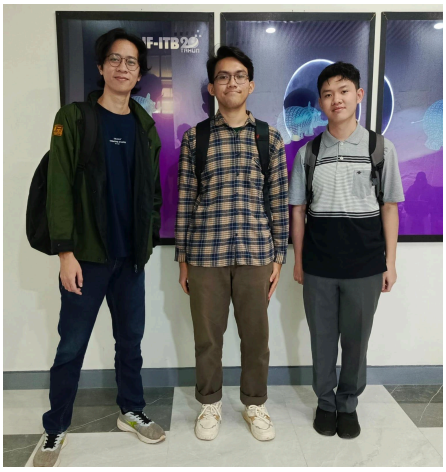
TUGAS BESAR 1 IF2211
Strategi Algoritma



Sabilul Huda	13523072
Henry Filberto Shenelo	13523108
Andrew Isra Saputra DB	13523110

Diampu oleh:

Dr. Ir. Rinaldi Munir, M.T.

Disusun oleh:

Kelompok 36
“Cool”

SEKOLAH TEKNIK ELEKTRO DAN INFORMATIKA
INSTITUT TEKNOLOGI BANDUNG

2025

Daftar Isi

BAB 1: DESKRIPSI TUGAS

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

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

Komponen-komponen dari permainan ini antara lain:

1. Rounds dan Turns

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

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

   Pada setiap turn, sebuah bot dapat:

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

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

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

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

2. Batas Waktu Giliran

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

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

3. Energi

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

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

4. Peluru

   Semakin banyak energi (daya tembak) yang digunakan untuk menembakkan peluru, semakin berat peluru tersebut dan semakin lambat gerakannya. Namun, peluru yang lebih

berat juga menghasilkan lebih banyak kerusakan dan memungkinkan bot mendapatkan lebih banyak energi saat mengenai bot musuh.

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

5. Panas Meriam (Gun Heat)

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

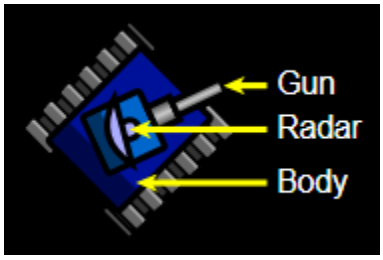
6. Tabrakan

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

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

7. Bagian Tubuh Tank

   Tubuh tank terdiri dari 3 bagian:



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

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

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

8. Pergerakan

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

9. Berbelok

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

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

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

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

10. Pemindaian

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

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

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

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

11. Skor

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

- **Bullet Damage:** Bot mendapatkan **poin sebesar *damage*** yang dibuat kepada bot musuh menggunakan peluru.
- **Bullet Damage Bonus:** Apabila peluru berhasil membunuh bot musuh, bot mendapatkan **poin sebesar 20% dari *damage*** yang dibuat kepada musuh yang terbunuh.
- **Survival Score:** Setiap ada bot yang mati, bot 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.

Spesifikasi Wajib

1. 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.
2. Tugas dikerjakan berkelompok dengan anggota **minimal 2 orang** dan **maksimal 3 orang**, boleh lintas kelas dan lintas kampus.
3. Strategi *greedy* yang diimplementasikan setiap kelompok harus dikaitkan dengan fungsi objektif dari permainan ini, yaitu memperoleh skor setinggi mungkin pada akhir pertempuran. Hal ini dapat dilakukan dengan mengoptimalkan komponen skor yang telah dijelaskan diatas.
4. Strategi *greedy* yang diimplementasikan **harus berbeda** untuk setiap bot yang diimplementasikan dan setiap strategi greedy harus menggunakan **heuristic** yang berbeda.
5. **Bot yang dibuat TIDAK BOLEH sama dengan SAMPEL yang diberikan sebagai CONTOH. Baik dari starter pack maupun dari repository engine asli.**
6. Buatlah strategi *greedy* terbaik, karena setiap **bot utama** dari masing-masing kelompok akan diadu dalam kompetisi Tubes 1.
7. 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.
8. 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.
9. 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
10. 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

Spesifikasi Bonus

1. (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.**
2. (maks 10) Beberapa kelompok pemenang lomba kompetisi akan mendapatkan nilai tambahan berdasarkan posisi yang diraih.

BAB 2: LANDASAN TEORI

2.1 Definisi

Algoritma greedy adalah algoritma yang memecahkan persoalan secara langkah per langkah sedemikian sehingga pada setiap langkah akan diambil pilihan yang terbaik yang dapat diperoleh pada saat itu tanpa menghiraukan konsekuensi selanjutnya (prinsip "take what you can get now!") dan berharap dengan memilih optimum lokal pada setiap langkah akan didapatkan hasil akhir yang optimum global. Algoritma greedy adalah algoritma yang paling populer dan sederhana untuk memecahkan masalah optimasi. Persoalan optimasi adalah persoalan mencari solusi yang optimal (maksimasi atau minimasi).

2.2 Elemen-Elemen pada Algoritma Greedy

Berikut adalah elemen-elemen yang digunakan pada algoritma *greedy*:

1. Himpunan kandidat,C : berisi kandidat yang akan dipilih pada setiap langkah (misal: simpul/sisi di dalam graf, job, task, koin, benda, karakteristik, dsb)
2. Himpunan solusi,S : berisi kandidat yang sudah dipilih
3. Fungsi solusi: menentukan apakah himpunan kandidat yang dipilih sudah memberikan solusi
4. Fungsi seleksi (selection function): memilih kandidat berdasarkan strategi *greedy* tertentu. Strategi *greedy* ini bersifat heuristik
5. Fungsi kelayakan (feasible): memeriksa apakah kandidat yang dipilih dapat dimasukkan ke dalam himpunan solusi (layak atau tidak)
6. Fungsi obyektif: memaksimumkan atau meminimumkan.

Berdasarkan elemen-elemen di atas, dapat dikatakan bahwa, algoritma *greedy* melibatkan pencarian sebuah himpunan bagian, S, dari himpunan kandidat, C, yang dalam hal ini, S harus memenuhi beberapa kriteria yang ditentukan, yaitu S menyatakan suatu solusi dan S dioptimasi oleh fungsi objektif.

2.3 Cara Kerja Program

Robocode Tank Royale adalah game strategi berbasis giliran (*turn-based strategy*) dalam bentuk simulasi pertempuran tank. Pemain membuat bot yang diprogram untuk bertahan dan mengalahkan lawan hingga menjadi yang terakhir bertahan. Setiap bot berinteraksi melalui Application Programming Interface (API). Secara umum, alur kerja bot dalam Robocode Tank Royale adalah sebagai berikut:

a. Bot menerima event dan data dari game engine melalui API, seperti lokasi musuh, peluru yang menghantam, atau batas arena. Bot memproses data yang diterima.

b. Bot mengambil keputusan dan menjalankan perintah berdasarkan data yang diterima dan strategi yang telah diprogram, bot menentukan tindakan terbaik, misalnya Fire(), TurnRadarRight(), san masih banyak lagi.
c. Bot mengirim perintah kembali ke game engine melalui API, melaksanakan aksi yang dipilih. Setelah itu, bot menunggu *turn* berikutnya hingga pertandingan berakhir.

