targetDirection) * targetSpeed *
deltaTime

    i ← Floor(targetDistance / 100)
    While i > 0 Do
        targetDistance ← DistanceTo(xLead, yLead)
        deltaTime ← targetDistance / bulletSpeed
        xLead ← e.X + cos(targetDirection) * targetSpeed *
deltaTime
        yLead ← e.Y + sin(targetDirection) * targetSpeed *
deltaTime
        i ← i - 1
    End While

    SetTurnGunLeft(GunBearingTo(xLead, yLead))

    If |GunBearingTo(xLead, yLead)| < 5 Then
        Fire(firepower)
    End If
End Function

Function OnHitBot(event e):
    SetTurnRadarLeft(RadarBearingTo(e.X, e.Y) * 1.2)
End Function
```

Pada tank *FlankerD*, tank akan mengincar bagian belakang tank musuh yang terdeteksi. Jika tank terlalu jauh, maka tank akan mendekat sembari berputar untuk mendapatkan posisi belakang musuh.

FlankerD juga memprediksi arah gerak musuh untuk memaksimalkan poin penembakkan kepada tank musuh. Dalam hal ini, *FlankerD* akan memilih strategi *greedy* dengan menembak. Karena posisi *FlankerD* yang berubah dan berputar, akan menyulitkan lawan untuk menyerah *FlankerD*. Maka dari itu, tank ini juga memanfaatkan *greedy by survive*.

IV.III Implementasi dari Bot IkanLele

```
FUNCTION OnScannedBot(event):
    // Kalkulasi Radar
    radarTurn ← NormalizeRelativeAngle(RadarBearingTo(event.X,
event.Y))
    enemyDistance ← DistanceTo(event.X, event.Y)

    extraTurn ← MIN(ATAN(36.0 / enemyDistance) * (180 / PI),
MaxRadarTurnRate)

    // Kalkulasi Senjata
    IF event.Direction < 0 THEN
        enemyDirection ← (event.Direction + 180) % 360
```

```
    ELSE
        enemyDirection ← event.Direction
    ENDIF

    bulletSpeed ← CalcBulletSpeed(dmg)
    timeImpact ← enemyDistance / bulletSpeed
    futureX ← event.X + COS(enemyDirection) * event.Speed *
timeImpact
    futureY ← event.Y + SIN(enemyDirection) * event.Speed *
timeImpact
    gunTurn ← NormalizeRelativeAngle(GunBearingTo(futureX,
futureY))

    radarTurn ← radarTurn + (extraTurn IF radarTurn > 0 ELSE -
extraTurn)

    SetTurnRadarLeft(radarTurn)
    SetTurnGunLeft(gunTurn)

    IF ABS(GunBearingTo(event.X, event.Y)) < 10 THEN
        IF Energy > 20 THEN
            SetFireAssist(TRUE)
            Fire(dmg)
        ELSE IF ABS(GunBearingTo(event.X, event.Y)) < 5 THEN
            dmg ← 2
            SetFireAssist(TRUE)
            Fire(dmg)
        ENDIF
    ENDIF
END FUNCTION
```

Dalam bot *IkanLele*, bot akan mengunci bot yang terdeteksi di radar dalam variabel *radarTurn*. *radarTurn* mendapatkan koordinat X dan Y dari musuh yang terdeteksi, lalu koordinat tersebut akan dijadikan sudut yang ditempuh dari orientasi radar saat ini.

Kalkulasi arah dan kerusakan senjata juga dilakukan ketika keadaan *OnScannedBot()* terjadi. Karena bot akan selalu terpindai oleh radar, maka arah senjata juga akan menyesuaikan posisi yang didapatkan dari radar.

Posisi senjata akan disesuaikan dengan orientasi dan kecepatan dari tank yang terpindai. maka dari itu digunakan X dan Y yang akan diperkirakan oleh tank sebagai sudut untuk memutar senjata.

NormalizeRelativeAngle() akan memberikan sudut dalam rentang [-180,180]. Dengan ini, rotasi baik dari senjata, maupun radar akan diputar ke sudut yang paling dekat.

Dengan kondisi ini, diharapkan tank dapat mengunci musuh, memprediksi musuh, sehingga tank dapat menembakkan peluru lebih akurat.

