LAPORAN TUGAS BESAR I IF2211 STRATEGI ALGORITMA

SEMESTER II TAHUN 2024/2025

PEMANFAATAN ALGORITMA *GREEDY* DALAM PEMBUATAN BOT PERMAINAN ROBOCODE TANK ROYALE

Oleh:

Kelompok YaMainLastWarLah

13523049 - Muhammad Fithra Rizki

13523055 - Muhammad Timur Kanigara

13523074 - Ahsan Malik Al Farisi

PROGRAM STUDI TEKNIK INFORMATIKA

SEKOLAH TEKNIK ELEKTRO DAN INFORMATIKA

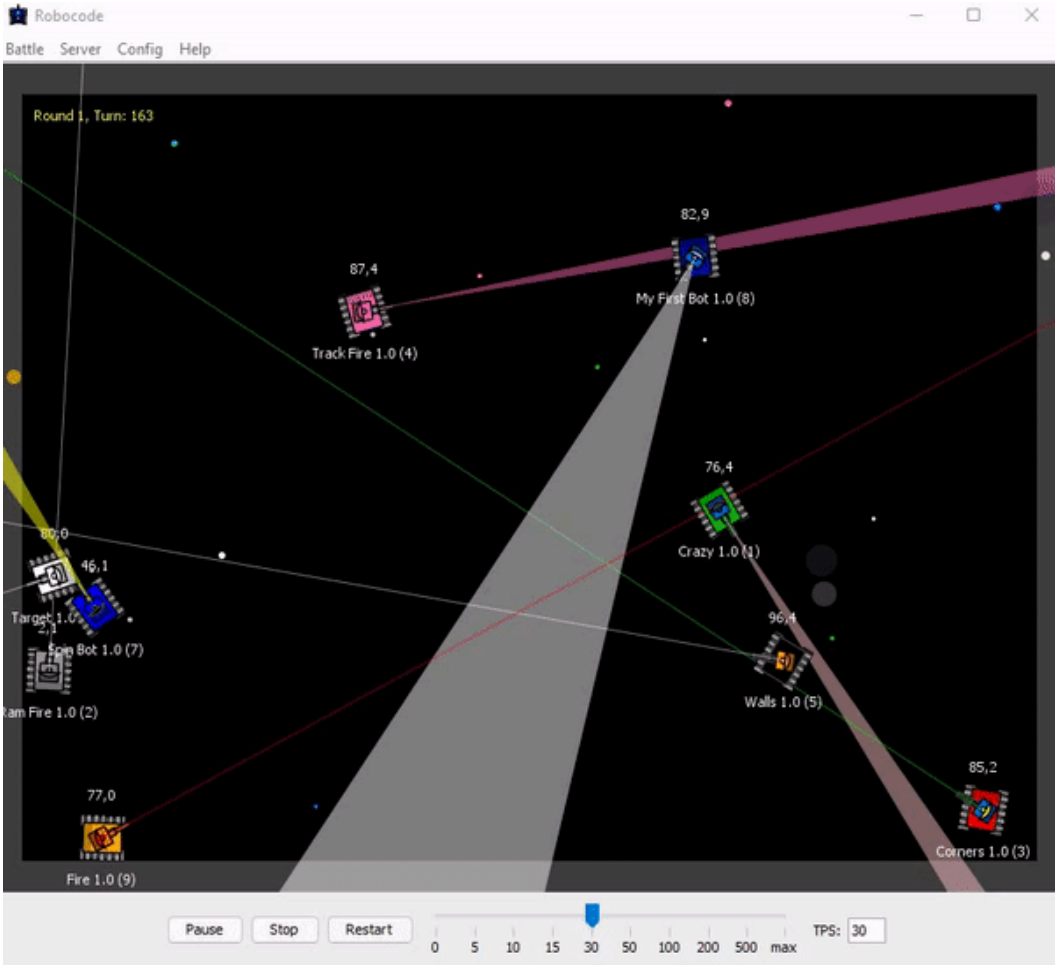
INSTITUT TEKNOLOGI BANDUNG

JL. GANESA 10, BANDUNG 40132

2024

DAFTAR ISI

BAB I
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.

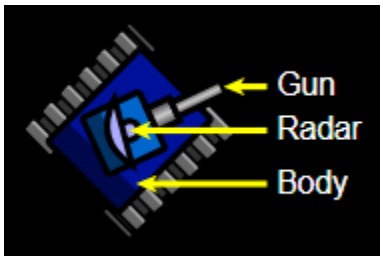
6. Tabrakan

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.