Untuk mengimplementasikan algoritma *greedy* ke dalam bot, pemain bisa membuat logika yang selalu memilih aksi terbaik di setiap langkah berdasarkan data yang diterima untuk mendapatkan poin terbanyak di akhir *battle.* Misalnya, bot diprogram untuk selalu menembak sebanyak mungkin (*greedy)* untuk selalu menembak untuk mendapatkan poin terbanyak dengan API Fire() di setiap *turn*-nya.

Cara menjalankan bot secara garis besar di Robocode Tank Royale dilakukan dengan:

a. Pemain membuat kode bot dalam bahasa pemrograman yang didukung (dalam tugas ini menggunakan C# (.cs)) memanfaatkan *event* dan aksi yang tersedia melalui API.
b. Build dan compile bot menggunakan .NET Core untuk C#. Pastikan semua file yang diperlukan (seperti BotTemplate.json, BotTemplate.csproj) ada dalam folder yang sama.
c. Menjalankan game engine dengan menjalankan file .jar dari Robocode Tank Royale GUI, lalu melakukan konfigurasi dan *booting* bot-bot yang akan dipertandingkan pada *game engine* menggunakan alamat server lokal.
d. Klik “Start Battle” dan mengamati *battle* hingga selesai.

Pada umumnya, setiap bot akan memiliki struktur folder sebagai berikut:

/BotTemplate

|-- BotTemplate.json

|-- BotTemplate.cs

|-- BotTemplate.csproj

|-- BotTemplate.cmd

|-- BotTemplate.sh

Adapun isi file .cs harus setidaknya mengandung sebagai berikut:

```
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
             Forward(100);
            TurnGunRight(360);
            Back(100);
           TurnGunRight(360);

        }
    }
}
```

BAB 3: APLIKASI STRATEGI GREEDY

3.1 Mapping persoalan *Robocode Tank Royale* Menjadi Elemen-Elemen Algoritma *Greedy*

a. Himpunan Kandidat, C: Semua aksi yang dapat dilakukan bot dalam satu *turn*. Hal-hal utama yang dapat dilakukan adalah:
    1. Menembakkan peluru (Fire)
    2. Bergerak maju/mundur (Forward/Back)
    3. Memutar radar sebanyak x derajat (TurnRadarLeft/TurnRadarRight)
    4. Memutar *body* sebanyak x derajat (TurnrLeft/TurnRight)
    5. Memutar *gun* sebanyak x derajat (TurnGunLeft/TurnGunRight)

    Semua aksi ini dapat dilakukan bersamaan dengan tambahan kata "Set" di awal perintah dan Go() untuk mengeksekusi. Aksi-aksi ini dapat berupa reaksi dari *event* yang terjadi maupun aksi aktif dari bot tersebut itu sendiri.
b. Himpunan Solusi, S: Rangkaian aksi yang dipilih untuk dieksekusi oleh bot dalam satu *turn*. Contoh:

    ```
    SetForward(100)
    TurnRadarLeft(45)
    Fire(3)
    Go();
    ```
c. Fungsi Solusi: Menentukan apakah kumpulan dari solusi yang telah dikumpulkan sebelumnya dapat menjalankan bot pada setiap *turn* hingga berakhirnya suatu *round* dan memperoleh poin.
d. Fungsi Seleksi: Menentukan solusi menggunakan strategi *greedy* tertentu. Fungsi ini bersifat heuristik, artinya dirancang untuk menemukan solusi yang paling optimal secara praktis, meskipun belum tentu optimal secara matematis. Seleksi ini dapat menekankan pada *bullet damage, ram damage,* ataupun *survival score.*
e. Fungsi Kelayakan: Memeriksa apakah aksi valid berdasarkan kondisi bot saat ini. Misalnya, apakah *gun heat* = 0 saat menembak, tidak menabrak dinding, sudut putaran radar ≤ 45° tiap *turn*.
f. Fungsi Objektif: Mendapatkan akumulasi poin tertinggi dari keseluruhan *round* yang ada.

3.2 Eksplorasi Alternatif Solusi

a. Main Bot (BOTanchan)

    BOTanchan menggunakan algoritma *greedy* dengan fokus memaksimalkan *damage* berdasarkan jarak ke musuh. Bot ini selalu menargetkan satu musuh terdekat dan menyesuaikan daya tembakan (bullet power) secara *greedy*: semakin dekat musuh, semakin tinggi daya tembakan yang digunakan. Untuk bertahan, bot bergerak secara acak dan mundur jika energi rendah, untuk menghindari tembakan musuh. Keputusan diambil

berdasarkan jarak dan energi musuh. Jika energi sangat rendah, bertambah agresif dan menyerang bot lain.

b. Alternative Bot 1 (PraBOTwo)

PraBOTwo mengimplementasikan strategi *orbital targeting* dengan agresivitas yang menyesuaikan berdasarkan level energi musuh. Bot ini fokus pada satu target dalam satu waktu, mengatur pergerakan dan daya tembak berdasarkan energi target. Target dengan energi tinggi diserang dari jarak jauh dengan peluru berdaya rendah sembari berputar-putar untuk menghindar dari peluru, sedangkan target dengan energi rendah didekati secara agresif dengan tembakan berdaya tinggi.

c. Alternative Bot 2 (LittleBot)

LittleBot merupakan bot dengan algoritma Greedy yang berusaha memaksimalkan pemindaian bot sebanyak mungkin. Algoritma ini memandang prinsip bahwa semakin banyak bot yang dipindai – dan tentunya segera dieksekusi, maka semakin banyak peluang untuk mengeluarkan peluru yang mengenai lawan. Prinsip ini bekerja dengan asumsi bahwa dalam sebuah game terdapat banyak *bot participants.* Strategi ini direalisasikan beberapa tahap, yaitu bot akan berusaha menuju ke daerah pusat arena dengan jari-jari 80 satuan (tingginya peluang bot tertabrak atau ditembak menjadi tantangan), bot berputar dengan kecepatan linear tertentu, bot memindai musuh sambil berputar, dan bot akan berusaha menuju ke pusat lagi jika mengenai tembok (tidak 100% menuju ke pusat agar perilakunya lebih dinamis).

d. Alternative Bot 3 (BOTentahlah)

BOTentahlah menggunakan algoritma greedy dengan fokus utama menyerang musuh berdasarkan jarak musuh dengan bot tersebut. Musuh dengan jarak < 50 pixels akan ditabrak sehingga memungkinkan bot mendapat poin. Sedangkan musuh dengan jarak 50-600 pixels akan ditembak dengan power yang menyesuaikan jarak bot dengan musuh, semakin dekat jaraknya semakin tinggi power yang dikeluarkan, ini memungkinkan bot mendapat poin yang maksimum dan mengurangi kemungkinan energy loss karena peluru yang tidak tepat sasaran .

3.3 Analisis efisiensi dan efektivitas

a. Main Bot (BOTanchan)

Strategi untuk memilih musuh berdasarkan jarak terdekat sekaligus energi yang dimiliki cukup efisien karena tidak memerlukan perhitungan yang kompleks dan mengandalkan menghemat energi bot. Hal ini juga cukup efektif karena musuh yang lebih dekat akan lebih mungkin kena, walaupun kekurangannya karena menargetkan satu, terkadang jika radar tidak memindai musuh yang lain, serangan menjadi kurang efektif.