```
FUNCTION OnHitBot(event):
    bearing ← BearingTo(event.X, event.Y)

    IF bearing > -10 AND bearing < 10 THEN
```

```
        Fire(3)
    ENDIF

    IF event.IsRammed THEN

SetTurnRadarLeft(NormalizeRelativeAngle(RadarBearingTo(event.X,
event.Y)))

SetTurnGunLeft(NormalizeRelativeAngle(GunBearingTo(event.X,
event.Y)))
        SetTurnLeft(10)
    ENDIF
END FUNCTION

FUNCTION OnHitByBullet(event):
    IF isLeft THEN
        TurnRight(NormalizeRelativeAngle(90   -   (Direction   -
event.Bullet.Direction)))
        isLeft ← FALSE
    ELSE
        TurnLeft(NormalizeRelativeAngle(90   -   (Direction   -
event.Bullet.Direction)))
        isLeft ← TRUE
    ENDIF
END FUNCTION
```

Ketika musuh menabrak tank, hal ini menandakan adanya tank yang mencoba mengunci *IkanLele*, maka dari itu perlu untuk mengunci tank yang paling membahayakan tank sendiri

Jika terkena tembakkan, *IkanLele* akan mengubah orientasi putarnya untuk mengecoh pergerakan.

Pada tank *IkanLele*, **strategi *greedy* yang paling menonjol adalah *greedy* untuk menembak**. Seperti yang dijelaskan pada **II.IV**, ketika menembak akan mendapatkan poin dan ketika musuh hancur oleh tembakkan, maka akan mendapatkan poin bonus 20%. Hal ini yang diutamakan oleh bot *IkanLele*. Untuk memaksimalkan poin tersebut, dilakukan dengan mengunci, dan memprediksi target yang ada pada permainan.

IV.IV Implementasi dari Bot SuperSub

```
Function Run():
    # Set warna tampilan bot
    BodyColor ← Gray
    RadarColor ← Red
    ScanColor ← Yellow

    # Inisialisasi variabel kontrol
    movingForward ← True
    isLeft ← True
    stop ← False
    dmg ← 3

    # Loop utama selama bot masih berjalan
    While IsRunning Do
        If stop = False Then
```

```
            MaxSpeed ← 5  # Batasi kecepatan maksimum
            SetForward(40_000)  # Maju ke depan dalam jarak
sangat jauh
            SetTurnLeft(40_000)  # Terus berputar ke kiri dalam
jarak sangat jauh
        Else
            Stop()  # Hentikan pergerakan
        End If

        Go()  # Jalankan aksi bot dalam loop
    End While
End Function

Function OnScannedBot(event e):
    # Hitung sudut ke musuh
    bearing ← BearingTo(e.X, e.Y)

    # Hitung jarak ke musuh
    enemyDistance ← DistanceTo(e.X, e.Y)

    # Jika hanya tersisa 2 musuh atau kurang, dan energi bot
cukup tinggi
    If EnemyCount ≤ 2 AND Energy > 30 Then
        SetFireAssist(True)  # Aktifkan bantuan tembakan
        SetTurnLeft(bearing * 10_000)  # Mengunci ke arah musuh
secara agresif

        # Jika musuh cukup dekat, hentikan pergerakan
        If enemyDistance < 150 Then
            stop ← True
        Else
            SetForward(dist)  # Bergerak maju sejauh `dist`
            stop ← False
        End If
    End If

    If EnemyCount = 1 Then
        SetFireAssist(True)  # Aktifkan bantuan tembakan
        SetTurnLeft(bearing * 10_000)  # Mengunci ke arah musuh
secara agresif

        # Jika musuh cukup dekat, hentikan pergerakan
        If enemyDistance < 150 Then
            stop ← True
        Else
            SetForward(dist)  # Bergerak maju sejauh `dist`
            stop ← False
        End If
    End If


    # Jika energi kurang dari 60, kurangi kekuatan tembakan
    If Energy < 60 Then
        dmg ← 2
    End If
```

```
    # Tembak dengan kekuatan `dmg`
    Fire(dmg)
End Function
```

Pada bot ini, utamanya bot akan berputar dan menembak jika mendeteksi musuh. Kenapa memilih gerakkan seperti ini? karena arah putaran justru lebih mudah mengecoh lawan dan lebih mudah untuk menembakkan senjata dibanding dengan memberikan rangkaian gerakkan acan pada tank. Dengan hal itu, diharapkan tank dapat bertahan sembari merusak musuh

Strategi *greedy* pada tank ini akan terlihat lebih jelas ketika musuh yang tersisa kurang dari 3. *SuperSub* akan mengincar dan mendekat ke musuh tersebut sembari menembak. Dengan melakukan strategi tersebut, tank berkemungkinan besar untuk menjadi tank yang terakhir bertahan ketika ronde selesai. Seperti yang dikeathui pada **II.IV**. Terdapat poin ketika musuh mati dan menjadi satu-satunya yang tersisa. Hal ini yang akan dimaksimalkan dalam *SuperSub*. Strategi *greedy* yang dipilih oleh tank ini adalah **menjadi satu-satunya yang tersisa**.

IV.V Implementasi dari Bot RimRunnerz

