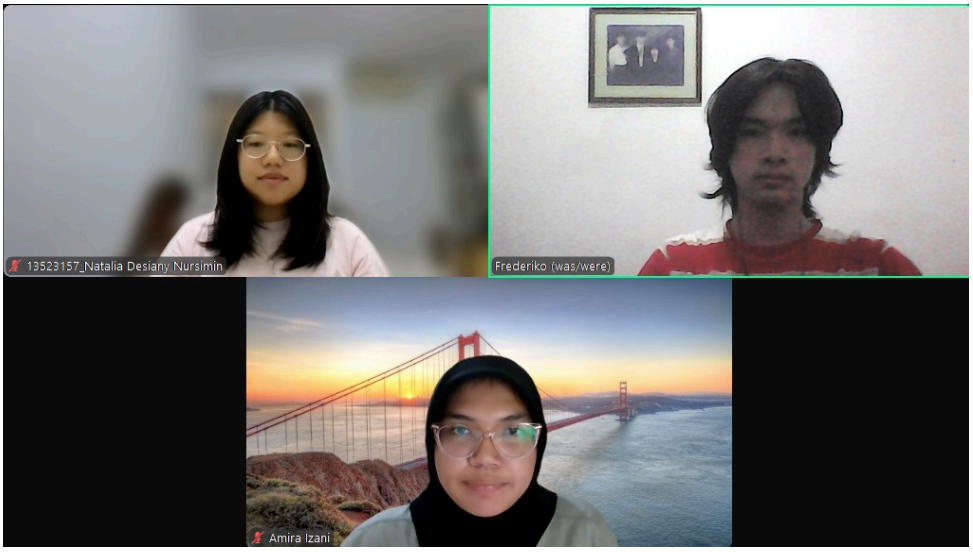
LAPORAN TUGAS BESAR 1
IF2211 STRATEGI ALGORITMA



Dipersiapkan oleh:

Kelompok 51

Amira Izani	13523143
Frederiko Eldad Mugiyono	13523147
Natalia Desiany Nursimin	13523157

Program Studi Teknik Informatika

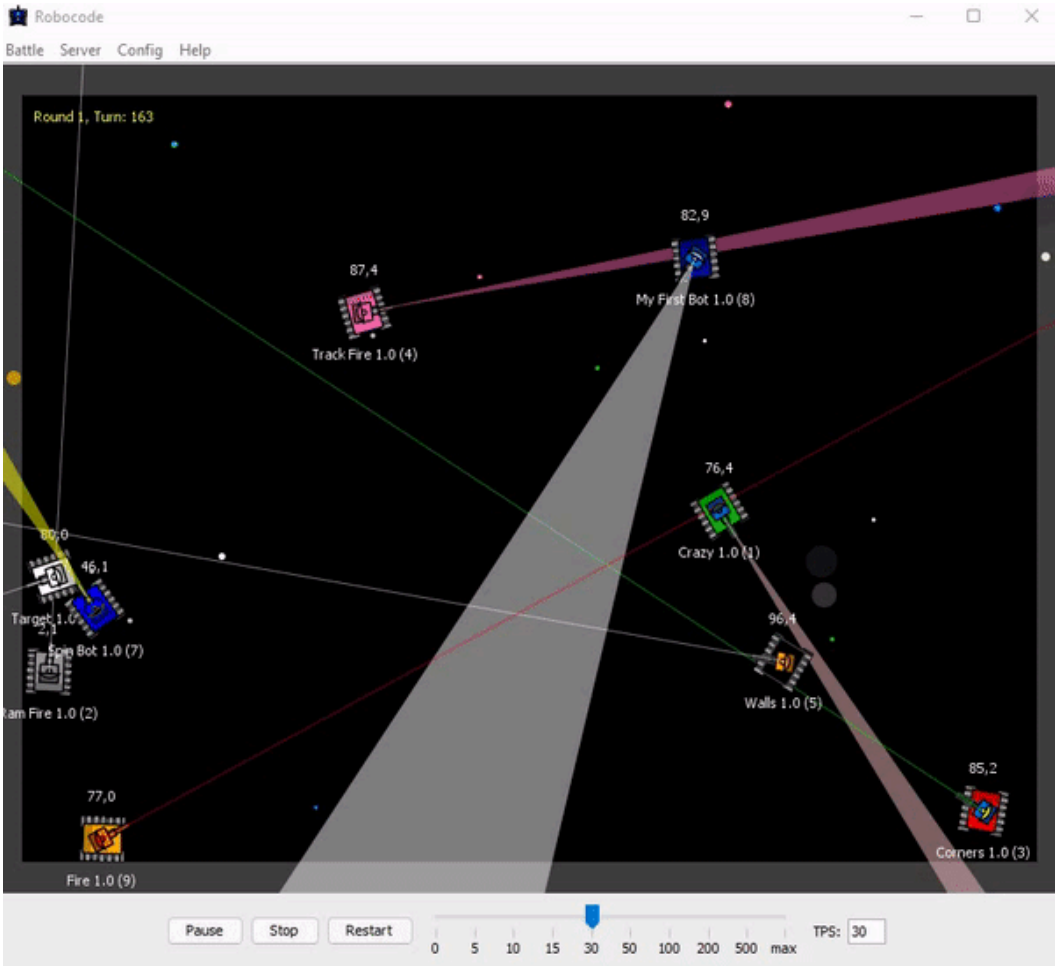
Sekolah Teknik Elektro dan Informatika

Institut Teknologi Bandung

2025

DAFTAR ISI

BAB 1 DESKRIPSI TUGAS



Gambar 1 Robocode Tank Royale

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

bot bergerak, mendeteksi bot lawan, menembakkan senjatanya, serta bagaimana bot bereaksi terhadap berbagai kejadian selama pertempuran.

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

Komponen-komponen dari permainan ini antara lain:

1. Rounds dan Turns

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

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

   Pada setiap turn, sebuah bot dapat:

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

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

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

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

2. Batas Waktu Giliran

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

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

3. Energi

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

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

4. Peluru

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

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

5. Panas Meriam (Gun Heat)

   Saat menembakkan peluru, meriam akan menjadi panas. Peluru yang lebih berat menghasilkan lebih banyak panas dibandingkan peluru yang lebih ringan. Ketika meriam terlalu panas, bot tidak dapat menembak hingga suhu meriam turun ke nol.

Selain itu, meriam juga sudah dalam keadaan panas di awal round dan perlu waktu untuk mendingin sebelum bisa digunakan untuk pertama kalinya.

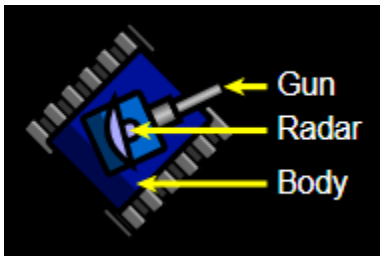
6. Tabrakan

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

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

7. Bagian Tubuh Tank

   Tubuh tank terdiri dari 3 bagian:

   Body adalah bagian utama dari tank yang digunakan untuk menggerakkan tank.

   Gun digunakan untuk menembakkan peluru dan dapat berputar bersama *body* atau independen dari *body*.

   Radar digunakan untuk memindai posisi musuh dan dapat berputar bersama *body* atau independen dari *body*.

8. Pergerakan

   Bot dapat bergerak maju dan mundur hingga kecepatan maksimum. Dibutuhkan beberapa giliran untuk mencapai kecepatan maksimum. Bot dapat mengalami percepatan maksimum sebesar 1 unit per giliran dan pengereman dengan perlambatan maksimum 2 unit per giliran. Percepatan dan perlambatan maksimum tidak bergantung pada kecepatan bot saat itu.

9. Berbelok

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

Setiap bagian tubuh memiliki kecepatan putar yang berbeda. Radar adalah bagian tercepat dan dapat berputar hingga 45 derajat per giliran, yang berarti dapat berputar 360 derajat dalam 8 giliran. Turret dan meriam dapat berputar hingga 20 derajat per giliran.

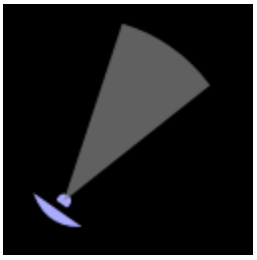
Bagian paling lambat adalah tubuh tank, yang dalam kondisi terbaik dapat berputar hingga 10 derajat per giliran. Namun, ini bergantung pada kecepatan bot saat ini. Semakin cepat bot bergerak, semakin lambat kemampuannya untuk berbelok.

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

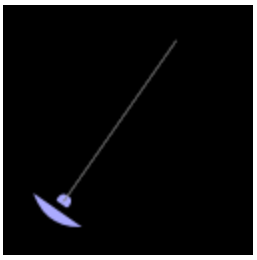
10. Pemindaian

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

    Penting untuk diperhatikan bahwa sebuah bot hanya dapat memindai bot musuh yang berada dalam jangkauan sudut pemindaian (scan arc)-nya. Sudut pemindaian ini merupakan "sapuan radar" dari arah radar sebelumnya ke arah radar saat ini dalam satu giliran.

    Jika radar tidak bergerak dalam suatu giliran, artinya radar tetap mengarah ke arah yang sama seperti pada giliran sebelumnya, maka sudut pemindaian akan menjadi nol derajat, dan bot tidak akan dapat mendeteksi musuh.



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

11. Skor

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

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

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

Untuk informasi lebih lengkap, silahkan buka dokumentasi Tank Royale pada link berikut.

BAB 2 LANDASAN TEORI

2.1. Dasar Teori

Algoritma greedy merupakan sebuah metode yang dapat digunakan untuk menyelesaikan permasalahan optimasi dengan cara membangun solusi langkah demi langkah. Pada setiap langkah, algoritma tersebut akan memilih solusi yang tampak paling menguntungkan tanpa mempertimbangkan bagaimana keputusan tersebut akan berdampak secara jangka panjang. Prinsip utama dari algoritma greedy adalah *"take what you can get now!"*. Prinsip ini selalu memilih solusi terbaik yang tersedia pada saat itu dengan harapan bahwa keputusan lokal yang optimal akan dapat menghasilkan solusi global yang optimal.