Dari segi gerakan yang cukup random, efektif untuk menghindari serangan musuh, tetapi buruk ketika 1 vs 1 karena jarang mendekat ke musuh dan juga gerakan yang random menyebabkan bot sering terbentur dinding.

b. Alternative Bot 1 (PraBOTwo)

PraBOTwo efektif dalam mengeliminasi musuh karena menargetkan bot-bot yang berenergi rendah dan cenderung mengeluarkan banyak peluru dengan daya yang besar sehingga musuh cepat tereliminasi. Dari segi pergerakan, bot ini lumayan efektif karena gerakan yang berputar-putar dan terkadang mendekat lebih susah diprediksi walaupun jika ada musuh yang mendekat di luar radar, bot ini cukup rentan terkena peluru.

Dari segi efisiensi, PraBOTwo mengoptimalkan penggunaan energi melalui alokasi daya tembak yang proporsional dengan kondisi musuh dan dirinya sendiri. Namun, beberapa keterbatasan masih muncul, seperti prediksi gerakan musuh yang masih linier dan pemindaian radar yang kurang optimal saat kehilangan target. Inefisiensi kecil juga terlihat saat bot terjebak di sudut arena, membutuhkan waktu beberapa tick untuk kembali ke posisi ideal. Untuk pengembangan selanjutnya, peningkatan sistem prediksi gerakan musuh dan optimasi algoritma pemindaian radar akan menjadi fokus utama.

c. Alternative Bot 2 (LittleBot)

Strategi menuju pusat lalu berputar sambil memindai musuh cukup efektif jika musuh tergolong banyak. Hal ini karena peluang bot memindai musuh ataupun menabraknya akan tinggi. Namun algoritma ini memiliki kekurangan jika bot yang tersisa hanya dua (1 vs 1) karena sangat sulit untuk mengenai peluru ke lawan dengan kondisi berputar.

d. Alternative Bot 3 (BOTentahlah)

Strategi memilih musuh berdasarkan jarak terdekat cukup efisien karena tidak memerlukan perhitungan kompleks, seperti prediksi posisi lawan atau penyimpanan data histori musuh. Dengan hanya mengandalkan hasil pemindaian terbaru, bot dapat segera memutuskan apakah akan menembak atau melakukan ramming. Hal ini juga efektif karena musuh yang lebih dekat lebih mungkin terkena serangan, terutama dengan ramming yang memberikan damage langsung. Namun, karena bot selalu menyerang target terdekat, ada kemungkinan bot mengabaikan lawan lain yang lebih berbahaya atau memiliki strategi yang lebih defensif, sehingga kurang optimal dalam situasi tertentu.

Dari segi gerakan, BOTentahlah cukup efisien karena hanya menggunakan pergerakan linear dengan kecepatan maksimum, tanpa algoritma penghindaran kompleks. Dengan terus bergerak maju dan menghindari dinding saat terlalu dekat, bot ini dapat memanfaatkan kecepatan tinggi untuk menghindari serangan lawan tanpa banyak

pengolahan data tambahan. Namun, efektivitas strategi ini berkurang jika bot terjebak dalam area sempit atau jika lawan memiliki pola tembakan prediktif, yang dapat memanfaatkan pergerakan linear ini untuk meningkatkan akurasi tembakan mereka.

Dalam situasi 1 vs 1, efektivitas BOTentahlah cukup terbatas karena bot tidak memiliki mekanisme yang secara khusus mendekati atau menjaga jarak optimal dari lawan. Selain itu, penggunaan pergerakan linear tanpa pola penghindaran yang lebih canggih dapat menyebabkan bot lebih mudah diprediksi. Meskipun strategi serangan langsungnya cukup efektif dalam pertarungan jarak dekat, kelemahan dalam manuver dan kurangnya adaptasi terhadap strategi musuh dapat menjadi tantangan saat menghadapi lawan yang lebih dinamis.

3.4 Strategi dan Pertimbangan Solusi Alternatif

a. Main Bot (BOTanchan)
Strategi 1: Memilih target berdasarkan jarak terdekat. Selalu prioritaskan 1 bot terdekat untuk diserang, karena jarak yang pendek meningkatkan akurasi dan *bullet damage* (karena daya peluru lebih tinggi pada jarak dekat). Jika ada bot lain yang lebih dekat, akan pindah fokus ke bot tersebut.

Strategi 2: Gunakan peluru dengan energi tinggi saat energi banyak dan jarak dekat, serta daya rendah saat energi sedikit atau musuh jauh. Hal ini untuk meningkatkan efektivitas serangan dan menjaga energi agar lebih lama bertahan dan peluang mendapat poin lagi dari menyerang atau *survive* lebih tinggi. Saat energi tersisa sedikit atau musuh tersisa 1, tetap menyerang dengan peluru dengan energi kecil hingga energi habis. Hal ini karena dengan bermain bertahan, kemungkinannya akan tertabrak *wall* atau diserang musuh sehingga berharap dengan menyerang secara agresif untuk mendapat energi lagi.

Strategi 3: Ketika energi tinggi (>30), bergerak maju dengan gerakan yang acak untuk menghindari serangan bot lain. Jika energi rendah (<30), mundur sambil bergerak acak jika musuh banyak, atau maju dengan lebih perlahan ketika musuh tersisa satu atau energi < 10. Strategi ini pertimbangannya adalah musuh yang bergerak dengan acak akan susah terkena peluru, sekaligus dapat mendekat untuk menyerang. Selain itu, walau cukup acak, pada suatu saat akan bergerak mendekati musuh.

b. Alternative Bot 1 (PraBOTwo)
Strategi 1: PraBOTwo memprioritaskan satu musuh yang terdeteksi radar dan akan beralih target hanya jika menemukan musuh dengan energi lebih rendah. Hal ini memaksimalkan serangan agar dapat membunuh musuh dan mendapat bonus sekaligus

mengurangi jumlah musuh. Energi yang lebih rendah juga mengindikasikan bot yang lemah.

Strategi 2: Daya peluru diatur berdasarkan energi musuh. Semakin rendah energi musuh, semakin tinggi daya peluru yang digunakan untuk memastikan eliminasi cepat. Namun, jika energi bot sendiri sangat rendah dan musuh masih kuat, PraBOTwo akan beralih ke tembakan berdaya rendah untuk menghemat energi.

Strategi 3: Bot ini menggunakan gerakan orbital/berputar-putar pada jarak tertentu untuk menghindar dari musuh. Jika musuh memiliki energi tinggi, PraBOTwo akan bergerak melingkar di sekitar musuh pada jarak aman. Jika terlalu dekat, bot akan mundur sambil melakukan manuver acak untuk menjauh. Sebaliknya, jika musuh lemah, bot akan bergerak mendekat.

c. Alternative Bot 2 (LittleBot)
Strategi 1: Berusaha menuju ke pusat arena agar kemungkinan pemindaian musuh sebanyak mungkin tinggi. Semakin banyak musuh yang dipindai, semakin banyak kemungkinan peluru yang keluar mengenai lawan. Namun terdapat resiko pada strategi ini yaitu akan menjadi pusat sasaran banyak bot lain.