```
Function SpinOrbitTarget():
    # Hitung jarak ke target
    distance ← DistanceTo(targetX, targetY)

    # Jika target terlalu jauh, batalkan
    If distance > MAX_RANGE Then return

    # Hitung sudut untuk penembakan prediktif
    absoluteBearing ← DirectionTo(targetX, targetY)
    gunBearing ← GunBearingTo(targetX, targetY)

    # Prediksi posisi target berdasarkan kecepatan
    If targetVelocity > 0 Then
        bulletPower ← 2.0
        bulletSpeed ← 20 - 3 * bulletPower
        timeToHit ← distance / bulletSpeed

        futureX ← targetX + sin(targetHeading) * targetVelocity
* timeToHit
        futureY ← targetY + cos(targetHeading) * targetVelocity
* timeToHit

        gunBearing ← GunBearingTo(futureX, futureY)
    End If

    SetTurnGunLeft(gunBearing)

    # Hitung sudut orbit (tegak lurus terhadap target)
    If orbitClockwise Then
        orbitAngle ← absoluteBearing + 90
    Else
        orbitAngle ← absoluteBearing - 90
    End If

    # Bergerak dalam orbit
```

```
    SetTurnLeft(NormalizeAngle(orbitAngle - Direction))

    # Jaga jarak optimal
    If distance > ORBIT_DISTANCE + 20 Then
        SetForward(distance - ORBIT_DISTANCE)
    Else If distance < ORBIT_DISTANCE - 20 Then
        SetBack(ORBIT_DISTANCE - distance)
    Else
        SetForward(50)
    End If

    # Tembak jika posisi tepat dan target valid
    If |gunBearing| ≤ 3 AND GunHeat = 0 Then
        # Sesuaikan kekuatan berdasarkan energi dan jarak
        firePower ← 2.0  # Default

        If Energy > 50 Then
            If distance < CLOSE_RANGE Then firePower ← 3.0
            Else If distance < MAX_RANGE Then firePower ← 2.0
            Else firePower ← 1.0
        Else If Energy > LOW_ENERGY Then
            If distance < CLOSE_RANGE Then firePower ← 2.0
            Else firePower ← 1.0
        Else
            firePower ← 0.5
        End If