Dalam penerapannya, algoritma greedy menggunakan berbagai elemen. Pertama-tama, algoritma akan bekerja dengan menentukan himpunan kandidat (C), yaitu kumpulan elemen yang dapat dipilih sebagai bagian dari solusi untuk setiap langkah. Dari himpunan ini, algoritma akan kemudian menggunakan fungsi seleksi untuk menentukan elemen terbaik berdasarkan strategi greedy tertentu, seperti memilih elemen dengan nilai tertinggi atau dengan bobot terkecil. Setelah elemen dipilih, algoritma akan mengevaluasi apakah elemen tersebut dapat dimasukkan ke dalam solusi dengan menggunakan fungsi kelayakan. Fungsi ini berguna untuk memastikan bahwa elemen yang dipilih tidak melanggar batasan yang ada. Jika elemen tersebut layak, maka elemen tersebut ditambahkan ke dalam himpunan solusi (S), yaitu sekumpulan elemen yang sudah dipilih sebagai bagian dari hasil akhir.

Setelah setiap elemen ditambahkan ke dalam solusi, algoritma akan menggunakan fungsi solusi untuk mengevaluasi apakah himpunan solusi yang terbentuk sudah dapat memberikan sebuah solusi yang akan memberikan hasil optimal. Solusi yang diberikan akan dievaluasi sesuai dengan obyektif, baik dalam bentuk nilai maksimum maupun minimum.

Meskipun strategi greedy sering kali menghasilkan solusi yang mendekati optimal, metode ini tidak selalu menjamin bahwa solusi yang diperoleh adalah yang terbaik secara global. Hal ini dikarenakan sifat algoritma yang hanya mempertimbangkan keputusan terbaik di setiap langkah tanpa memperhitungkan efek jangka panjangnya. Akibatnya, solusi yang dihasilkan terkadang dapat bersifat sub-optimum. Suatu permasalahan harus memiliki dua sifat utama agar algoritma greedy dapat memberikan solusi optimal, yaitu *greedy-choice property* dan *optimal substructure*. *Greedy-choice property* berarti bahwa keputusan terbaik dapat diambil pada setiap langkah tanpa harus mempertimbangkan solusi sebelumnya. Sedangkan, *optimal substructure* menyatakan bahwa solusi optimal dari suatu masalah dapat diperoleh dengan menggabungkan solusi optimal dari sub-masalahnya. Jika salah satu dari dua sifat ini tidak terpenuhi, maka algoritma greedy kemungkinan besar tidak akan memberikan solusi optimal.

Algoritma greedy memiliki banyak keunggulan, namun juga memiliki kelemahan. Keunggulan utama dari algoritma greedy berada pada efisiensi waktu dan kemudahan implementasi.

Algoritma greedy juga bersifat lebih sederhana karena tidak membutuhkan struktur data tambahan yang kompleks atau penyimpanan hasil perhitungan sebelumnya. Namun, kelemahannya adalah algoritma greedy tidak selalu menjamin solusi optimal. Dalam beberapa kasus, algoritma dapat terjebak dalam *local optimum*.

Dalam penerapannya, algoritma greedy dapat digunakan untuk menyelesaikan berbagai permasalahan optimasi, berikut adalah beberapa diantaranya:

- Persoalan penukaran uang (*coin change problem)* : Dapat digunakan untuk menentukan jumlah minimum koin yang diperlukan untuk mencapai nilai tertentu.
- Persoalan memilih aktivitas (*activity selection problem*) : Dapat digunakan untuk memilih sebanyak mungkin aktivitas yang tidak saling bertumpuk berdasarkan waktu selesai paling awal.
- Minimisasi waktu dalam sistem (*minimizing time in a system*) : Dapat digunakan dalam berbagai konteks optimasi, seperti untuk penyimpanan data pada pita kaset atau penjadwalan antrian. Tujuannya adalah untuk mengurangi total waktu penyelesaian suatu proses.
- Masalah Knapsack (*Knapsack Problem*)
    a. *Integer Knapsack Problem* : Dapat digunakan untuk memilih kombinasi barang dengan nilai maksimum tanpa bisa membagi barang, sehingga setiap barang harus diambil secara utuh. Namun, seringkali solusi yang diberi oleh algoritma greedy tidak optimal.
    b. *Fractional Knapsack Problem* : Dapat digunakan untuk memungkinkan pembagian barang secara parsial untuk memenuhi kapasitas yang tersedia. Solusi yang diberi oleh algoritma greedy optimal.
- Penjadwalan job dengan tenggat waktu (*job scheduling with deadlines*) : Dapat digunakan untuk menentukan urutan pekerjaan yang harus dieksekusi agar jumlah pekerjaan yang diselesaikan sebelum tenggat waktu bisa dimaksimalkan.
- Pohon merentang minimum (*minimum spanning tree*) : Dapat digunakan untuk menemukan pohon merentang dengan bobot minimum dalam graf menggunakan algoritma Kruskal atau Prim.
- Lintasan terpendek (*shortest path problem*) : Dapat digunakan untuk menemukan jalur terpendek dari satu titik ke titik lainnya dalam graf berbobot.

- Kode huffman (*huffman coding*) : Dapat digunakan dalam kompresi data untuk mengurangi ukuran penyimpanan dengan membuat kode biner berdasarkan frekuensi kemunculan karakter dalam teks.
- Pecahan mesir (*Egyptian fraction*) : Dapat digunakan untuk mendekomposisi pecahan menjadi jumlah pecahan dengan penyebut berbeda yang lebih kecil, sesuai dengan sistem angka yang digunakan dalam matematika Mesir kuno.
- *Travelling salesperson problem (TSP)* : Dapat digunakan untuk menemukan rute perjalanan dengan jarak minimum yang mengunjungi semua kota tepat satu kali. Namun, seringkali solusi yang diberikan oleh algoritma greedy tidak optimal.
- Permainan othello : Dapat digunakan dalam pengambilan keputusan untuk memperoleh jumlah koin terbanyak di akhir permainan agar dapat memenangkan permainan.

2.2. Cara Kerja Program

Program bot pada Robocode Tank Royale bekerja berdasarkan instruksi masing-masing bot yang telah ditulis dalam bahasa pemrograman C#. Setiap bot akan beroperasi dalam pertempuran sesuai dengan aturan yang telah ditentukan. Bot sendiri tidak dapat dikendalikan secara langsung oleh pemain selama pertempuran berlangsung, sehingga semua tindakan bot sepenuhnya bergantung pada algoritma strategi greedy yang telah diimplementasikan dalam kode program. Berikut merupakan penjelasan mengenai bagaimana bot melakukan aksinya, penerapan algoritma greedy, serta cara menjalankan bot di dalam lingkungan permainan.

A. Struktur Direktori Program

Program bot yang dibuat memiliki struktur direktori sebagai berikut:

```
Tubes1_Bakwan-Jagung (Root Directory)
├── doc
│   └── Bakwan Jagung.pdf
├── src
│   ├── alternative-bots
│   │   ├── Amethyst
│   │   │   ├── Amethyst.cmd
│   │   │   ├── Amethyst.cs
│   │   │   ├── Amethyst.csproj
│   │   │   ├── Amethyst.json
│   │   │   └── Amethyst.sh
```

```
│   │   ├── Handoko
│   │   │   ├── Handoko.cmd
│   │   │   ├── Handoko.cs
│   │   │   ├── Handoko.csproj
│   │   │   ├── Handoko.json
│   │   │   └── Handoko.sh
│   │   └── kakureMomojiri
│   │       ├── kakureMomojiri.cmd
│   │       ├── kakureMomojiri.cs
│   │       ├── kakureMomojiri.csproj
│   │       ├── kakureMomojiri.json
│   │       └── kakureMomojiri.sh
│   └── main-bot
│       └── Ako
│           ├── Ako.cmd
│           ├── Ako.cs
│           ├── Ako.csproj
│           ├── Ako.json
│           └── Ako.sh
└── README.md
```

B. Cara Bot Melakukan Aksinya

Setiap bot dalam permainan Robocode Tank Royale beroperasi dengan siklus *turn-based.* Pada setiap giliran *(turn)* bot akan mendapatkan kesempatan untuk melakukan aksi berdasarkan instruksi yang sebelumnya telah diprogramkan pada kode bot, sehingga pemain tidak dapat mengontrol bot secara langsung di saat pertempuran.

Setiap bot akan menerima input dari *game engine* pada saat pertempuran melalui API resmi permainan. API ini akan memberikan berbagai informasi penting, seperti posisi bot sendiri, status energi, deteksi musuh, serta kejadian di arena seperti **ta**brakan dengan dinding atau bot lain, dan tembakan peluru musuh.. Berdasarkan input ini, bot akan kemudian mengambil keputusan terbaik berdasarkan kode pada program mereka untuk memberikan output berupa perintah aksi kepada server game, seperti perintah bergerak, perintah memindai musuh dengan radar, dan perintah menembak.

Bot memiliki waktu yang terbatas dalam melakukan giliran, yaitu sekitar 30-50 ms. Namun, batas waktu ini dapat diatur sebelum awal pertempuran. Jika melewati batas waktu tersebut, bot akan kehilangan giliran dan tidak dapat melakukan aksi apa pun pada *turn* tersebut.

Berikut adalah penjelasan mengenai aksi-aksi utama yang dapat dilakukan bot dalam pertarungan:

1. Gerakan (*Movement*)
   Bot dapat melakukan aksi berupa bergerak maju, mundur, dan berputar . Hal ini dapat dilakukan bot untuk mendekati target atau menghindari ancaman. Untuk menghindari tabrakan dengan dinding arena atau bot musuh yang dapat mengurangi energi, bot dapat berbelok atau berhenti.