Strategi 2: Gerakan melingkar dengan kecepatan dan jari-jari putar yang diperhitungkan seefektif mungkin sehingga bot susah untuk dikenai peluru tetapi mudah untuk memindai musuh. Radar dan Gun bot akan mengikuti pergerakan putaran bot sehingga bot yang akan segera bertabrakan dengan bot ini akan berpotensi hancur lebih dulu.

Strategi 3: Jika bot menabrak tembok, algoritmanya akan memerintahkan untuk mencoba menuju ke pusat kembali. Jika ternyata pada saat menuju pusat terdapat bot lain, maka perilaku bot akan berubah. Hal ini dilakukan agar perilaku bot di midgame hingga lategame lebih dinamis.

d. Alternative Bot 3 (BOTentahlah)
Strategi 1: Memilih target berdasarkan jarak terdekat. BOTentahlah selalu menyerang musuh yang paling dekat dengan bot. Jika musuh berada dalam jarak < 50 px, bot akan menabrak (ramming), sedangkan jika dalam jarak 50 – 600 px, bot akan menembak dengan kekuatan peluru yang bervariasi. Dengan strategi ini akan meningkatkan akurasi tembakan karena peluru atau proses ramming perlu waktu yang lebih singkat untuk mencapai target. Jarak dekat juga meningkatkan efektivitas ramming, yang bisa memberikan damage tambahan tanpa harus membuang energi untuk menembak.

Strategi 2: Gerakan Linear dengan Kecepatan Maksimum (Menghindari Stuck dan Menyesuaikan Arah di Dekat Dinding). BOTentahlah bergerak dalam garis lurus dengan kecepatan maksimum untuk menghindari serangan. Jika mendekati dinding, bot akan memperlambat dan mengubah arah agar tidak menabrak. Dengan strategi ini, akan mempertahankan stabilitas kecepatannya sehingga mempersulit bot untuk ditembak.

BAB 4: IMPLEMENTASI DAN PENGUJIAN

4.1 Implementasi Solusi

a. Main Bot (BOTanchan)

```
BEGIN
START BOTanchan
FUNCTION Run()
    // Set bot colors
    SET BodyColor ← DodgerBlue
    SET TurretColor ← DeepSkyBlue
    SET RadarColor ← DarkCyan
    SET BulletColor ← Crimson
    SET ScanColor ← LawnGreen
    SET TracksColor ← DarkSlateGray
    SET GunColor ← BlueViolet

    // Main bot loop
    WHILE IsRunning DO
        // Movement strategy based on energy
        IF Energy < 30 THEN
            CALL RetreatMovement()
        ELSE
            CALL AttackMovement()
        ENDIF

        // Wall avoidance check
        CALL AvoidWalls()

        // Radar targeting logic
        IF HasTarget() THEN
            CALL TrackTarget()
        ELSE
            CALL SearchForTargets()
        ENDIF

        ExecuteCommands()
    ENDWHILE
END FUNCTION

FUNCTION RetreatMovement()
    SET moveDirection ← -1
    SET randomPattern ← 45 * SIN(TurnNumber * 0.3)
    SET TurnLeft(randomPattern)

    IF EnemyCount == 1 OR Energy < 10 THEN
        SET moveDirection ← 1
    ENDIF
```

```
    SET Forward(80 * moveDirection)
END FUNCTION

FUNCTION AttackMovement()
    SET moveDirection ← 1
    SET randomPattern ← 30 * SIN(TurnNumber * 0.2) +
RANDOM(-15,15)
    SET TurnLeft(randomPattern)
    SET Forward(100 * moveDirection)
END FUNCTION

FUNCTION AvoidWalls()
    SET safeDistance ← 125
    SET closestWall ← FIND_CLOSEST_WALL()

    IF closestWall.distance < safeDistance THEN
        SET centerAngle ← DirectionTo(ArenaWidth/2, ArenaHeight/2)
        IF closestWall.side == "left" THEN
            SET TurnLeft(centerAngle + 20)
        ELSE
            SET TurnLeft(centerAngle)
        ENDIF
        SET Forward(100)
    ENDIF
END FUNCTION

FUNCTION HasTarget()
    RETURN currentTargetId != -1 AND enemyHistory CONTAINS
currentTargetId
END FUNCTION

FUNCTION TrackTarget()
    IF EnemyCount < 4 THEN
        SET AdjustRadarForBodyTurn ← TRUE
    ENDIF

    SET lastKnownPosition ← GET_LAST_TARGET_POSITION()
    SET radarTurn ← CALCULATE_RADAR_TURN(lastKnownPosition)
    SET TurnRadarLeft(radarTurn)

    IF TargetLostFor(3) THEN
        SET TurnRadarRight(45)
    ENDIF

    IF TargetLostFor(5) THEN
        SET currentTargetId ← -1
        SET AdjustRadarForBodyTurn ← FALSE
    ENDIF
END FUNCTION
```

```
FUNCTION SearchForTargets()
    SET TurnRadarLeft(45)
END FUNCTION

// Event: When another bot is scanned
FUNCTION OnScannedBot(event)
    CALL RecordEnemyData(event)

    IF ShouldSwitchTarget(event) THEN
        SET currentTargetId ← event.ScannedBotId
    ENDIF

    CALL DetermineFiringStrategy(event)
    ExecuteCommands()
END FUNCTION

FUNCTION RecordEnemyData(event)
    IF NOT enemyHistory CONTAINS event.ScannedBotId THEN
        CREATE_NEW_ENEMY_RECORD(event.ScannedBotId)
    ENDIF

    ADD_ENEMY_DATA_POINT(event)

    IF ENEMY_DATA_POINTS_COUNT(event.ScannedBotId) > 10 THEN
        REMOVE_OLDEST_DATA_POINT(event.ScannedBotId)
    ENDIF
END FUNCTION

FUNCTION DetermineFiringStrategy(event)
    SET distance ← CALCULATE_DISTANCE(event)
    SET bulletPower ← CALCULATE_OPTIMAL_POWER(distance)

    IF bulletPower > 0 THEN
        SET predictedPosition ← PREDICT_ENEMY_POSITION(event)
        SET gunAngle ← CALCULATE_GUN_ANGLE(predictedPosition)
        SET TurnGunLeft(gunAngle)

        IF GunAligned() THEN
            Fire(bulletPower)
            CALL EvasiveManeuver()
        ENDIF
    ENDIF
END FUNCTION

// Event: When hit by a bullet
FUNCTION OnHitByBullet(event)
    SET TurnLeft(45)
    SET Forward(50)
    ExecuteCommands()
END FUNCTION
```

```
// Event: When hitting a wall
FUNCTION OnHitWall(event)
    SET centerAngle ← DirectionTo(ArenaWidth/2, ArenaHeight/2)
    SET TurnLeft(centerAngle)
    SET Forward(100)
    ExecuteCommands()
END FUNCTION

// Event: When colliding with another bot
FUNCTION OnHitBot(event)
    SET TurnLeft(45)
    SET Back(100)
    ExecuteCommands()
END FUNCTION

// Event: When round starts
FUNCTION OnRoundStarted(event)
    SET moveDirection ← 1
    SET currentTargetId ← -1
END FUNCTION

END PROGRAM
```