        Fire(firePower)
    End If
End Function
```

Strategi *greedy by distance* menjadi inti dari pengambilan keputusan pada bot RimRunnerz. Bot selalu mengorbit musuh terdekat dan menyesuaikan perilakunya berdasarkan jarak dan energi.

Pendekatan *greedy* yang memilih musuh terdekat dan jarak orbit optimal membuat bot dapat mengambil keputusan secara cepat. Sistem adaptasi energi memastikan bot tetap efektif bahkan saat energi rendah. Kombinasi pergerakan orbit dan penembakan prediktif memaksimalkan peluang bot untuk mendapatkan skor tinggi dengan merusak musuh sambil tetap bertahan.

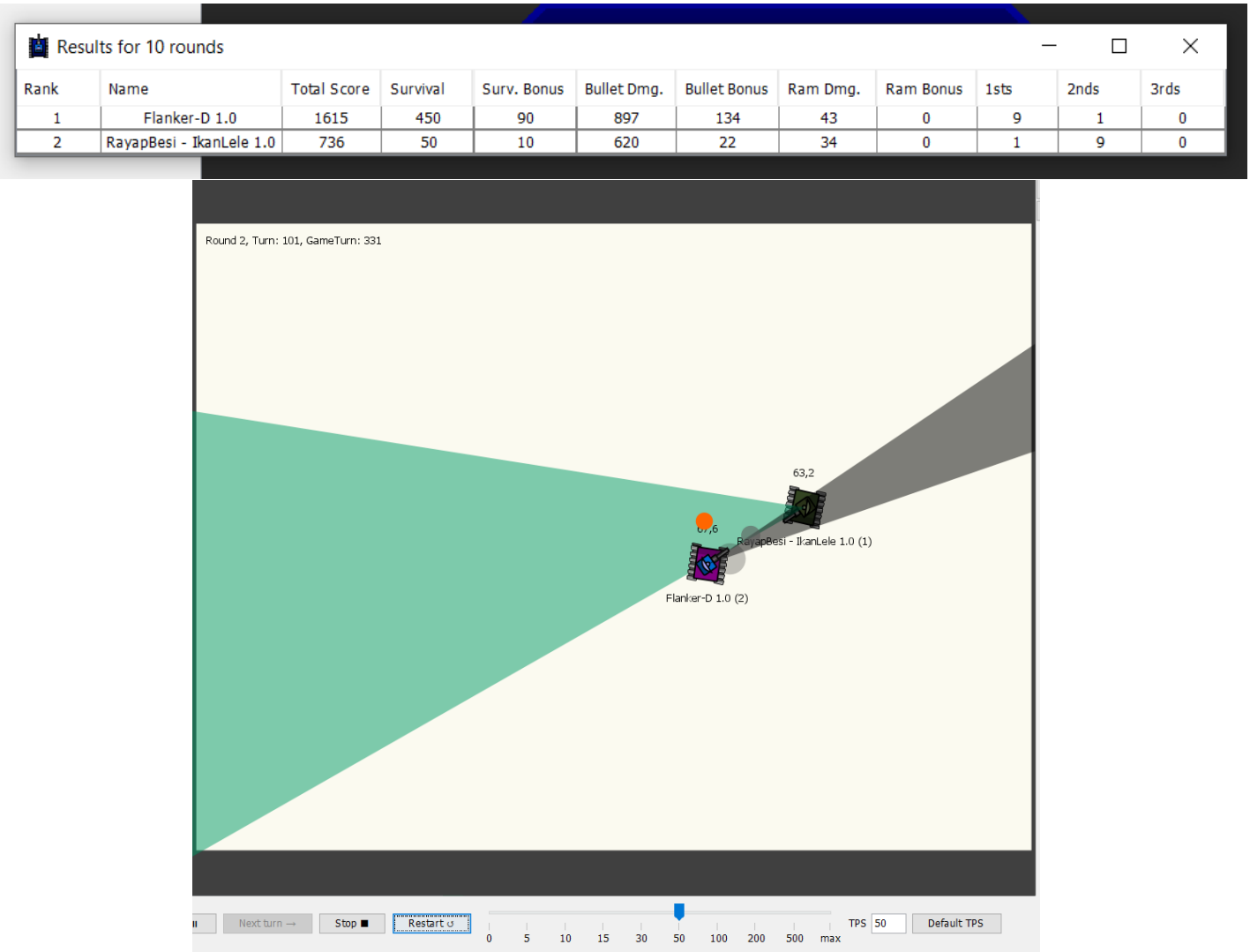
IV.VI Analisa Permainan

Tank yang dipilih oleh kelompok kami adalah tank *FlankerD*. Tank ini dipilih karena kemampuannya dalam memaksimalkan *greedy* yang ada pada **II.V**, yakni mengunci dan memprediksi arah musuh melalui titik lemah dan baik dalam menghindari peluru.

FlankerD akan mengincar bagian belakang musuh. Dengan mengincar posisi belakang tersebut diharapkan *FlankerD* dapat dengan baik menyerang musuh. Selain itu, posisi *FlankerD* yang dekat dengan musuh dapat memaksimalkan akurasi tembakkan.

Untuk dapat mengetahui kelemahan dari *FlankerD*, kami akan melakukan percobaan untuk melawan *FlankerD* dengan semua bot yang kami buat dengan metode 1 vs 1.

FlankerD vs IkanLele

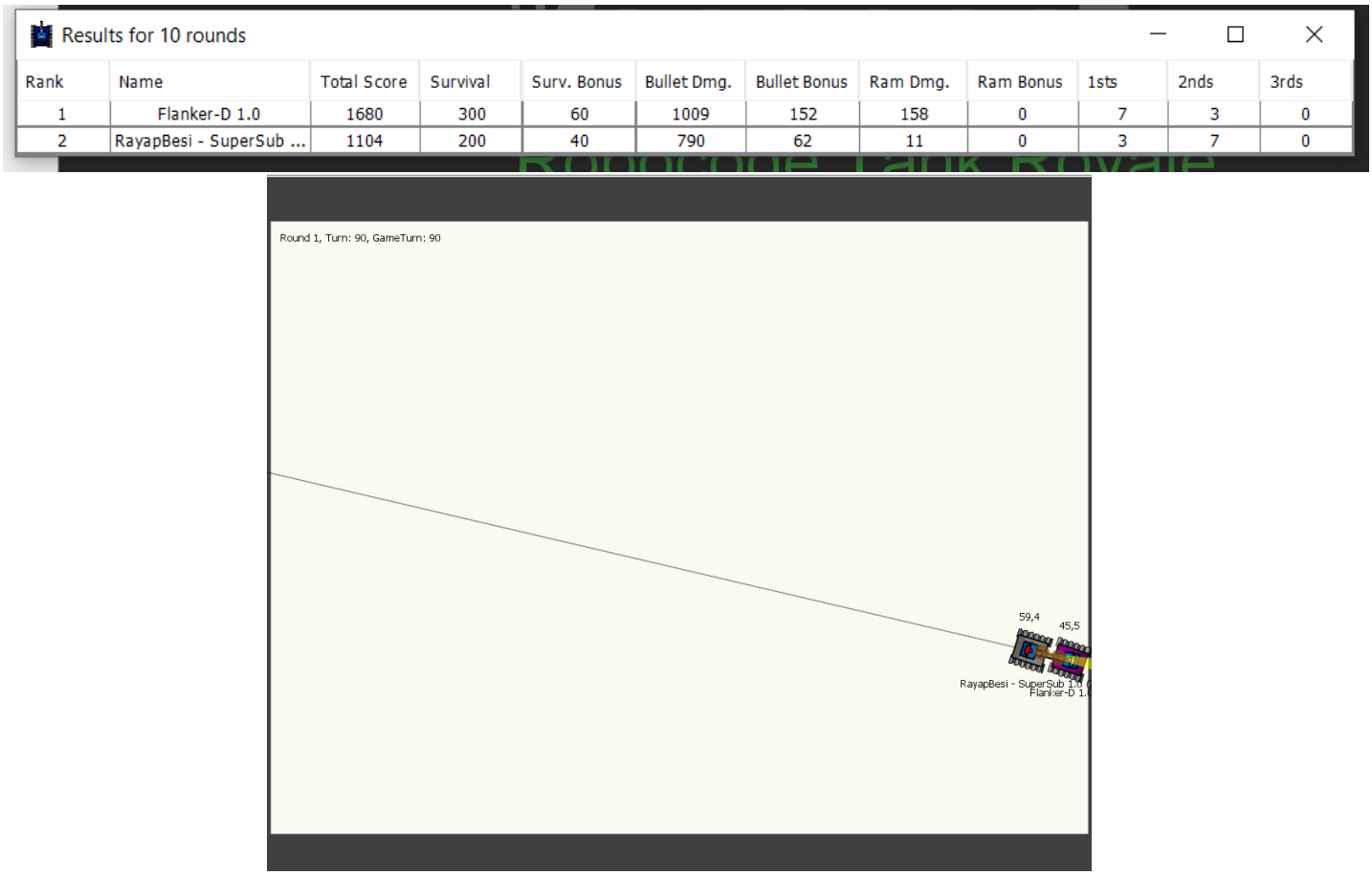


Gambar 4.1. Pertarungan 1 vs 1 antara FlankerD dan IkanLele