2. Menembak (*Shooting*)
   Untuk aksi menembak, bot dapat memutuskan kapan dan dengan kekuatan berapa ia ingin menembak. Sebelum menembak, bot akan biasanya terlebih dahulu mempertimbangkan jarak dan energi dari musuh. berdasarkan jarak dan energi musuh. Biasanya, bot akan menyerang musuh dengan energi terendah atau jarak terdekat untuk memaksimalkan peluang mengalahkan musuh tersebut. Selain itu, dalam aksi menembak, bot juga dapat menyesuaikan sudut tembakan untuk memaksimalkan kemungkinan tembakan mengenai target.

3. Pemindaian (*Scanning)*
   Selain itu, bot juga dapat melakukan aksi pemindaian dengan menggunakan radar untuk mendeteksi musuh dalam jangkauan hingga 1200 piksel. Pada aksi ini, jika tidak ada musuh yang terdeteksi, radar akan berputar terus dengan kecepatan maksimum untuk memperluas cakupan penglihatan. Ketika radar berhasil mendeteksi musuh, radar akan secara otomatis mengunci target tersebut untuk terus memantau pergerakannya.

4. Bertahan Hidup *(Survival)*
   Jika energi bot rendah, bot akan cenderung memilih untuk beralih strategi dan melakukan aksi memprioritaskan bertahan hidup. Saat melakukan aksi ini, bot cenderung akan bergerak menjauh atau bersembunyi di sudut arena untuk meminimalkan risiko diserang musuh.

C. Cara Mengimplementasikan Algoritma Greedy Ke Dalam Bot

Implementasi algoritma greedy pada bot dalam Robocode Tank Royale bertujuan untuk membuat bot mengambil keputusan terbaik secara lokal pada setiap turn permainan. Untuk mengimplementasikan algoritma greedy yang telah dibuat ke dalam bot, perlu dibuat terlebih dahulu sebuah folder baru pada direktori src/main-bot/ atau src/alternative-bots/ , misalnya Ako. Lalu, buatlah file baru dengan nama yang sesuai

dengan bot, misalnya Ako.cs, di dalam folder tersebut. File ini akan berisi logika utama bot serta penerapan algoritma greedy.

Selanjutnya, buat sebuah kelas baru dengan nama bot yang akan menurunkan kelas Bot dari API. Dalam kelas ini, konstruktor dan metode-metode penting yang diperlukan, seperti `Run()`, `OnScannedBot()`, `OnHitWall()`, dan `OnBotDeath()` harus diimplementasikan, dengan menerapkan algoritma greedy di setiap pengambilan keputusan bot. Berikut adalah contoh implementasi kelas bot:

Program bot yang dibuat memiliki struktur direktori sebagai berikut:

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

public class Ako : Bot
{

    public Ako() : base(BotInfo.FromFile("Ako.json")) { }

    public override void Run()
    {
        BodyColor = Color.FromArgb(167, 199, 231);
        TurretColor = Color.White;
        RadarColor = Color.FromArgb(45, 62, 80);
        BulletColor = Color.FromArgb(255, 215, 0);
        ScanColor = Color.FromArgb(0, 255, 255);

        while (IsRunning)
        {
            SetForward(double.PositiveInfinity);
            SetTurnLeft(double.PositiveInfinity);
            Rescan();
        }
    }

    public override void OnScannedBot(ScannedBotEvent e)
    {
        FollowTarget(e);
    }
```

```
    public override void OnHitWall(HitWallEvent e)
    {
        SetBack(5000);
        SetTurnLeft(90);
        Rescan();
    }

    public override void OnBotDeath(BotDeathEvent e)
    {
        Rescan();
    }

    private void FollowTarget(ScannedBotEvent e)
    {
        Target(e.X, e.Y);
        SetForward(double.PositiveInfinity);
        Rescan();
    }

    public override void OnHitBot(HitBotEvent e)
    {
        Target(e.X, e.Y);
        Fire(3);
        Fire(3);
        Fire(3);
        Rescan();
    }

    private void Target(double x, double y)
    {
        var bearing = BearingTo(x, y);
        SetTurnLeft(bearing);
    }

    static void Main() => new Ako().Start();
}
```

Pada contoh di atas, algoritma greedy diterapkan dalam pengambilan keputusan bot secara lokal dengan keterangan sebagai berikut:

- `Run()`:
  Bot bergerak maju tanpa henti dan berputar secara konstan untuk memindai

musuh dengan cepat. Strategi ini memastikan bot tidak statis dan selalu siap mengambil keputusan terbaik berdasarkan situasi terbaru.

- `OnScannedBot()`:
  Saat musuh terdeteksi oleh radar bot, bot akan segera mengikuti dan menyerang target menggunakan metode `FollowTarget()`. Dalam pendekatan greedy, bot tidak akan menunda untuk menyerang dan akan memanfaatkan momen ketika musuh terlihat. Jika musuh berada dalam jarak dekat, bot bergerak mendekat untuk memaksimalkan *damage* dari tembakan.

- `OnHitWall()`:
  Ketika bot menabrak dinding arena, bot langsung mundur dengan cepat dan berputar 90 derajat untuk keluar dari situasi sulit. Strategi greedy dalam situasi ini meminimalkan kerugian akibat tabrakan dengan segera bergerak menjauh.

- `OnHitBot()`:
  Saat bertabrakan dengan musuh, bot akan langsung menembak dengan kekuatan maksimum (3) sebanyak tiga kali secara cepat. Strategi greedy ini dirancang untuk mendapatkan keuntungan maksimum dalam waktu singkat, terutama ketika musuh dalam kondisi rentan akibat tabrakan.

- `OnBotDeath()`:
  Ketika ada bot musuh yang mati, bot segera memindai ulang untuk mendapatkan target baru. Pendekatan greedy ini memastikan bot selalu memiliki target, memaksimalkan potensi skor, dan meminimalkan waktu bot dalam keadaan pasif tanpa target.

D. Cara Menjalankan Bot

Program bot yang telah dibuat dapat dijalankan dengan menggunakan aplikasi terminal atau pada aplikasi vscode. Sebelum menjalankan bot, pastikan Starter Pack dan API sudah terlebih dahulu di download, serta folder struktur direktori sudah sesuai. Berikut adalah langkah-langkah cara untuk menjalankan bot dan build game engine:

1. Masuk ke Directory "tank-royale-0.30.0"

```
cd tank-royale-0.30.0
```

2. Lakukan Clean & Build GUI App dengan Gradlew

```
./gradlew :gui-app:clean
./gradlew :gui-app:build
```

3. Kemudian, jalankan executable jar