b. Alternative Bot 1 (PraBOTwo)

```
BEGIN
START PraBOTwo
SET bot configuration

// Constants
CONSTANT OrbitalDistance = 200
CONSTANT AggressionThreshold = 50

// Variables
SET moveDirection = 1
SET enemyHistory = NEW Dictionary<int, List<EnemyData>>
SET currentTargetId = -1
SET random = NEW Random()

// Enemy data structure
CLASS EnemyData
    PUBLIC Time
    PUBLIC X
    PUBLIC Y
    PUBLIC Energy
    PUBLIC Velocity
    PUBLIC Heading
END CLASS

FUNCTION Run()
    // Set bot colors
    SET BodyColor ← DeepPink
    SET TurretColor ← Lime
    SET RadarColor ← Orange
    SET BulletColor ← BlueViolet

    // Main bot loop
    WHILE IsRunning DO
        // Basic movement pattern
        SET TurnRight(30)
        SET Forward(100)
        SET TurnRadarLeft(360)

        // Strategic behaviors
        CALL ScanTargets()
        CALL AdjustMovement()
        CALL AvoidWalls()

        ExecuteCommands()
    ENDWHILE
END FUNCTION

FUNCTION ScanTargets()
    IF currentTargetId == -1 OR NOT enemyHistory CONTAINS
currentTargetId THEN
        // Full scan when no target
```

```
        SET TurnRadarLeft(360)
    ELSE
        // Track current target
        targetData = GET_LAST_ENEMY_DATA(currentTargetId)
        radarTurn = CALCULATE_RADAR_TURN(targetData.X,
targetData.Y)
        SET TurnRadarLeft(radarTurn)

        IF TARGET_LOST_FOR(5) THEN
            SET currentTargetId = -1
        ENDIF
    ENDIF
END FUNCTION

FUNCTION AdjustMovement()
    IF currentTargetId != -1 AND enemyHistory CONTAINS
currentTargetId THEN
        targetData = GET_LAST_ENEMY_DATA(currentTargetId)
        distance = CALCULATE_DISTANCE(targetData.X, targetData.Y)

        IF targetData.Energy < AggressionThreshold THEN
            // Aggressive approach for weak enemies
            angle = CALCULATE_TARGET_ANGLE(targetData.X,
targetData.Y)
            SET TurnLeft(angle + 45)  // Orbital offset
            SET Forward(100)
        ELSE
            // Defensive movement for strong enemies
            IF distance < OrbitalDistance THEN
                SET TurnLeft(RANDOM(-30, 30))
                SET Forward(50)
            ENDIF
        ENDIF
    ELSE
        // Random movement when no target
        SET TurnRight(RANDOM(-30, 30))
        SET Forward(100 * moveDirection)
        SET moveDirection = moveDirection * -1
    ENDIF
END FUNCTION

FUNCTION OnScannedBot(event)
    CALL RECORD_ENEMY_DATA(event)

    IF currentTargetId == -1 OR SHOULD_SWITCH_TARGET(event) THEN
        SET currentTargetId = event.ScannedBotId
    ENDIF

    bulletPower = CALCULATE_BULLET_POWER(event.Energy)
    IF bulletPower > 0 THEN
        predictedPos = PREDICT_ENEMY_POSITION(event)
```

```
        fireAngle = CALCULATE_GUN_ANGLE(predictedPos)
        SET TurnGunLeft(fireAngle)

        IF GUN_ALIGNED() THEN
            Fire(bulletPower)
            SET TurnRight(20)
            SET Forward(100)
        ENDIF
    ENDIF
END FUNCTION

FUNCTION RECORD_ENEMY_DATA(event)
    IF NOT enemyHistory CONTAINS event.ScannedBotId THEN
        CREATE_NEW_ENEMY_RECORD(event.ScannedBotId)
    ENDIF

    ADD_ENEMY_DATA(
        Time: TurnNumber,
        X: event.X,
        Y: event.Y,
        Energy: event.Energy,
        Velocity: event.Speed,
        Heading: event.Direction
    )

    IF ENEMY_DATA_COUNT(event.ScannedBotId) > 10 THEN
        REMOVE_OLDEST_DATA_POINT(event.ScannedBotId)
    ENDIF
END FUNCTION

FUNCTION CALCULATE_BULLET_POWER(targetEnergy)
    IF Energy < 10 AND EnemyCount < 3 AND targetEnergy > 20 THEN
        RETURN 1.0
    ENDIF
    IF targetEnergy > 70 THEN RETURN 1.5
    IF targetEnergy > 50 THEN RETURN 2.0
    IF targetEnergy > 30 THEN RETURN 2.5
    RETURN 3.0
END FUNCTION

FUNCTION SHOULD_SWITCH_TARGET(newEnemy)
    IF currentTargetId == -1 THEN RETURN true

    currentData = GET_LAST_ENEMY_DATA(currentTargetId)
    RETURN newEnemy.Energy < currentData.Energy
END FUNCTION

FUNCTION AvoidWalls()
    margin = 100
    IF NEAR_WALL(margin) THEN
        centerAngle = CALCULATE_CENTER_ANGLE()
```

```
        SET TurnLeft(centerAngle)
        IF TURN_COMPLETE() THEN
            SET Forward(100)
        ENDIF
    ENDIF
END FUNCTION

FUNCTION OnHitByBullet(event)
    SET TurnLeft(RANDOM(-45, 45))
    SET Forward(50 * moveDirection)
    SET moveDirection = moveDirection * -1
END FUNCTION

FUNCTION OnHitBot(event)
    SET TurnLeft(30)
    SET Back(100)
    ExecuteCommands()
END FUNCTION

FUNCTION OnRoundStarted(event)
    SET moveDirection = 1
    SET currentTargetId = -1
END FUNCTION
END PROGRAM
```