FlankerD bisa dengan mudah melawan *IkanLele*, hal ini karena gerakkan *IkanLele* yang cenderung semi-statis sehingga mudah untuk diserang oleh *FlankerD*. Gerakkan *FlankerD* juga cukup sulit diprediksi oleh *IkanLele* sehingga nilai kebertahanan yang dimiliki oleh *FlankerD* cukup baik.

Akurasi yang dimiliki oleh *FlankerD* lebih baik dibandingkan dengan *IkanLele*. Hal ini mungkin karena *FlankerD* mengincar posisi yang dekat dan posisi belakang *IkanLele* sehingga *IkanLele* lebih susah untuk menanganinya.

FlankerD vs SuperSub

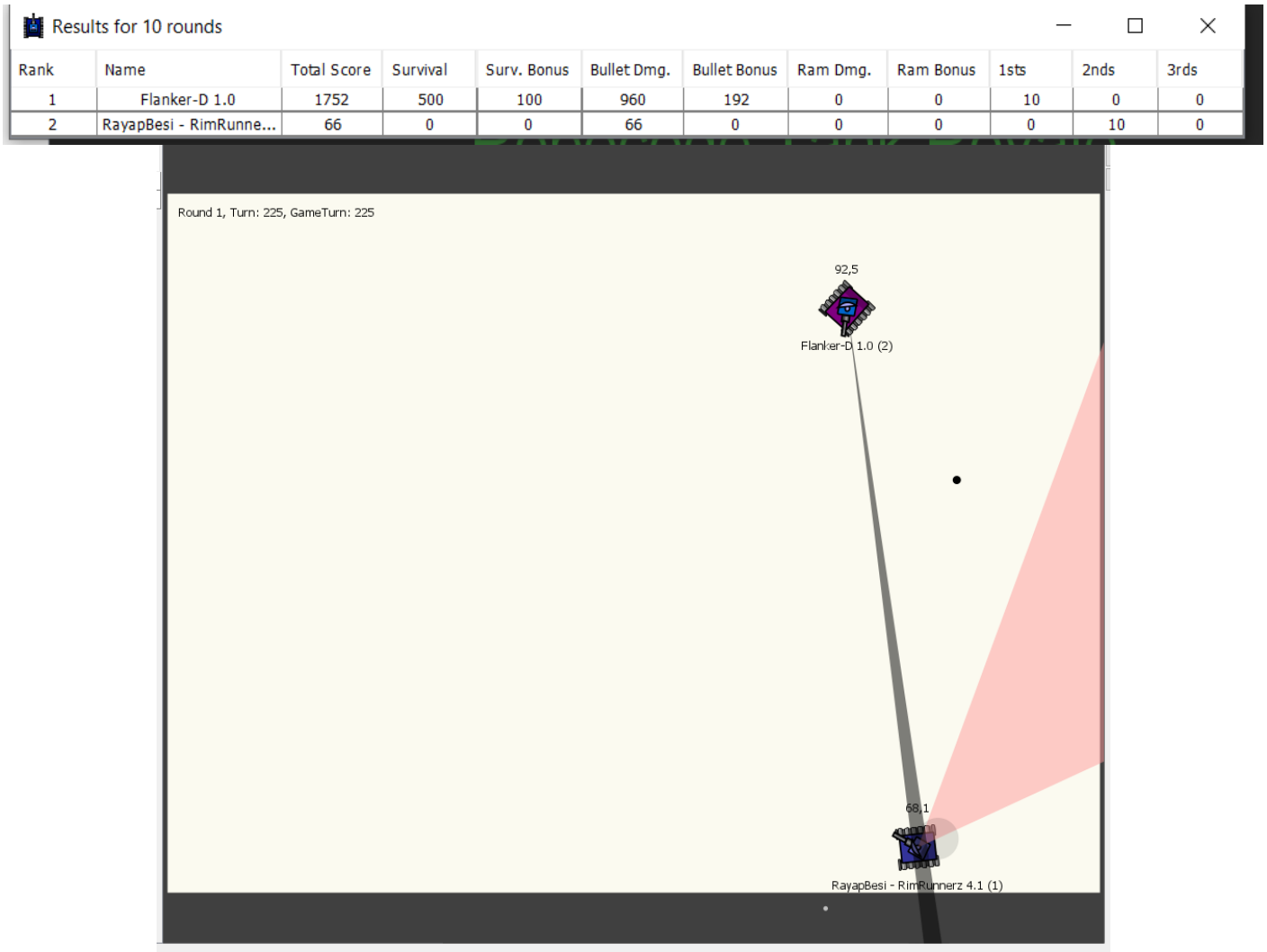


Gambar 4.2. Pertarungan 1 vs 1 antara FlankerD dan SuperSub

Ketika berhadapan dengan *SuperSub*, *FlankerD* cukup kewalahan dalam menghadapinya. Seperti yang diketahui, *SuperSub* memiliki strategi agresif dengan mengunci target dan berusaha semaksimal mungkin untuk menjadi satu-satunya bot yang tersisa. Dengan sistem pelacakan yang ketat, *SuperSub* mampu mempertahankan tekanan terus-menerus pada lawannya, membuat *FlankerD* sulit untuk menghindari serangan secara langsung. Dalam kondisi tertentu, *SuperSub* bahkan dapat mengendalikan ritme pertempuran dan membatasi ruang gerak *FlankerD*, menjadikannya lawan yang sangat tangguh.

Namun, ada momen ketika *FlankerD* berhasil keluar dari kuncian *SuperSub* dan memutar balik keadaan. *SuperSub* memiliki kelemahan saat menghadapi lawan yang memiliki kecepatan lebih tinggi dibandingkan dengan rotasi tubuhnya. Dengan kelincahannya, *FlankerD* dapat mencari celah untuk keluar dari tekanan, bergerak ke sisi belakang *SuperSub*, dan mengambil posisi serangan yang lebih menguntungkan. Meski menghadapi kesulitan di awal, strategi pergerakan cepat dan serangan balik dari *FlankerD* membuatnya masih cukup efektif dalam menangani ancaman dari *SuperSub*.