```
java -jar ./gui-app/build/libs/robocode-tankroyale-gui-0.30.0.jar
```

Setelah itu, iu untuk menjalankan battle pada aplikasi robocode tank royale, dapat mengikuti langkah-langkah sesuai dibawah ini:

1. Pada game engine, klik **Config → Bot Root Directories** dan tambahkan folder bot kamu.
2. Setelah itu, klik **Battle → Start Battle**.
3. Kemudian, pilihlah bot di kotak kiri-atas dan klik **Boot →**.
4. Lalu, tambahkan bot ke dalam permainan dengan **Add →** atau **Add All →**.
5. Terakhir, tekan **Start Battle** untuk memulai permainan.

BAB 3 APLIKASI STRATEGI GREEDY

3.1. Proses Mapping Persoalan Robocode dengan Algoritma Greedy

Algoritma greedy merupakan pendekatan dalam pemecahan masalah yang berfokus pada pengambilan keputusan lokal terbaik pada setiap langkah dengan harapan bahwa keputusan ini akan menghasilkan solusi global yang optimal. Prinsip utama dari algoritma greedy adalah selalu memilih opsi yang tampak paling menguntungkan pada saat itu tanpa mempertimbangkan konsekuensi jangka panjang.

Dalam konteks Robocode, strategi greedy dapat dimodelkan dalam elemen-elemen berikut:

- **Himpunan Kandidat**: Himpunan keputusan yang mungkin diambil oleh bot, seperti bergerak, menembak, atau menghindari musuh.
- **Himpunan Solusi**: Rangkaian keputusan yang dipilih oleh bot untuk mencapai tujuan tertentu, seperti bertahan lebih lama atau mengalahkan lawan.
- **Fungsi Solusi**: Mengukur efektivitas strategi yang diterapkan bot, misalnya jumlah serangan yang berhasil atau jumlah serangan lawan yang dihindari.
- **Fungsi Seleksi**: Memilih keputusan terbaik berdasarkan kriteria tertentu, seperti menargetkan musuh terdekat atau berpindah ke posisi yang lebih aman.
- **Fungsi Kelayakan**: Memeriksa apakah keputusan yang diambil masih valid, misalnya tidak melanggar batas arena atau tidak dalam posisi rentan terhadap serangan lawan.
- **Fungsi Objektif**: Mengoptimalkan strategi bot, baik dalam hal menyerang secara efektif maupun bertahan lebih lama di arena dengan memanfaatkan setiap keputusan yang diambil secara efisien.

3.2. Algoritma Greedy pada Robocode Pertama

Bot Ako mengimplementasikan strategi greedy berbasis pergerakan agresif dan eksekusi tembakan langsung terhadap target yang paling mudah diserang. Fokus utama dari bot ini adalah terus bergerak di arena, mendeteksi musuh, dan langsung menembak begitu terjadi interaksi dengan lawan.

Untuk menjalankan strategi ini, bot akan memilih target yang paling mudah ditembak dengan pendekatan berikut:

- **Target terbaru**: Bot selalu menyesuaikan arah menuju musuh yang terdeteksi paling baru

- **Interaksi langsung**: Saat terjadi tabrakan dengan musuh, bot langsung menembak dengan kekuatan maksimal.

Berikut adalah tahapan algoritma pada bot pertama:

1. Pergerakan Dasar:
    a. Bot akan terus bergerak maju dalam pola memutar ke kiri secara tak terbatas.
    b. Jika bot menabrak dinding, bot akan segera bergerak mundur sejauh 5000 unit dan berbelok ke kiri sebesar 90° sebelum melanjutkan kembali pola pergerakan sebelumnya.
    c. Pergerakan ini bertujuan untuk membuat bot selalu aktif dan sulit menjadi target tetap bagi musuh.
2. Pemindaian dan Pemilihan Target:
    a. Bot akan melakukan pemindaian secara kontinu dan selalu mengarah ke target yang terdeteksi paling baru.
    b. Jika bot mendeteksi musuh dalam arena, bot akan menyesuaikan arah menuju musuh dengan memutar badan ke arah target.
    c. Bot tidak menyimpan informasi HP lawan, tetapi secara greedy selalu memilih target terbaru karena lebih mudah ditembak.
3. Eksekusi Serangan:
    a. Saat terjadi tabrakan dengan bot lain, bot langsung menembak tiga kali berturut-turut dengan kekuatan tembakan 3.
    b. Bot tidak melakukan perhitungan strategi berbasis kondisi musuh, tetapi mengeksekusi tembakan secara agresif begitu kontak terjadi.
4. Penanganan kolisi:
    a. Jika bot menabrak dinding, bot akan bergerak mundur sejauh 5000 unit dan berbelok 90° ke kiri.
    b. Jika bot bertabrakan dengan bot lain, bot akan langsung menembak tiga kali dan kemudian melanjutkan pemindaian ulang.

3.3. Algoritma Greedy pada Robocode Kedua

Bot Handoko mengimplementasikan strategi greedy berbasis pemilihan target dengan jarak terdekat dan eksekusi tembakan agresif saat terjadi interaksi langsung dengan musuh. Fokus utama dari bot ini adalah terus bergerak, mendeteksi musuh, dan menyerang target yang memiliki jarak paling dekat.

Untuk menjalankan strategi ini, bot akan memilih target dengan pendekatan berikut:

- **Target terdekat**: Bot selalu memilih target dengan jarak paling dekat di antara semua musuh yang terdeteksi.

- **Interaksi langsung**: Saat terjadi tabrakan dengan musuh, bot langsung menembak dengan kekuatan maksimal.

Berikut adalah tahapan algoritma pada bot kedua:

1. Pergerakan Dasar:
    a. Bot akan terus bergerak maju dalam pola memutar ke kiri secara tak terbatas.
    b. Jika bot menabrak dinding, bot akan segera bergerak mundur sejauh 5000 unit dan berbelok ke kiri sebesar 90° sebelum melanjutkan kembali pola pergerakan sebelumnya.
    c. Pergerakan ini bertujuan untuk menjaga bot tetap aktif dan menghindari menjadi sasaran tetap bagi musuh.
2. Pemindaian dan Pemilihan Target:
    a. Bot akan melakukan pemindaian secara kontinu dan selalu memilih target dengan jarak terdekat.
    b. Jika musuh yang baru terdeteksi lebih dekat daripada target sebelumnya, bot akan memperbarui targetnya.
    c. Jika hanya tersisa satu musuh di arena, bot langsung fokus ke musuh tersebut tanpa membandingkan jarak.
3. Eksekusi Serangan:
    a. Saat terjadi tabrakan dengan bot lain, bot langsung menembak tiga kali berturut-turut dengan kekuatan tembakan 3.
    b. Bot menyesuaikan arah tembakan dengan memutar badan ke arah target dengan dua kali sudut perhitungan jarak.
4. Penanganan kolisi:
    a. Jika bot menabrak dinding, bot akan bergerak mundur sejauh 5000 unit dan berbelok 90° ke kiri.
    b. Jika bot bertabrakan dengan bot lain, bot akan langsung menembak tiga kali dan kemudian melanjutkan pemindaian ulang.

3.4. Algoritma Greedy pada Robocode Ketiga

Bot Amethyst mengimplementasikan strategi greedy dengan sifat *"Opportunistic Survivalist"*. Strategi greedy bot ini mengutamakan penghindaran di awal permainan hingga ia berhasil menemukan sebuah peluang. Ketika, Amethyst menemukan peluang baik untuk memenangkan pertempuran dengan bot lawan, iia akan langsung beralih ke serangan agresif berupa *ramming*. Fokus utama dari bot ini adalah untuk terus bertahan hidup selama mungkin sambil memanfaatkan setiap kesempatan sebaik mungkin untuk memperoleh banyak skor dan mengalahkan bot lawan.

Untuk menjalankan strategi ini,bot akan memilih target yang paling lemah dan paling mudah dikalahkan dengan pendekatan berikut:

- **Peluang Serangan:** Bot akan secara aktif berusaha untuk mencari target dengan energi rendah (≤ 40) atau jika hanya ada satu musuh tersisa, bot akan menunggu hingga terdapat selisih energi sebesar 20 poin lebih tinggi dari musuh sebelum memulai serangan secara agresif.
- **Serangan Opportunistik:** Jika peluang besar untuk menyerang muncul, bot akan segera beralih ke mode *ramming* dan menabrak musuh secara agresif dan terus-menerus.

Berikut adalah tahapan algoritma pada bot ketiga:

1. Pergerakan Dasar:
   Jika Amethyst sedang tidak berada dalam mode ramming atau waktu ramming sudah melebihi batas maksimal, bot akan fokus untuk melakukan pola pergerakan dasar untuk menghindar yang terdiri dari:

   a. Penghindaran Acak: Saat tidak dalam mode ramming, bot akan menggunakan fungsi `PerformAdvancedDodging()` yang memilih secara acak satu dari lima pola pergerakan.
      - Kasus 0: Maju sejauh 100 unit, lalu belok kanan 45°.
      - Kasus 1: Mundur sejauh 90 unit, lalu belok kiri 50°.
      - Kasus 2: Maju sejauh 70 unit, belok kanan 30°, lalu mundur sejauh 50 unit.
      - Kasus 3: Mundur sejauh 110 unit, lalu belok kiri 40°.
      - Kasus 4: Maju sejauh 80 unit, belok kanan 20°, lalu mundur sejauh 60 unit.

   b. Jika bot menabrak dinding, bot akan langsung mundur sejauh 50 unit untuk menghindari posisi terjepit, kemudian berbelok secara acak sebesar $135° \pm 30°$ untuk mengubah arah dengan cepat, dan maju sejauh 150 ± 30 unit agar segera kembali ke pergerakan normal.

   c. Pergerakan ini bertujuan untuk membuat bot dapat selalu dengan efektif menghindari serangan dari musuh

2. Pemindaian dan Pemilihan Target:
   a. Radar bot diputar sebesar 60° secara konstan untuk mencari musuh. Saat mendeteksi musuh, bot akan mengunci radar pada target untuk memastikan posisi tetap terpantau.
   b. Jika musuh memiliki energi ≤ 40 atau jika hanya ada satu musuh tersisa dengan energi yang lebih rendah ≥ 20 poin dibanding bot, maka bot beralih ke mode *ramming* dan menyerang musuh.

3. Eksekusi Serangan:

Bot akan melakukan serangan agresif dengan teknik *ramming* yang menabrak langsung musuh untuk menghabiskan energinya. Implementasi serangan ini terdiri dari tiga tahap utama:

a. Pertama-tama, bot akan mengunci target dengan memutar radar untuk memastikan posisi target terpantau dengan menggunakan fungsi `LockRadarOnTarget(e)`. Tujuan dari penguncian ini adalah agar bot tidak kehilangan target saat mendekat untuk melakukan ramming.
b. Setelah target terkunci, bot akan bergerak maju secara agresif dengan perintah `Forward(9999)` untuk mendekati target secepat mungkin. Saat mendekati target, bot akan terus memantau energi musuh.. Jika energi musuh turun hingga 0, bot akan segera menghentikan mode ramming dan kembali ke pola penghindaran dengan fungsi `EnergyPerformAdvancedDodging()`
c. Setelah keluar dari mode ramming, bot akan kembali menggunakan pola penghindaran dengan fungsi `isRammingPerformAdvancedDodging()` hingga memperoleh kembali peluang untuk menyerang.

4. Penanganan kolisi:
    a. Ketika bot menabrak dinding arena, bot akan mundur sejauh 50 unit untuk menjaga jarak aman dari batas arena. Selanjutnya, bot berbelok kanan sebesar 135° ± 30° secara acak, kemudian maju sejauh 150 ± 30 unit untuk menghindari kemungkinan terperangkap di pojok arena.
    b. Jika terjadi tabrakan dengan musuh saat berada dalam mode ramming dan musuh masih memiliki energi lebih besar dari 0, bot akan terus maju dengan perintah `Forward(9999)` untuk memaksimalkan dampak serangan. Namun, jika tabrakan terjadi di luar mode ramming, bot akan mundur sejauh 100 ± 20 unit secara acak untuk menjaga jarak aman dari musuh dan kembali mencari target potensial.

3.5. Algoritma Greedy pada Robocode Keempat

Bot Kakure Momojiri mengimplementasi strategi greedy dengan fokus bergerak secara zig-zag untuk menghindari serangan musuh sambil menyerang ketika ada kesempatan. Bot ini menyerang musuh dengan menembak sesuai dengan energi Bot saat itu. Bot ini juga menghindari tabrakan dengan dinding. Fokus utama dari Bot ini adalah bertahan hidup lebih lama juga memanfaatkan energi untuk serangan yang presisi.

Strategi yang diterapkan oleh bot Kakure Momojiri berfokus pada:

- **Pergerakan Zig-zag**: Bot bergerak dalam pola zig-zag lurus untuk menghindari kemungkinan terkena tembakan dan tabrakan dengan dinding:
- **Serangan Berdasarkan Energi**: Bot menembak ketika ada kesempatan dan menembak sesuai dengan energi bot saat itu agar tembakan optimal.

Berikut adalah tahapan algoritma untuk Bot keempat:

1. Pergerakan Dasar
    a. Gerakan Zig-zag: Bot bergerak maju sejauh 100 unit dengan sudut 30 derajat kekiri, kemudian bergerak maju sejauh 100 unit dengan sudut 90 derajat ke kanan, pola ini akan selalu berulang.
    b. Posisi Bot: Apabila posisi bot kurang dari 60 unit dari batas arena, bot akan mengidentifikasi bahwa ia terlalu dekat dengan dinding. Untuk menghindari tertabrak dengan dinding, bot akan segera berbelok 90 derajat dan bergerak mundur sejauh 150 untuk untuk kembali ke posisi yang aman sebelum melanjutkan pergerakan zig-zag
2. Pemindaian dan Pemilihan Target
    a. Radar berputar 360 derajat ke kanan untuk mencari musuh di sekitar bot
    b. Jika ada musuh yang terdeteksi, arah meriam disesuaikan ke posisi musuh agar sudut tembak akurat
3. Eksesuki Serangan
    a. Bot akan menembak jika sudut antara meriam dan musk kurang dari 2.5 derajat. Peluru juga ditembakkan dengan daya maksimal yang disesuaikan dengan energi tersisa untuk efisiensi
4. Penanganan Kolisi
    a. Jika Tertabrak bot, Bot berbelok 90 derajat ke kiri dan maju 200 unit untuk menghindari tabrakan lanjutan dengan bot dan bergerak cepat untuk kabur dari musuh.
    b. Jika Tertabrak dinding, Bot berbelok 90 derajat ke kiri dan maju 150 unit untuk menghindari tabrakan lanjutan dan bergerak sedikit agar mengurangi dampak tabrakan yang lebih besar, apabila tertabrak lagi saat bergerak.
    c. Jika tertabrak peluru, Bot berbelok 30° ke kanan dan maju 200 unit, bot hanya berbelok 30 derajat untuk mempercepat gerakan dan bergerak cepat untuk menghindari tembakan peluru.

3.6. Analisis Efisiensi dan Efektivitas dari Berbagai Robocode

Pada bagian ini, dilakukan analisis perbandingan terhadap empat strategi greedy yang diterapkan oleh masing-masing bot: Ako, Handoko, Amethyst, dan Kakure Momojiri. Meskipun

keempatnya menggunakan prinsip greedy, pendekatan yang digunakan memiliki variasi dalam pemilihan target, pergerakan, dan strategi serangan.

Aspek	Ako	Handoko	Amethyst	Kakure Momojiri
Pemilihan Target	Musuh terbaru	Musuh terdekat	Musuh dengan energi rendah	Tidak spesifik, fokus menghindar
Pergerakan	Memutar kiri terus	Memutar kiri dengan responsif	Acak untuk menghindari tembakan	Zig-zag untuk mobilitas tinggi
Eksekusi Serangan	Menyerang setiap interaksi	Sudut serangan lebih cermat	Ramming jika peluang menang besar	Menyesuaikan daya tembak berdasarkan energi
Penanganan Kolisi	Mundur 5000 unit dan berbelok	Mirip Ako	Mundur 50 unit dan ubah arah acak	Belok 90° dan mundur

Tabel 3.1. Analisis Efisiensi dan Efektivitas Berbagai Robocode

Efektivitas masing-masing bot bergantung pada kondisi pertempuran. Ako dan Handoko unggul dalam pertempuran cepat karena agresivitasnya, tetapi rentan terhadap serangan balik akibat kurangnya pertahanan. Amethyst lebih efisien dalam bertahan dan memilih waktu yang tepat untuk menyerang, meskipun strategi rammingnya berisiko jika salah eksekusi. Kakure Momojiri memiliki mobilitas tinggi dan efisiensi serangan yang lebih baik, tetapi kurang efektif dalam duel langsung karena tidak memiliki fokus serangan yang spesifik.

Secara keseluruhan, tidak ada strategi yang secara absolut lebih unggul, tetapi setiap pendekatan memiliki kelebihan dan kekurangan tergantung pada situasi di arena.

BAB 4 IMPLEMENTASI DAN PENGUJIAN

4.1. Robocode 1

a. Pseudocode