c. Alternative Bot 2 (LittleBot)

```
PROGRAM LittleBot

BEGIN
    START LittleBot
    SET bot configuration (colors, name, etc.)

    FUNCTION Run()
        SET arenaWidth ← ArenaWidth
        SET arenaHeight ← ArenaHeight
        SET IsInCircle ← FALSE

        // Mengatur warna bot
        SET BodyColor ← HotPink
        SET RadarColor ← MediumPurple
        SET GunColor ← White
        SET TracksColor ← DeepPink
        SET BulletColor ← Red

        // Bergerak ke pusat arena terus menerus hingga benar-benar
berada dalam lingkaran berjari-jari 80
        WHILE NOT IsInCircle DO
            CALL MoveToAreaOfCenter(arenaWidth, arenaHeight)
            IF Distance from Center < 80 THEN
                PRINT "Yeay sudah di tengah!"
                IsInCircle ← TRUE
            ELSE
                PRINT "Belum di tengah, coba lagi"
                IsInCircle ← FALSE
            ENDIF
        ENDWHILE

        // Loop utama saat bot berjalan
        WHILE IsRunning DO
            WHILE NOT IsInCircle DO
                CALL MoveToAreaOfCenter(arenaWidth, arenaHeight)
                IF Distance from Center < 80 THEN
                    PRINT "Yeay sudah di tengah!"
                    IsInCircle ← TRUE
                ELSE
                    PRINT "Belum di tengah, coba lagi"
                    IsInCircle ← FALSE
                ENDIF
            ENDWHILE

            PRINT "Saatnya berputar!"
            SET TurnLeft(5000)
            SET MaxSpeed ← 10
            Forward(10000)
        ENDWHILE
    END FUNCTION
```

```
// Fungsi untuk bergerak ke pusat arena
FUNCTION MoveToAreaOfCenter(arenaWidth, arenaHeight)
    PRINT "Menuju tengah!"
    IF X < arenaWidth/2 THEN
        IF Y < arenaHeight/2 THEN
            TurnRight(Direction - 45)
        ELSE
            TurnRight(Direction + 45)
        ENDIF
    ELSE
        IF Y < arenaHeight/2 THEN
            TurnRight(Direction + 225)
        ELSE
            TurnRight(Direction + 135)
        ENDIF
    ENDIF
    SET distance ← DistanceTo(arenaWidth / 2, arenaHeight / 2)
    SET MaxSpeed ← 10
    Forward(distance)
END FUNCTION

// Event: Jika menabrak dinding
FUNCTION OnHitWall(event)
    PRINT "Nabrak dinding!"
    TurnRight(75)
    Forward(400)
    CALL MoveToAreaOfCenter(ArenaWidth, ArenaHeight)
END FUNCTION

// Event: Jika bertemu bot lain
FUNCTION OnScannedBot(event)
    SET distanceWithOtherBot ← DistanceTo(event.X, event.Y)
    IF distanceWithOtherBot > 400 THEN
        IF event.Speed == 0 THEN
            PRINT "Tembak Jauh dua kali"
            Fire(1)
            Fire(1)
        ELSE
            PRINT "Tembak Jauh satu kali"
            Fire(1)
        ENDIF
    ELSE IF distanceWithOtherBot > 200 THEN
        IF event.Speed > 6 THEN
            PRINT "Tembak Jauh"
            Fire(1)
        ELSE
            PRINT "Tembak Sedang"
            Fire(2)
        ENDIF
    ELSE IF distanceWithOtherBot > 50 THEN
        IF event.Speed > 4 THEN
```

```
                PRINT "Tembak Sedang"
                Fire(2)
            ELSE
                PRINT "Tembak Dekat"
                Fire(3)
            ENDIF
        ELSE
            PRINT "Tembak Sangat Dekat"
            Fire(4)
            Fire(1)
        ENDIF
    END FUNCTION

    // Event: Jika bertabrakan dengan bot lain
    FUNCTION OnHitBot(event)
        SET bearing ← BearingTo(event.X, event.Y)
        IF bearing > -10 AND bearing < 10 THEN
            PRINT "Di depan mata ditabrak"
            Forward(25)
            Fire(3)
        ENDIF
        IF event.IsRammed THEN
            PRINT "Ditabrak duhhh sakit!"
            TurnLeft(10)
        ENDIF
    END FUNCTION

END PROGRAM
```

d. Alternative Bot 3 (BOTentahlah)

```
BEGIN BOTentahlah

    INITIALIZE enemyDistance = MAX_VALUE
    SET BodyColor = RED
    SET GunColor = BLACK
    SET RadarColor = WHITE
    SET MaxSpeed = 8

    PRINT "BOTentahlah is running!"

    WHILE bot is running DO
        TurnRadarRight(360)
        MoveLinear()

    END WHILE

    FUNCTION OnScannedBot(event e)
        SET distance = DistanceTo(e.X, e.Y)
        SET enemyDistance = distance

        IF distance < 50 THEN
            PRINT "Adjusting for ramming!"
            SET angleToEnemy = BearingTo(e.X, e.Y)
            SET turnAngle = angleToEnemy - Direction
            TurnRight(turnAngle)

            PRINT "Ramming enemy!"
            SET MaxSpeed = 4
            Forward(distance - 10)
            SET MaxSpeed = 8
        ELSE IF distance >= 50 AND distance <= 600 THEN
            SET firePower = ChooseFirePower(distance)
            SET angleToEnemy = BearingTo(e.X, e.Y)
            SET gunTurn = angleToEnemy - GunDirection
            TurnGunRight(gunTurn)
            Fire(firePower)
            PRINT "Shooting with power:", firePower
        END IF
    END FUNCTION

    FUNCTION OnHitWall(event e)
        PRINT "Hit wall! Adjusting path..."
        SET MaxSpeed = 4
        Back(50)
        TurnRight(90)
        SET MaxSpeed = 8
    END FUNCTION

    FUNCTION OnHitBot(event e)
        PRINT "Hit another bot! Adjusting..."
        Back(30)
```

```
        TurnRight(45)
    END FUNCTION

    FUNCTION MoveLinear()
        IF X < 51 OR X > ArenaWidth - 51 OR Y < 51 OR Y > ArenaHeight
- 51 THEN
            Back(100)
            TurnRight(90)
            SET MaxSpeed = 8
            PRINT "Avoid wall"
        ELSE
            SET MaxSpeed = 8
            Forward(300)
        END IF
    END FUNCTION

    FUNCTION ChooseFirePower(distance)
        IF distance >= 50 AND distance < 100 THEN
            RETURN 3
        ELSE IF distance >= 100 AND distance < 200 THEN
            RETURN 2.5
        ELSE IF distance >= 200 AND distance < 300 THEN
            RETURN 2
        ELSE IF distance >= 300 AND distance < 400 THEN
            RETURN 1.5
        ELSE IF distance >= 400 AND distance < 500 THEN
            RETURN 1
        ELSE IF distance >= 500 AND distance < 600 THEN
            RETURN 0.5
        ELSE
            RETURN 0
        END IF
    END FUNCTION

END BOTentahlah
```

4.2 Struktur Data, Fungsi, dan Prosedur pada Solusi

a. Main Bot (BOTanchan)

BOTanchan menggunakan strategi greedy dengan fokus pada target terdekat untuk memaksimalkan akurasi tembakan. Bot ini menggunakan pendekatan agresif ketika memiliki energi cukup dan beralih ke mode bertahan saat energi rendah.

Struktur data utamanya adalah enemyHistory, sebuah *dictionary* yang menyimpan data historis musuh berdasarkan ID. Setiap entri berisi informasi seperti waktu terakhir terdeteksi, koordinat (X,Y), kecepatan, dan arah gerakan musuh. Data ini digunakan untuk memprediksi pergerakan musuh dan menentukan target.

Fungsi pemilihan target (ShouldSwitchTarget) selalu memilih musuh terdekat sebagai prioritas, yang merupakan algoritma greedy dengan memilih opsi terbaik untuk bot inii. Manajemen energi dilakukan melalui CalculateBulletPower yang menyesuaikan daya tembak berdasarkan jarak dan energi sendiri. Ketika energi rendah (<30), bot beralih ke RetreatMovement dengan pola acak untuk menghindari serangan, sementara saat energi cukup, AttackMovement digunakan dengan gerakan agresif maju. Untuk menghindari terjebak di sudut, AvoidWalls mengarahkan bot ke tengah arena ketika mendekati dinding.

b. Alternative Bot 1 (PraBOTwo)

PraBOTwo mengimplementasikan strategi greedy berbeda dengan memprioritaskan musuh berenergi terendah untuk eliminasi cepat. Bot ini menggunakan orbital movement untuk mempertahankan jarak optimal dan menyesuaikan daya tembak berdasarkan energi musuh.