BAB II
LANDASAN TEORI

2.1 Algoritma Greedy

2.1.1 Dasar Teori Algoritma Greedy

Algoritma greedy menjadi salah satu metode sederhana yang digunakan dalam memecahkan suatu permasalahan. Algoritma ini pada dasarnya dijalankan dengan memilih keputusan terbaik pada setiap langkahnya dengan harapan nantinya akan tercapai solusi yang paling optimal dari hasil langkah-langkah tersebut tanpa memikirkan implikasi yang ditimbulkan nantinya. Pemikiran setiap langkah yang optimal pada algoritma ini juga tidak memikirkan konsekuensi yang dihasilkan pada langkah selanjutnya.

Persoalan yang dapat dioptimasi dengan algoritma greedy biasanya akan mencari antara nilai tertinggi yang biasa disebut maksimasi atau nilai terendah yang biasa disebut minimasi. Contoh dari minimasi adalah permasalahan penukaran uang atau *coin exchange problem*. Permasalahan ini mencari susunan koin minimal dari koin yang telah disediakan agar sama dengan suatu nilai koin. Contoh dari maksimasi adalah persoalan memilih aktivitas atau *activity selection problem*. Persoalan ini mencari bagaimana cara memilih sebanyak mungkin aktivitas yang telah disediakan sesuai jadwalnya masing-masing.

Algoritma greedy dalam memecahkan permasalahan tentu akan menghasilkan solusi yang bersifat optimum global. Namun, solusi ini bukanlah solusi yang terbaik atau solusi yang paling optimum, bisa jadi merupakan solusi *sub-optimum* atau *pseudo-optimum*. Alasannya adalah algoritma greedy tidak beroperasi secara menyeluruh terhadap semua kemungkinan solusi yang ada. Selain itu, terdapat beberapa fungsi seleksi yang berbeda, sehingga kita harus memilih fungsi yang tepat jika kita ingin algoritma menghasilkan solusi optimal. Jadi, pada sebagian persoalan, algoritma greedy tidak selalu berhasil memberikan solusi yang optimal, namun sub-optimal.

2.1.2 Elemen pada Algoritma Greedy

Algoritma greedy memiliki beberapa elemen, antara lain:

1. Himpunan Kandidat (C)
   Himpunan ini berisi kandidat yang akan dipilih pada setiap langkah, seperti simpul/sisi di dalam graf, job, task, koin, benda, karakter, dan lainnya
2. Himpunan Solusi (S)
   Himpunan ini berisi kandidat yang sudah dipilih
3. Fungsi Solusi
   Fungsi ini menentukan apakah himpunan kandidat yang sudah dipilih memiliki solusi terhadap permasalahan
4. Fungsi Seleksi atau *selection function*

Fungsi ini memilih kandidat berdasarkan strategi greedy tertentu. Strategi greedy ini bersifat heuristik

5. Fungsi Kelayakan atau *feasible*
   Fungsi ini memeriksa apakah kandidat yang dipilih dapat dimasukkan ke dalam himpunan solusi (layak atau tidak)
6. Fungsi Objektif
   Fungsi ini bersifat memaksimumkan atau meminimumkan

2.2 Cara Kerja Program

2.2.1 API Bot

Bot pada permainan Robocode ini memiliki beberapa dasar API yang digunakan untuk membuat bot, mengatur perilaku bot, dan juga aksi-aksi yang dapat dilakukan. API terkait bot yang terdiri dari Classes dan Interface, dengan Classes merupakan implementasi dari Interface yang ada. Interface terdiri dari BotInfo.IBuilder untuk mengisi informasi sekitar bot yang dibikin, Droid untuk Bot yang hanya menerima perintah dan tidak dipakai dalam tugas kali ini, IBaseBot untuk implementasi interface dasar yang dipakai untuk seluruh bot, dan terakhir adalah IBot yang merupakan interface lanjutan dari IBaseBot yang mencakup fungsionalitas yang lebih kompleks. Interface ini kemudian akan diimplementasi untuk Classes yang ada. Terdapat banyak Classes yang tersedia dengan fungsinya masing-masing, namun dua Class yang penting adalah Class BaseBot dan Bot untuk membangun bot baru.