```
// Inisialisasi bot
BEGIN Ako

    SET bodyColor = (167, 199, 231)
    SET turretColor = (255, 255, 255)
    SET radarColor = (45, 62, 80)
    SET bulletColor = (255, 215, 0)
    SET scanColor = (0, 255, 255)

    START BOT

    WHILE IsRunning DO
        SetForward(INFINITY)  // Maju terus
        SetTurnLeft(INFINITY)  // Putar terus berlawanan arah jarum
jam
        Rescan()  // Lakukan scanning ulang
    END WHILE

END Ako

// Event ketika mendeteksi bot musuh
ON EVENT OnScannedBot(evt)
    FollowTarget(evt)
END EVENT

// Event ketika menabrak dinding (handle stuck)
ON EVENT OnHitWall(evt)
    SetBack(5000)  // Mundur jauh dari dinding
    SetTurnLeft(90)  // Putar kiri 90 derajat
    Rescan()  // Lakukan scanning ulang
END EVENT

// Event ketika bot musuh mati
ON EVENT OnBotDeath(evt)
    Rescan()  // Lakukan scanning ulang
END EVENT

// Event ketika bertabrakan dengan bot lain
ON EVENT OnHitBot(evt)
    Target(evt.X, evt.Y)  // Arahkan bot ke lawan
    Fire(3)
```

```
    Fire(3)
    Fire(3)  // Tembak tiga kali dengan kekuatan penuh
    Rescan()  // Lakukan scanning ulang
END EVENT

// Fungsi untuk mengikuti target
FUNCTION FollowTarget(evt)
    Target(evt.X, evt.Y)  // Arahkan bot ke target
    SetForward(INFINITY)  // Kejar terus
    Rescan()  // Lakukan scanning ulang
END FUNCTION

// Fungsi untuk mengarahkan bot ke target
FUNCTION Target(x, y)
    SET bearing = BearingTo(x, y)
    SetTurnLeft(bearing)  // Putar bot sesuai arah target
END FUNCTION

END Ako
```

b. Struktur Data

Bot ini menggunakan struktur data primitif dari bahasa pemrograman untuk menyimpan informasi pertempuran. Berikut beberapa variabel utama yang digunakan:

- bearing (double) → Menyimpan arah relatif ke target untuk navigasi.

c. Fungsi dan Prosedur yang Digunakan dalam Solusi Greedy

Bot ini menggunakan algoritma greedy yang menargetkan target terbaru setiap `Rescan()` dipanggil tanpa mempertimbangkan resiko jangka panjang. Fungsi dan prosedur yang digunakan dikategorikan berdasarkan proses utama sebagai berikut:

1. Pemindaian & Pemilihan Target *(Greedy Selection)*
   Bot selalu mencari target terbaru untuk dikejar untuk setiap iterasi.
   a. `Rescan ()` → melakukan pemindaian ulang untuk mencari target baru.
   b. `BearingTo(x, y)` → menghitung derajat arah dari bot ke target.
2. Menyerang Target *(Greedy Attack)*
   Begitu musuh terdeteksi, bot langsung menyerang tanpa analisis lebih lanjut.
   a. `Fire(3)` → menembak dengan kekuatan 3 tanpa menganalisis lebih lanjut.

3. Pergerakan Bot (Greedy Movement)
   Bot selalu bergerak maju dan terus berputar untuk mencari target.

   a. `SetForward(distance)`, `SetBack(distance)` → maju / mundur sebanyak `distance` unit untuk bergerak mendekati / menjauhi / mencari target.
   b. `SetTurnLeft(angle)` → berputar sebesar `angle` unit untuk mengarahkan diri ke target.
   c. `FollowTarget(evt)` → mengejar target untuk dihancurkan.
   d. `Target(evt.X, evt.Y)` → mengarahkan bot ke target.

4. Penanganan Kolisi & Target yang Hilang (Greedy Handler)
   Jika target mati, bot segera mencari target baru. Sedangkan jika bot bertabrakan dengan tembok, bot akan berotasi dan menjauh dari tembok.

   a. `ON EVENT OnBotDeath(evt)` → bot akan melakukan `Rescan()` untuk memindai target baru.
   b. `ON EVENT OnHitWall(evt)` → bot akan mundur sebanyak 5000 unit dan berotasi ke arah kiri sebanyak 90 derajat.
   c. `ON EVENT OnHitBot(evt)` → bot akan menembak dengan kekuatan 3 tanpa menganalisis lebih lanjut dengan harapan mengenai target dari jarak yang sangat dekat.

4.2. Robocode 2

a. Pseudocode

```
// Inisialisasi bot
BEGIN Handoko

    SET bodyColor = (255, 160, 180)
    SET turretColor = (255, 255, 255)
    SET radarColor = (250, 140, 180)
    SET bulletColor = (138, 43, 226)
    SET scanColor = (0, 128, 0)

    SET target = NULL  // Target awal tidak ada
    SET enemy = EnemyCount  // Simpan jumlah musuh saat mulai

    START BOT

    // Loop utama bot
    WHILE IsRunning DO
        SetForward(INFINITY)  // Maju terus tanpa batas
        SetTurnLeft(INFINITY)  // Berputar terus berlawanan arah jarum
jam
        Rescan()  // Lakukan scanning ulang untuk menemukan musuh
    END WHILE

END Handoko

// Event ketika mendeteksi bot musuh
ON EVENT OnScannedBot(evt)
    // Pilih target jika belum ada atau musuh lebih dekat dari target
sebelumnya
    IF target == NULL OR DistanceTo(evt.X, evt.Y) <
DistanceTo(target.X, target.Y) OR enemy == 1 THEN
        SET target = evt  // Simpan musuh sebagai target baru
    END IF
    FollowTarget(target)  // Kejar target
END EVENT

// Event ketika menabrak dinding (handle stuck)
ON EVENT OnHitWall(evt)
    SetBack(5000)  // Mundur jauh untuk keluar dari dinding
    SetTurnLeft(90)  // Putar kiri 90 derajat agar tidak terjebak
    Rescan()  // Lakukan scanning ulang
END EVENT

// Event ketika bot musuh mati
ON EVENT OnBotDeath(evt)
```

```
    enemy = enemy - 1  // Kurangi jumlah musuh yang tersisa
    Rescan()  // Cari target baru
END EVENT