Struktur datanya mirip dengan BOTanchan namun menambahkan pencatatan energi musuh. Fungsi ShouldSwitchTarget selalu memilih musuh dengan energi paling rendah sebagai target, yang merupakan algoritma *greedy* yang memilih opsi yang paling menguntungkan saat ini (musuh paling lemah). Daya peluru (CalculateBulletPower) dilakukan secara dinamis: semakin tinggi energi musuh, semakin rendah daya tembak yang digunakan untuk menghemat energi, dan sebaliknya. Gerakan bot dikendalikan oleh AdjustMovement yang menggunakan orbital strategy: mempertahankan jarak 200px dari musuh kuat (>50 energi) dan mendekat untuk menghabisi musuh lemah (<50 energi). Ketika energi sangat rendah (<10), bot menghemat daya tembak hingga 1.0 untuk mempertahankan kelangsungan hidup.

c. Alternative Bot 2 (LittleBot)

LittleBot mengimplementasikan strategi Greedy yang memaksimalkan jumlah pemindaian dalam satuan waktu. Struktur data dari bot ini hanya berupa penyimpanan variabel double arenaHeight dan arenaWidth agar method tersebut tidak dipanggil berulang kali

Ada 4 method override yang digunakan pada bot ini, yaitu Run (perilaku utama dari bot yang melingkar), OnHitBot (perilaku saat bertabrakan dengan bot), OnHitWall (perilaku saat bot menabrak dinding), dan OnScannedBot (perilaku ketika bot berhasil memindai/mendeteksi musuh). Adapun tambahan satu method baru yaitu MoveToAreaOfCenter untuk menangani perintah bergerak menuju pusar arena. Prosedur dari bot ini dimulai dari pergerakan bot menuju pusat arena, lalu setelah bot memasuki area yang telah didefenisikan, bot akan bergerak melingkar. Jika bot keluar area tersebut dan menabrak dinding (disebabkan oleh perilaku bot ketika ditembak atau ditabrak),

maka bot akan diperintahkan kembali menuju area tersebut – bisa saja tidak berhasil ke area karena method MoveToAreaOfCenter hanya dipanggil satu kali disini.

d. Alternative Bot 3 (BOTentahlah)

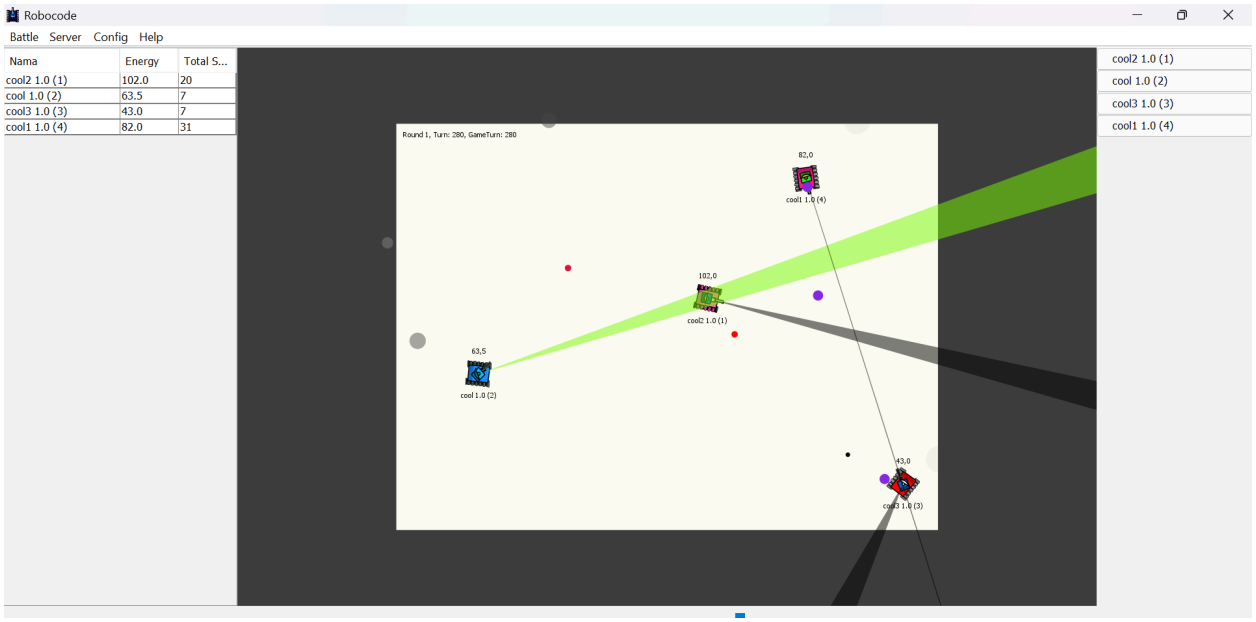
BOTentahlah menggunakan beberapa variabel global untuk menyimpan informasi penting selama pertandingan. Variabel enemyDistance digunakan untuk menyimpan jarak musuh terakhir yang terdeteksi, sehingga bot dapat menentukan apakah akan melakukan ramming atau menembak. Selain itu, variabel MaxSpeed berfungsi untuk mengontrol kecepatan bot, memungkinkan bot untuk bergerak cepat dalam kondisi normal dan memperlambat ketika melakukan manuver tertentu seperti ramming atau menghindari dinding.

Bot ini memiliki beberapa fungsi utama yang mendukung strategi pergerakan dan serangan. MoveLinear() bertanggung jawab atas pergerakan bot dalam garis lurus dengan kecepatan tinggi, sekaligus menangani logika menghindari dinding dengan mundur dan berbelok jika terlalu dekat dengan batas arena. OnScannedBot(ScannedBotEvent e) adalah fungsi utama dalam deteksi musuh, yang menentukan apakah bot harus menabrak atau menembak berdasarkan jarak musuh. Jika jarak cukup dekat, bot akan melakukan ramming dengan memperlambat dan mendekati musuh, sedangkan jika jarak lebih jauh, bot akan menyesuaikan daya tembakan dengan bantuan fungsi ChooseFirePower(double distance), yang menentukan kekuatan peluru berdasarkan jarak musuh untuk efisiensi energi.

Selain itu, terdapat fungsi OnHitWall(HitWallEvent e) dan OnHitBot(HitBotEvent e), yang menangani kondisi saat bot menabrak dinding atau bot lain. Jika bot menabrak dinding, fungsi ini akan memperlambat pergerakan, mundur, dan berbelok untuk menghindari stuck, sedangkan jika bertabrakan dengan bot lain, bot akan mundur dan mengubah arah untuk keluar dari situasi yang tidak menguntungkan. Dengan kombinasi variabel global dan fungsi ini, BOTentahlah dapat mempertahankan pergerakan cepat dan agresif sambil tetap mampu beradaptasi dengan lingkungan sekitar.