2.2.2 Cara Kerja Bot

Dalam membangun bot diperlukan inisialisasi bot dengan mengimplementasikan class bot (bisa dari Bot atau BaseBot) yang sudah ada untuk kemudian membaca file bot info berbentuk json dan untuk memulai bot dengan cara memanggil method Start() di bot pada fungsi Main(). Setelah itu perilaku bot dapat diatur dalam fungsi void run() dan juga berbagai method lainnya yang dimiliki oleh bot. Pengaturan sikap bot dilakukan melalui API yang tersedia, seperti Forward(), DistanceTo(), dan masih banyak lagi. Secara umum dalam memberikan perintah kepada bot terdapat dua jenis, yaitu blocking method dengan langsung memanggil API seperti Forward() yang akan melaksanakan fungsi turn tersebut hingga selesai dan terdapat non blocking method yang diawali dengan set berguna untuk melakukan pergerakan bot secara paralel karena setiap pemanggilan method tersebut dapat digabungkan atau di-overwrite oleh API lain yang dianggap lebih penting.

2.2.3 Implementasi Greedy Pada Bot

Strategi Greedy dalam Bot dapat diterapkan dengan menganalisis faktor bagaimana cara memperoleh skor, apa saja yang dapat memengaruhi perolehan skor, faktor apa saja yang menghambat perolehan skor, dan prioritas pengambilan pilihan dalam memaksimalkan perolehan skor. Beberapa cara untuk mengimplementasikannya strategi greedy dalam bot yaitu,

menghindari bahaya secara langsung misalnya ketika terdapat banyak musuh maka bot kita akan bergerak lebih banyak untuk menghindari tembakan yang diarahkan ke kita, cara lain misalnya ketika kita kita terkena peluru maka kita akan bereaksi dengan berpindah posisi agar peluru selanjutnya yang datang tidak akan kena lagi. Contoh lain dalam mengimplementasikan Strategi Greedy, yaitu menargetkan musuh pertama yang terdeteksi, sehingga kita dapat langsung mencari poin dengan menembakkan peluru atau menabrakkan bot kita agar musuh tereliminasi. Selain dari itu terdapat banyak sekali cara untuk mengimplementasikan greedy pada bot, karena terdapat banyak sekali faktor yang mempengaruhi perolehan poin suatu bot, seperti cara bergerak bot, cara mendeteksi musuh yang ada, cara menembakkan peluru, jumlah energi yang dipakai untuk menembakkan peluru, cara melawan strategi musuh, dan masih banyak lagi.

2.2.4 Menjalankan Bot

Untuk menjalankan bot dalam permainan Robocode, perlu dilakukan beberapa langkah berikut ini:

1. Persiapan *Environment*
    a. Instalasi .NET SDK (karena bot ditulis dalam bahasa C#)
    b. Mengunduh dan menginstal game engine Robocode Tank Royale
    c. Memastikan semua dependensi yang dibutuhkan telah terpasang
2. Menjalankan Bot
    a. Buka terminal atau command prompt
    b. Masuk ke direktori tempat bot disimpan
    c. Gunakan perintah dotnet build untuk membangun program
    d. Gunakan perintah dotnet run untuk menjalankan bot dan menghubungkan ke game engine
3. Memasukkan Bot ke Pertandingan
    a. Buka Robocode Tank Royale dengan menjalankan perintah java -jar robocode-tankroyale-gui-0.30.0.jar di direktori terkait
    b. Jalankan server atau langsung menggunakan *shortcut* ctrl+S
    c. Pilih menu battle lalu *Boot* bot yang ingin dimasukkan
    d. Pilih bot yang sudah di-*boot* untuk menjalankan pertandingan

BAB III

APLIKASI STRATEGI *GREEDY*

3.1 Mapping dengan Algoritma *Greedy*

Berdasarkan elemen-elemen pada algoritma *greedy*, permasalahan ini dapat diklasifikasikan sesuai dengan elemen nya masing-masing sebagai berikut.

1. Himpunan Kandidat
   Himpunan ini berisi aksi yang dapat dilakukan oleh bot, seperti bergerak maju atau mundur, berputar sesuai derajat yang diatur, memutar radar/gun, menembak, dan menghindar
2. Himpunan Solusi
   Aksi-aksi yang dipilih sesuai dengan permasalahan pada kondisi permainan, seperti posisi musuh, tembok, dan energi
3. Fungsi Solusi
   Evaluasi dari solusi-solusi yang ada, seperti lama *survival time*, jumlah tembakan yang sukses, jumlah bot yang dikalahkan, besar *damage* yang diberikan
4. Fungsi Seleksi
   Aksi yang dipilih bot berdasarkan prioritasnya, yaitu saat terlihat musuh fokus bot menjadi mengunci musuh sambil menembakinya, saat belum mendapat musuh di trackingan bergerak acak sambil mencari musuh, dan saat menabrak musuh atau tembok bergerak menjauh dan menghindar.
5. Fungsi Kelayakan
   Bot mengecek kelayakan suatu aksi, seperti menabrak tembok, energi yang cukup untuk menembak, sudut tembakan mendekati musuh, dan mengatur target yang diserang
6. Fungsi Objektif
   Memaksimalkan skor pertandingan dengan waktu bertahan selama mungkin, mengeliminasi musuh, dan menghindari kerusakan

3.2 Deskripsi dan Analisis Strategi Greedy pada Tank

Kami membuat 4 alternatif strategi pada robot tank dengan metode heuristik nya masing-masing.