// Event ketika bertabrakan dengan bot lain
ON EVENT OnHitBot(evt)
    Target(evt.X, evt.Y)  // Arahkan bot ke musuh
    Fire(3)  // Tembak tiga kali dengan kekuatan penuh
    Fire(3)
    Fire(3)
    Rescan()  // Lakukan scanning ulang
END EVENT

// Fungsi untuk mengikuti target
FUNCTION FollowTarget(evt)
    Target(evt.X, evt.Y)  // Arahkan bot ke target
    SetForward(INFINITY)  // Kejar target tanpa batas
    Rescan()  // Lakukan scanning ulang
END FUNCTION

// Fungsi untuk mengarahkan bot ke target
FUNCTION Target(x, y)
    SET bearing = BearingTo(x, y)
    SetTurnLeft(2 * bearing)  // Putar bot dua kali sudut relatif ke
target agar lebih cepat mengunci
END FUNCTION

END Handoko
```

b. Struktur Data

Bot ini menggunakan struktur data primitif dari bahasa pemrograman untuk menyimpan informasi pertempuran. Berikut beberapa variabel utama yang digunakan:

- target (ScannedBotEvent) → Menyimpan informasi bot musuh terbaru yang terdeteksi.
- enemy (integer) → Menyimpan jumlah total musuh yang tersisa dalam arena.
- bearing (double) → Menyimpan arah relatif ke target untuk navigasi.

c. Fungsi dan Prosedur yang Digunakan dalam Solusi Greedy

Bot ini menggunakan algoritma greedy yang menargetkan target dengan jarak terdekat tanpa mempertimbangkan resiko jangka panjang. Fungsi dan prosedur yang digunakan dikategorikan berdasarkan proses utama sebagai berikut:

1. Pemindaian & Pemilihan Target *(Greedy Selection)*
   Bot selalu mencari target terbaru untuk dikejar untuk setiap iterasi.

   a. `Rescan ()` → melakukan pemindaian ulang untuk mencari target baru.
   b. `BearingTo(x, y)` → menghitung derajat arah dari bot ke target.
   c. IF target == NULL OR DistanceTo(evt.X, evt.Y) < DistanceTo(target.X, target.Y) OR enemy == 1 THEN SET target = evt → menargetkan musuh dengan jarak terdekat.

2. Menyerang Target (*Greedy Attack)*
   Begitu musuh terdeteksi, bot langsung menyerang tanpa analisis lebih lanjut.

   a. `Fire(3)` → menembak dengan kekuatan 3 tanpa menganalisis lebih lanjut.

3. Pergerakan Bot *(Greedy Movement)*
   Bot selalu bergerak maju dan terus berputar untuk mencari target.

   a. `SetForward(distance)`, `SetBack(distance)` → maju / mundur sebanyak `distance` unit untuk bergerak mendekati / menjauhi / mencari target.
   b. `SetTurnLeft(angle)` → berputar sebesar `angle` unit untuk mengarahkan diri ke target.
   c. `FollowTarget(evt)` → mengejar target untuk dihancurkan.
   d. `Target(evt.X, evt.Y)` → mengarahkan bot ke target.

4. Penanganan Kolisi & Target yang Hilang *(Greedy Handler)*
   Jika target mati, bot segera mencari target baru. Sedangkan jika bot bertabrakan dengan tembok, bot akan berotasi dan menjauh dari tembok.

   a. `ON EVENT OnBotDeath(evt)` → bot akan melakukan `Rescan()` untuk memindai target baru.
   b. `ON EVENT OnHitWall(evt)` → bot akan mundur sebanyak 5000 unit dan berotasi ke arah kiri sebanyak 90 derajat.
   c. `ON EVENT OnHitBot(evt)` → bot akan menembak dengan kekuatan 3 tanpa menganalisis lebih lanjut dengan harapan mengenai target dari jarak yang sangat dekat.

4.3. Robocode 3

a. Pseudocode

```
// Inisialisasi bot
BEGIN Amethyst

    SET BodyColor = (138, 43, 226)
    SET TurretColor = (230, 230, 250)
    SET RadarColor = (148, 0, 211)
    SET ScanColor = (255, 182, 193)
    SET TracksColor = (75, 0, 130)
    SET GunColor = (102, 51, 153)

    SET isRamming = FALSE                // Status mode ramming
    SET rammingTimeout = 0               // Waktu mode ramming
    SET maxRammingTimeout = 120          // Batas waktu mode ramming
    INISIALISASI list Enemies sebagai set kosong

    START BOT

    // Loop utama bot
    WHILE IsRunning DO
        IF NOT isRamming OR rammingTimeout > maxRammingTimeout THEN
            SET isRamming = FALSE
            SET rammingTimeout = 0
            Panggil PerformAdvancedDodging() // Menghindar secara acak
        END IF

        IF isRamming THEN
            Tambahkan rammingTimeout sebesar 1
        ELSE
            Putar Radar sebesar 60 derajat ke kanan untuk memindai
musuh
        END IF
    END WHILE
END Amethyst

// Event ketika mendeteksi bot musuh
ON ScannedBot(e)
    Tambahkan ID musuh e.ScannedBotId ke dalam Enemies
    HITUNG jarakMusuh = DistanceTo(e.X, e.Y)
    SET isSingleEnemy = (Jumlah Enemies == 1)  // Cek apakah hanya ada
satu musuh

    IF (e.Energy <= 40 OR (isSingleEnemy AND (Energi kita - e.Energy
>= 20))) AND jarakMusuh < 250 THEN
        SET isRamming = TRUE
```

```
        SET rammingTimeout = 0
        Panggil LockRadarOnTarget(e)    // Mengunci radar pada target
        Panggil PerformSuperRamming(e)  // Menyerang dengan ramming
    END IF
END EVENT

    // Event untuk menghindari serangan ketika terkena peluru
ON EVENT HitByBullet(e)
    IF NOT isRamming THEN
        PerformAdvancedDodging() // Panggil fungsi gerakan menghindar
acak
    END IF
END EVENT


// Event ketika menabrak dinding
ON EVENT HitWall(e)
    IF isRamming THEN
        SetBack(50)  // Mundur sejauh 50 unit jika dalam mode ramming
    END IF
    SetTurnRight(135 ± 30)  // Putar kanan sebesar 135° ± 30° secara
acak untuk keluar dari dinding
    SetForward(150 ± 30)    // Maju sejauh 150 ± 30 unit secara acak
END EVENT

// Event ketika bertabrakan dengan bot lain
ON EVENT HitBot(e)
    IF isRamming AND e.Energy > 0 THEN
    SetForward(9999)  // Maju terus dengan intensitas tinggi
ELSE
    SetBack(100 ± 20)  // Mundur secara acak untuk menghindar dan
memposisikan ulang
END IF
END EVENT

// Fungsi untuk melakukan gerakan menghindar acak
FUNCTION PerformAdvancedDodging()
    SET moveChoice = Angka acak antara 0 dan 4
    SWITCH moveChoice:
        CASE 0:
            SetForward(100)      // Maju 100 unit
            SetTurnRight(45)     // Putar kanan 45 derajat
        CASE 1:
            SetBack(90)          // Mundur 90 unit
            SetTurnLeft(50)      // Putar kiri 50 derajat
        CASE 2:
            SetForward(70)       // Maju 70 unit
            SetTurnRight(30)     // Putar kanan 30 derajat
```

```
            SetBack(50)          // Mundur 50 unit
        CASE 3:
            SetBack(110)         // Mundur 110 unit
            SetTurnLeft(40)      // Putar kiri 40 derajat
        CASE 4:
            SetForward(80)       // Maju 80 unit
            SetTurnRight(20)     // Putar kanan 20 derajat
            SetBack(60)          // Mundur 60 unit
END FUNCTION

// Fungsi untuk mengunci radar pada target
FUNCTION LockRadarOnTarget(e)
    CALCULATE angleToEnemy = BearingTo(e.X, e.Y) - RadarDirection
    SetTurnRadarRight(Normalized(angleToEnemy))  // Putar radar
sebesar sudut yang dinormalisasi
END FUNCTION

// Fungsi untuk melakukan serangan super ramming
FUNCTION PerformSuperRamming(e)
    WHILE isRamming AND e.Energy > 0
        CALCULATE angleToEnemy = BearingTo(e.X, e.Y) - Direction
        SetTurnRight(Normalized(angleToEnemy))  // Putar badan menuju
target
        SetForward(9999)  // Maju terus untuk tabrak musuh
    END WHILE

    // Jika energi musuh habis, hentikan mode ramming dan lakukan
penghindaran
    IF e.Energy <= 0 THEN
        SET isRamming = FALSE
        PerformAdvancedDodging()  // Lakukan penghindaran setelah
ramming selesai
    END IF
END FUNCTION

// Fungsi untuk menormalisasi sudut menjadi dalam rentang -180° hingga
180°
FUNCTION Normalized(angle)
    angle = angle MOD 360
    IF angle > 180 THEN
        angle = angle - 360
    ELSE IF angle < -180 THEN
        angle = angle + 360
    END IF
    RETURN angle  // Kembalikan sudut yang sudah dinormalisasi
END FUNCTION