4.3 Pengujian

Dokumentasi pertandingan empat bot 1 vs 1 vs 1 vs 1



Hasil bot utama sebagai pemenang

Results for 2 rounds

Rank	Name	Total Score	Survival	Surv. Bonus	Bullet D...	Bullet Bo...	Ram Dmg.	Ram Bonus	1sts	2nds	3rds
1	cool 1.0	522	250	30	191	26	18	6	1	1	0
2	cool2 1.0	432	250	30	110	0	42	0	1	0	1
3	cool1 1.0	341	100	0	169	0	72	0	0	1	1
4	cool3 1.0	14	0	0	0	0	14	0	0	0	0

Hasil bot utama sebagai pemenang

Results for 5 rounds

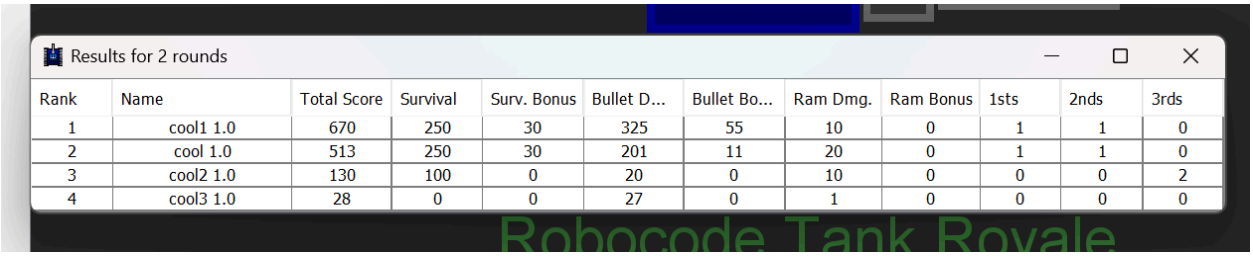
Rank	Name	Total Score	Survival	Surv. Bonus	Bullet Dmg.	Bullet Bo...	Ram Dmg.	Ram Bonus	1sts	2nds	3rds
1	cool 1.0	1153	500	60	535	34	24	0	3	1	0
2	cool2 1.0	881	450	60	232	14	92	32	1	2	1
3	cool1 1.0	360	50	0	205	11	94	0	0	1	1
4	cool3 1.0	232	200	0	27	0	5	0	0	0	2

Hasil bot alternative sebagai pemenang

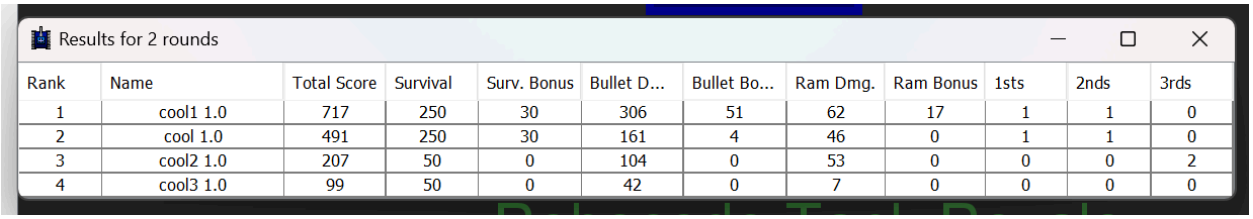
Results for 5 rounds

Rank	Name	Total Score	Survival	Surv. Bonus	Bullet Dmg.	Bullet Bo...	Ram Dmg.	Ram Bonus	1sts	2nds	3rds
1	cool1 1.0	1491	500	30	753	115	92	0	2	2	1
2	cool 1.0	1093	450	60	459	18	106	0	2	2	1
3	cool2 1.0	772	450	60	186	7	68	0	1	1	2
4	cool3 1.0	126	100	0	20	0	6	0	0	0	1

Hasil bot alternative pertama sebagai pemenang



Hasil bot alternative pertama sebagai pemenang



4.4 Analisis Hasil dari Pengujian

Berdasarkan hasil pengujian, terlihat bahwa cool1 (alterinatif bot 1) berhasil menang 2 dari 5 kali untuk pertandingan sebanyak 5 kali, sedangkan bot utama (cool) menang 2 kali dan berada di peringkat 2 sebanyak 3 kali. Dua bot alternatif lainnya cenderung berada di peringkat ketiga dan keempat walaupun cool2 sempat berada pada peringkat 2. Hal ini disebabkan bot cool dan cool2 memiliki algoritma yang cenderung lebih kuat ketika menghadapi lawan yang banyak sekaligus. Berbeda dengan bot cool1 yang dibangun dengan tujuan pertandingan 1 vs 1 (akurasi penguncian musuh untuk bot cool1 lebih baik).

Bot cool1 cenderung lebih agresif dengan sering menembakkan peluru dengan energi tinggi. Hal ini sangat efektif jika lebih banyak bot karena cenderung terkena salah satu bot. Namun, jika jumlah berkurang atau bot sangat lincah, strategi *greedy* ini kurang efektif dan cenderung boros energi sehingga terkadang kurang stabil dan bisa kalah.

Bot utama cenderung lebih berhati-hati dan banyak menghindar serta memperhitungkan jarak musuh sebelum menembak. Bot ini juga bergerak dengan cukup acak untuk menghindar dari peluru. Hal ini cukup optimal untuk bertahan hingga tersisa dua bot. Yang menjadi kekurangannya adalah ketika banyak bot, bot ini menggunakan daya yang terlalu rendah disebabkan jarak antar dua bot yang cenderung jauh karena gerakan random ini sehingga banyak bot lain yang bisa memperoleh poin lebih. Bot ini membutuhkan waktu yang lama untuk bisa menghabisi musuh, tergantung dari seberapa dekat dengan musuh, dan cenderung random kapan dekatnya.

Bot cool2 cenderung memiliki perilaku yang static di awal sehingga mudah ditebak oleh musuh. Adapun lingkup dari pergerakan bot cool2 tidak seluas bot dan sefleksibel bot lain. Kelebihan dari bot ini jika berhadapan dalam jarak yang sangat dekat. Di samping itu,

pergerakan bot yang melingkar dengan radar dan gun yang static membuat penyerangan terhadap musuh sulit jika bot musuh memiliki pergerakan yang tak terduga.

BAB 5: KESIMPULAN DAN SARAN

5.1 Kesimpulan

- Ada banyak implementasi algoritma Greedy di dunia ini. Setiap algoritma Greedy memiliki kelebihan dan kekurangan masing-masing sehingga tidak ada algoritma Greedy yang dijamin *paling mangkus untuk semua kondisi.*
- Algoritma Greedy memiliki kekurangan dalam mempertimbangkan output yang dihasilkan dalam beberapa langkah ke depan (hanya memperhitungkan keuntungan terbanyak pada kondisi saat itu juga).
- Prinsip Algoritma Greedy tergolong mudah diimplementasikan karena kesederhanaan prinsipnya yaitu “mengambil keuntungan sebanyak-banyaknya tanpa memikirkan efek samping”.

5.2 Saran

- Perlu adanya algoritma yang lebih mutakhir untuk memastikan setiap kondisi yang berbeda mendapatkan hasil yang optimal.

LAMPIRAN

Tautan repository github: https://github.com/bill2247/Tubes1_cool.git
Tautan video: youtu.be/34sCXD1AzyA?si=HwnOBiTwRMZy9oRq

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 (2025) Bag 1*. Institut Teknologi Bandung. Diambil dari https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2024-2025/04-Algoritma-Greedy-(2025)-Bag1.pdf

Documentation Robocode Api
https://robocode.sourceforge.io/docs/robocode/