FlankerD vs RimRunnerz



Gambar 4.3. Pertarungan 1 vs 1 antara FlankerD dan RimRunnerz

RimRunnerZ selalu berusaha menjaga jarak dari musuhnya dengan terus bergerak menjauhi lawan. Strategi ini bertujuan untuk menghindari tembakan langsung dan memperpanjang durasi pertarungan. Namun, meskipun tampaknya efektif dalam bertahan, pola gerakan *RimRunnerZ* yang cukup teratur dan mudah ditebak menjadi kelemahan utamanya.

FlankerD memanfaatkan kelemahan ini dengan sangat baik. Dengan algoritma yang mampu membaca pola gerakan lawan, *FlankerD* dapat memprediksi pergerakan *RimRunnerZ* dan menyesuaikan arah tembakan dengan akurat. Akurasi tembakannya meningkat seiring waktu, terutama karena RimRunnerZ tidak memiliki mekanisme penghindaran yang cukup adaptif.

Selain itu, *FlankerD* tidak hanya mengandalkan akurasi serangan tetapi juga memiliki strategi untuk mempersempit ruang gerak lawan. Dengan manuver

flanking yang agresif, ***FlankerD*** dapat memaksa *RimRunnerZ* ke sudut yang sulit untuk melarikan diri, sehingga meningkatkan peluang kemenangan dalam pertarungan.

Meskipun *RimRunnerZ* memiliki keunggulan dalam bertahan, strategi ini tidak cukup efektif ketika menghadapi lawan yang mampu mengantisipasi pergerakannya. Untuk meningkatkan performa, *RimRunnerZ* perlu mengembangkan pola gerakan yang lebih dinamis dan tidak dapat diprediksi, serta mempertimbangkan taktik serangan balik untuk menghadapi lawan seperti *FlankerD*.