END Amethyst
```

b. Struktur Data

Bot ini menggunakan struktur data primitif dari bahasa pemrograman C# untuk menyimpan informasi pertempuran. Berikut beberapa variabel utama yang digunakan:

- Enemies (HashSet<int>) → Menyimpan daftar ID bot musuh yang terdeteksi.
- isRamming (bool) → Menyimpan status apakah bot sedang melakukan strategi "ramming".
- rammingTimeout (int) → Menyimpan waktu maksimum bot dalam mode "ramming".
- maxRammingTimeout (int) → Batas waktu maksimal sebelum bot kembali ke mode normal.

c. Fungsi dan Prosedur yang Digunakan dalam Solusi Greedy

Bot ini menggunakan algoritma greedy yang menargetkan musuh saat kondisi dianggap paling menguntungkan, yaitu saat energi musuh rendah. Fungsi dan prosedur yang digunakan dikategorikan berdasarkan proses utama sebagai berikut:

1. Pemindaian & Pemilihan Target *(Greedy Selection)*
   Bot selalu mencari target baru untuk dikejar pada setiap iterasi.
   a. `TurnRadarRight(60)` → Memutar radar sebesar 60° untuk mencari target baru.
   b. `LockRadarOnTarget(e)` → Mengunci radar ke arah target agar bot tetap dapat mendeteksi posisinya.
   c. `DistanceTo(x,y)` → Menghitung jarak bot ke target yang terdeteksi.
   d. `BearingTo(x, y)` → Menghitung arah relatif bot ke target.
   e. `OnScannedBot(e)` → Jika musuh yang terdeteksi memiliki energi ≤ 40, atau hanya tersisa satu musuh dan bot memiliki selisih energi ≥ 20, maka bot akan langsung memulai strategi ramming untuk menyerang musuh secara agresif. Setelah target ditentukan, bot akan mengunci radar pada target agar tetap bisa memantau pergerakan musuh. Selanjutnya, bot akan mengeksekusi strategi ramming dengan memanggil fungsi `PerformSuperRamming(e)` untuk mendekati dan menyerang musuh tanpa mempertimbangkan strategi defensif.

2. Menyerang Target *(Greedy Attack)*
   Begitu musuh dengan energi rendah terdeteksi, yaitu ≤ 40 atau hanya tersisa satu musuh dan bot memiliki selisih energi ≥ 20, bot akan langsung mengubah strategi dari defensif menjadi agresif.

a. `PerformSuperRamming(e)` → Bot akan menjalankan strategi menabrak musuh hingga musuh kehabisan energi. . Selama musuh masih memiliki energi > 0, bot akan terus mengejarnya dengan menghitung sudut ke musuh menggunakan `BearingTo(e.X, e.Y) - Direction`, lalu mengarahkannya dengan `TurnRight(NormalizeAngle(angleToEnemy))` sebelum maju secara agresif dengan `Forward(9999)`. Jika musuh sudah tidak memiliki energi, bot akan segera menghentikan mode ramming dan beralih ke strategi menghindar dengan `PerformAdvancedDodging()` hingga memperoleh kesempatan kembali untuk menyerang.

3. Pergerakan Bot *(Greedy Movement)*
   Bot Amethyst akan selalu bergerak secara aktif untuk menghindari serangan musuh dan terus mencari peluang untuk memperoleh target baru.

   a. `PerformAdvancedDodging()` → Bot akan memilih salah satu dari lima pola pergerakan secara acak untuk menghindar dari serangan musuh.
   b. `Forward(distance), Back(distance)` → Bot akan maju atau mundur sejauh *distance* unit untuk menghindari atau mendekati target.
   c. `TurnRight(angle), TurnLeft(angle)` → Bot akan berputar sebesar *angle* derajat untuk mengubah arah dan mencari posisi yang lebih baik.
   d. `TurnRadarRight(60)` → Bot akan memutar radar sebesar 60° untuk mencari target baru jika tidak sedang dalam mode *ramming*.

4. Penanganan Tabrakan dan Serangan (*Greedy Handler)*
   Jika bot terkena tembakan, menabrak tembok, atau bertabrakan dengan bot lain, bot akan menyesuaikan pergerakannya untuk mempertahankan strategi menghindar hingga memperoleh kesempatan untuk menyerang.

   a. `ON EVENT OnHitByBullet(e)` → Jika bot terkena peluru dan tidak sedang dalam mode *ramming*, bot akan segera melakukan manuver menghindar menggunakan strategi `PerformAdvancedDodging()` untuk mengurangi kemungkinan terkena serangan berikutnya.
   b. `ON EVENT OnHitWall(e)` → Jika bot menabrak tembok, dalam mode *ramming*, bot akan mundur sejauh 50 unit sebelum melanjutkan pergerakan. Setelah itu, bot akan berputar ke kanan sebesar 135 derajat dengan tambahan nilai acak antara -30 hingga 30 derajat, lalu maju sejauh 150 unit ditambah nilai acak antara -30 hingga 30 unit untuk menjauhi tembok dan kembali ke pertempuran.

c. `ON EVENT OnHitBot(e)` → Jika bot bertabrakan dengan musuh, maka jika sedang dalam mode *ramming* dan musuh masih memiliki energi, bot akan terus maju tanpa henti. Namun, jika tidak dalam mode *ramming*, bot akan mundur sejauh 100 unit dengan tambahan nilai acak antara -20 hingga 20 unit untuk menghindari tabrakan lebih lanut dan mengatur ulang posisi.

4.4. Robocode 4

a. Pseudocode

```
// Inisialisasi bot
BEGIN kakureMomojiri

      SET BodyColor = Pink
      SET TurretColor = PeachPuff
      SET RadarColor = Pink
      SET BulletColor = Fuchsia

    START BOT

    // Loop utama bot
    WHILE IsRunning DO
        Panggil Radar()                   // Putar radar 360° untuk memindai
musuh
        SET closeToWall = CloseToWall()   // Cek apakah bot terlalu dekat
dengan dinding
        IF closeToWall > 0 THEN
            Panggil AvoidWalls()          // Hindari dinding jika terlalu
dekat
        END IF
        Panggil Move()                    // Lakukan gerakan zig-zag
    END WHILE

END kakureMomojiri

// Event ketika mendeteksi bot musuh
ON ScannedBot(e)
    SET GunAngle = GunBearingTo(e.X, e.Y)
    TurnGunLeft(GunAngle)

    IF ABS(GunAngle) <= 2.5 AND GunHeat == 0 AND Energy > 10 THEN
        Fire(MIN(2.5, Energy - 0.1))
    END IF

    IF GunAngle == 0 THEN           //Melakukan pemindaian ulang,
menghindari bot berhenti memindai
        Rescan()
    END IF
END EVENT

// Event ketika bertabrakan dengan bot lain
ON HitBot(e)
    SetTurnLeft(90)
    Forward(200)
```

```
END EVENT

// Event ketika terkena peluru
ON HitByBullet(e)
    TurnRight(30)
    Forward(200)
END EVENT

// Event ketika menabrak dinding
ON HitWall(e)
    SetTurnLeft(90)
    Forward(150)
END EVENT

// Fungsi untuk mengevaluasi apakah bot terlalu dekat dengan dinding
FUNCTION CloseToWall()
    IF X <= 60 OR X >= ArenaWidth - 60 OR Y <= 60 OR Y >= ArenaHeight - 60
THEN
        RETURN 1                        // Bot terlalu dekat dengan dinding
    ELSE
        RETURN 0                        // Bot berada di posisi aman
    END IF
END FUNCTION

// Fungsi untuk menghindari dinding
FUNCTION AvoidWalls()
    SetTurnLeft(90)
    Forward(150)
END FUNCTION

// Fungsi untuk gerakan zig-zag
FUNCTION Move()
    SetTurnLeft(30)
    Forward(100)
    SetTurnRight(90)
    Forward(100)
END FUNCTION

// Fungsi untuk memutar radar
FUNCTION Radar()
    TurnRadarRight(360)
END FUNCTION