Bab V. Kesimpulan dan Saran

V.I. Kesimpulan

Dengan ini, kami berhasil membuat strategi bot untuk permainan Robocode Tank Royale dengan pendekatan algoritma *greedy*. Algoritma *greedy* bisa menjadi alternatif untuk menyelesaikan suatu persoalan. Namun, tidak semua persoalan dapat dijalankan dengan baik dan banyak sekali *heuristic* yang dapat digunakan untuk mengimplementasikan algoritma *greedy*. Dalam kasus ini, kami telah mencoba berbagai *heuristic* yang digunakan dan mendapatkan kesimpulan bahwa algoritma yang dimiliki oleh *FlankerD* lebih baik dibandingkan dengan tank lain. Hal ini dapat dijelaskan karena walaupun *greedy* mencoba untuk mencari kondisi optimal, mungkin saja optimal itu bukan optimal global sehingga diperlukan pencarian yang lain.

V.II. Saran

Saran untuk kelompok tugas besar ini, diharapkan config serta tatanan yang digunakan dalam membuat Robocode Tank Royale yang diberikan asisten lebih baik lagi. Kerap terjadi error untuk *path* tertentu dan perlu disesuaikan dengan *device* masing-masing orang. Saran untuk kelompok, akan lebih optimal jika pengerjaan dilakukan lebih dulu dan disiplin dalam melaksanakan tugas sesuai dengan *timeline* yang telah disediakan.

Lampiran

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

Pranala Repositori:

https://github.com/Darsua/Tubes1_RayapBesi

Daftar Pustaka

Munir, R. (2025). *Algoritma Greedy - Bagian 1* [PDF]. Institut Teknologi Bandung. Diakses dari https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2024-2025/04-Algoritma-Greedy-(2025)-Bag1.pdf