END kakureMomojiri
```

b. Struktur Data

Bot ini menggunakan struktur data primitif dari bahasa pemrograman untuk menyimpan informasi pertempuran. Berikut beberapa variabel utama yang digunakan:

- closeToWall (integer) → Menyimpan status jarak bot terhadap dinding (1 jika terlalu dekat, dan 0 jika aman).
- gunAngle (double) → Menyimpan sudut relatif dari meriam bot ke posisi musuh yang terdeteksi.
- energy (double) → Menyimpan status energi bot untuk menentukan kapan menembak.

c. Fungsi dan Prosedur yang Digunakan dalam Solusi Greedy

Bot ini menggunakan algoritma greedy yang mengutamakan gerakan zig-zag untuk menghindari serangan musuh, menyerang secara efisien, dan menghindari dinding. Berikut Fungsi dan prosedur yang digunakan berdasarkan proses utama:

1. Pemindaian dan Pemilihan Target (*Greedy Selection*)
   Bot selalu memindai arena untuk mendeteksi musuh terdekat
   - `Radar()` → Memutar radar 360° untuk memindai musuh secara penuh.
   - `GunBearingTo(x, y)` → Menghitung sudut meriam relatif terhadap posisi musuh untuk menyesuaikan tembakan.
   - `IF ABS(gunAngle) ≤ 2.5 AND GunHeat == 0 AND Energy > 10 THEN Fire()` → Menargetkan musuh hanya jika sudut tembak cukup kecil dan energi cukup.

2. Menyerang Target (*Greedy Attack*)
   Jika musuh terdeteksi, bot akan menembak dengan kekuatan optimal sesuai dengan energi dan sudut tembak
   - `Fire(MIN(2.5, Energy - 0.1))` → Menembak musuh dengan kekuatan optimal berdasarkan energi yang tersedia.

3. Pergerakan Bot (*Greedy Movement*)
   Bot selalu bergerak dalam pola zig-zag untuk menghindari menjadi target tetap.
   - `Move()` → Gerakan zig-zag maju 100 unit, belok kiri 30 derajat, maju 100 unit, lalu belok kanan 90 derajat.
   - `AvoidWalls()` → Jika terlalu dekat dengan dinding, bot akan berbelok derajat dan mundur sejauh 150 unit untuk kembali ke posisi aman.

4. Penanganan Kolisi (Greedy Handler)

Jika bot bertabrakan dengan musuh, dinding, atau terkena peluru, bot segera mengambil tindakan untuk menghindari kerugian lebih lanjut.

- `ON EVENT OnHitBot(e)` → Bot akan berbelok 90 derajat ke kiri dan maju 200 unit untuk menghindari tabrakan lanjutan..
- `ON EVENT OnHitWall(e)` → Bot akan berbelok 90 derajat ke kiri dan maju 150 unit untuk menjauh dari dinding. Bot berbelok dengan derajat yang tinggi dan pergerakan yang rendah untuk menghindari tabrakan lebih lanjut serta mengurangi dampak pengurangan energi yang lebih besar.
- `ON EVENT OnHitByBullet(e)` → Bot akan berbelok 30 derajat ke kanan dan maju 200 unit untuk menghindari peluru berikutnya. Bot belok dengan derajat yang lebih rendah dan maju lebih cepat, untuk menghindari tembakan peluru yang cepat.

4.5. Pengujian Seluruh Robocode

Keempat bot telah diuji dalam tiga pertandingan 4-way battle (1 vs 1 vs 1 vs 1) yaitu Bakwan Jagung 1.0 (Ako) vs Bakwan Jagung Alt 1 1.0 (Handoko) vs Bakwan Jagung Alt 2 2.0 (Amethyst) vs Bakwan Jagung Alt 3 3.0 (KakureMomojiri). Berikut adalah hasil dari ketiga pengujian:

Results for 10 rounds

Rank	Name	Total Score	Survival	Surv. Bon...	Bullet D...	Bullet Bo...	Ram Dmg.	Ram Bon...	1sts	2nds	3rds
1	Bakwan Jagung 1.0	3909	1100	210	1248	189	1000	162	8	0	0
2	Bakwan Jagung Alt 1...	2762	700	30	864	18	1021	128	1	8	0
3	Bakwan Jagung Alt 3...	801	500	0	130	7	163	0	0	0	6
4	Bakwan Jagung Alt 2...	699	350	0	0	0	349	0	0	1	3

Results for 10 rounds

Rank	Name	Total Score	Survival	Surv. Bon...	Bullet D...	Bullet Bo...	Ram Dmg.	Ram Bon...	1sts	2nds	3rds
1	Bakwan Jagung 1.0	4375	1050	150	1440	203	1300	232	7	3	0
2	Bakwan Jagung Alt 1...	2776	800	90	736	105	959	86	3	5	2
3	Bakwan Jagung Alt 2...	1081	600	0	0	0	481	0	0	1	5
4	Bakwan Jagung Alt 3...	740	400	30	156	7	146	0	0	1	3

Results for 10 rounds

Rank	Name	Total Score	Survival	Surv. Bon...	Bullet D...	Bullet Bo...	Ram Dmg.	Ram Bon...	1sts	2nds	3rds
1	Bakwan Jagung 1.0	4053	800	60	1472	130	1348	242	7	1	1
2	Bakwan Jagung Alt 1...	2662	900	120	704	139	725	74	1	8	0
3	Bakwan Jagung Alt 3...	771	350	30	182	19	190	0	1	0	4
4	Bakwan Jagung Alt 2...	751	500	30	0	0	221	0	0	0	4

Tabel 4.1. Tangkapan layar hasil pengujian sebanyak tiga kali

4.6. Analisis Pengujian

Berdasarkan hasil dari 4.5 Pengujian Seluruh Robocode, Bakwan Jagung 1.0 (Ako) selalu berada di peringkat pertama dengan selisih skor yang sangat besar dibandingkan peringkat kedua, yaitu Bakwan Jagung Alt 1 1.0 (Handoko). Peringkat Bakwan Jagung Alt 2 2.0 (Amethyst) dan Bakwan Jagung Alt 2 3.0 (kakureMomojiri) saling bertukar posisi antara peringkat ketiga dan keempat. Berikut merupakan analisis strategi dari keempat bot:

a. Analisis untuk Bakwan Jagung 1.0 (Ako)

Bot Ako memiliki strategi agresif yang fokus pada serangan langsung ke musuh dengan pola ramming. Strategi ini membuat Ako menjadi ancaman bagi bot yang memiliki pola penghindaran lebih pasif, seperti Amethyst dan KakureMomojiri. Bot yang ditargetkan oleh Ako sering kesulitan menghindar, terutama jika terjebak di dekat dinding atau di antara bot lainnya. Pendekatan agresif ini menghasilkan skor yang sangat tinggi dibandingkan Handoko, Amethyst dan kakureMomojiri. Dengan demikian, strategi greedy pada bot ini sudah dapat dinilai optimal.

b. Analisis untuk Bakwan Jagung Alt 1 1.0 (Handoko)

Bot Handoko memiliki strategi yang lebih hati-hati, meskipun pergerakannya mirip dengan Ako. Handoko mendekati target dengan presisi dan menyesuaikan arah serangan tanpa langsung melakukan ramming. Hal ini membuat Handoko lebih efektif dalam memanfaatkan peluang tanpa risiko yang tinggi. Namun, karena pola gerakan dengan Ako yang mirip kadang menyebabkan keduanya saling mengejar atau melakukan ramming satu sama lain. Namun, Handoko mampu menghindari serangan Ako dengan baik. Bisa disimpulkan bahwa strategi pada bot ini diniai optimal dengan serangan yang lebih hati-hati juga penanganan kolisi yang sudah tepat.

c. Analisis untuk Bakwan Jagung Alt 2 2.0 (Amethyst)

Walaupun digunakan metode *dodging* yang variatif, pola penghindaran Amethyst masih kurang efektif melawan bot yang agresif dan akurat, sehingga sering kali menjadi target mudah dalam situasi pertempuran. Selain itu, Amethyst terlalu bergantung pada ramming sebagai satu-satunya metode untuk memberikan damage, yang membuatnya tidak mampu bersaing dengan bot yang dapat menyerang dari jarak jauh. Kelemahan ini menjadikan strategi greedy Amethyst kurang optimal dalam pertandingan, sehingga dibutuhkan pengembangan lebih lanjut.

d. Analisis untuk Bakwan Jagung Alt 3 3.0 (KakureMomojiri)

Penghindaran bot kakureMomojiri kurang tepat karena pola zig-zag yang digunakan tidak cukup optimal untuk menghindari serangan dari bot lain seperti Ako dan Handoko. Selain itu, akurasi penembakan kakureMomojiri juga masih kurang tepat sasaran. Penanganan terhadap ramming dari bot lain juga kurang efektif, menyebabkan bot ini mudah terjebak atau menerima kerusakan besar saat bertabrakan. Dengan demikian, dapat disimpulkan bahwa strategi greedy kakureMomojiri belum cukup optimal untuk bersaing dengan bot lain yang lebih agresif.

BAB 5 KESIMPULAN DAN SARAN

5.1. Kesimpulan

Setelah mengaplikasikan strategi greedy pada permainan Robocode Tank Royale, kelompok kami memperoleh kesimpulan bahwa algoritma greedy yang diterapkan dalam pengembangan bot pada permainan ini terbukti cukup efektif dalam menghasilkan strategi berpikir bot. Pengaplikasian strategi greedy pada program bot memungkinkan bot untuk dapat mengambil keputusan terbaik pada setiap giliran dalam menghadapi lawan dan mencapai skor optimal. Performa setiap bot sangat ditentukan pada kode program yang diterapkan, sehingga kualitas implementasi algoritma sangat mempengaruhi efektivitas dan adaptabilitas bot dalam permainan.

Selain itu, pengembangan empat bot dengan berbagai alternatif strategi greedy yang berbeda juga membantu kami untuk lebih memahami fleksibilitas algoritma greedy dalam menyelesaikan masalah dengan berbagai pendekatan. Eksplorasi dan variasi strategi yang diterapkan memperdalam pemahaman kami tentang cara memaksimalkan potensi algoritma greedy, mengeksplorasi heuristic yang paling efektif, serta belajar cara untuk mengembangkan bot dengan logika pengambilan keputusan yang adaptif.

5.2. Saran

Berdasarkan pengalaman yang kami dapat selama pengerjaan proyek ini, kami menyarankan beberapa hal yang bisa ditingkatkan, yaitu sebagai berikut:

1. Pengujian Lebih Lanjut
   Diperlukan lebih banyak pengujian lebih lanjut dengan berbagai skenario yang bervariasi, seperti jumlah bot lawan yang lebih banyak dan ukuran arena yang lebih besar. Hal ini akan membantu untuk memastikan bahwa strategi algoritma greedy bot dapat beradaptasi dalam berbagai kondisi dan tetap efektif.

2. Pengembangan Strategi Greedy
   Diperlukan lebih banyak eksplorasi untuk menggali lebih dalam berbagai varian strategi greedy yang belum dieksplorasi. Hal ini akan membantu mengembangkan bot agar dapat membuat keputusan lebih baik terkait berbagai situasi dalam pertandingan.

3. Analisis dan Evaluasi Lebih Detail
   Diperlukan analisis dan evaluasi lebih mendalam terhadap performa bot selama pertandingan. Hal ini bisa dapat berupa mencatat statistik kemenangan, skor bot, serta mengidentifikasi kelemahan bot dalam berbagai situasi pertempuran. Data yang dikumpulkan

akan dapat berguna untuk membantu memperbaiki dan mengembangkan strategi greedy bot agar menjadi lebih efektif.

LAMPIRAN

Tautan Repository GitHub:

https://github.com/susTuna/Tubes1_Bakwan-Jagung

Tautan Video YouTube:

https://youtu.be/N_Fr1SNHK_A?si=SSkqRPy-f_21ZzFc

Tautan Video Google Drive:

https://drive.google.com/file/d/19vRXk8XtPNUwiuRdkpi-ucjpUGaEJYFR/view?usp=drivesdk

Tabel Pengecekan Fitur:

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.	✓	

DAFTAR PUSTAKA

Munir, R. (2025). *Algoritma Greedy - Bagian 1*. Diakses pada 23 Maret 2025, dari https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2024-2025/04-Algoritma-Greedy-(2025)-Bag1.pdf

Munir, R. (2025). *Algoritma Greedy - Bagian 2*. Diakses pada 23 Maret 2025, dari https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2024-2025/05-Algoritma-Greedy-(2025)-Bag2.pdf

Munir, R. (2025). *Algoritma Greedy - Bagian 3*. Diakses pada 23 Maret 2025, dari https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2024-2025/06-Algoritma-Greedy-(2025)-Bag3.pdf