3.2.1 Strategi Greedy pada Tank “AEZAKMI”

3.2.1.1 Deskripsi Strategi

Heuristik utama dalam algoritma bot “AEZAKMI” adalah jarak dari musuh terakhir yang terdeteksi. Saat memilih arah gerakan, bot akan menghitung jarak antara posisi barunya dengan musuh terakhir. Semakin jauh jaraknya, semakin memungkinkan arah tersebut dipilih. Jika ada beberapa pilihan, bot mengambil yang memiliki jarak terjauh dari musuh tapi tetap berada dalam batas arena.
Beberapa algoritma greedy yang diterapkan dalam bot “AEZAKMI”:

a. Greedy dalam Pergerakan
   Ini adalah algoritma utama dari bot "AEZAKMI". Bot memilih arah pergerakan yang menjauhkannya sejauh mungkin dari musuh terakhir yang terdeteksi. Bot mencari berbagai kemungkinan arah pergerakan setiap 45 derajat dalam satu lingkaran penuh. Dari semua posisi yang memungkinkan, bot lalu memilih arah yang memberikan jarak terjauh dari musuh terakhir yang dipindai.
b. Greedy dalam Penembakan
   Bot menghitung posisi musuh berdasarkan kecepatan dan arah gerak musuh. Setelah mendapatkan estimasi posisi musuh, bot menyesuaikan sudut tembakan agar peluru memiliki peluang yang lebih besar untuk mengenai target. Jika posisi musuh bisa diprediksi dengan baik, bot menembak dengan energi yang optimal.

3.2.1.2 Analisis Strategi

Kelebihan Strategi yang Digunakan:

a. Perhitungan cepat : bot hanya mempertimbangkan kondisi saat ini, sehingga keputusan dapat dibuat dengan cepat tanpa menyimpan data historis atau melakukan simulasi kompleks.
b. Selalu bergerak menjauhi musuh : Dengan memiliki arah yang menjauhi musuh terakhir, bot bisa menghindari serangan langsung dengan lebih baik.
c. Tembakan prediktif yang efektif : Dengan memperkirakan posisi musuh di masa depan, bot meningkatkan peluang serangan untuk mengenai target dibandingkan sekadar menembak ke posisi saat ini.

Kekurangan Strategi yang Digunakan:

a. Tidak menyesuaikan diri dengan pola musuh : Jika musuh memiliki pola gerakan tertentu, bot tidak dapat beradaptasi dan hanya akan bereaksi berdasarkan kondisi saat ini.
b. Kemungkinan terjebak di sudut arena : Jika bot terus bergerak menjauhi musuh tanpa mempertimbangkan posisi dinding, ada kemungkinan bot terjebak di sudut dan sulit keluar.

3.2.2 Strategi Greedy pada Tank "Aedes Aegypti"

3.2.2.1 Deskripsi Strategi

Heuristik utama yang ada pada tank "Aedes Aegypti" adalah pada sisi reaksi terhadap visibilitas atau terdeteksinya musuh. Pada awalnya, bot akan bergerak sangat random dan acak untuk menghindari adanya peluru "liar" yang tiba-tiba mengenai bot. Namun, dalam pergerakannya, radar sekaligus gun pada bot akan terus berputar untuk melakukan scanning terhadap musuh di sekitar. Jika sudah ditemukan target, pergerakan bot berhenti menjadi random dan mulai untuk mengunci musuh yang menjadi target serta langsung melakukan penembakan sesuai dengan kondisi energi bot. Keputusan-keputusan ini dilangsungkan sesuai situasi permainan pada saat itu tanpa memikirkan efek jangka panjang dari perilakunya.
Berikut beberapa algoritma greedy yang diterapkan pada bot "Aedes Aegypti":

a. Greedy dalam Visibilitas Musuh
   Pada hal ini, keputusan greedy paling mendasar ditentukan. Bot akan memutuskan untuk bergerak acak ketika tidak ada musuh atau menyerang musuh yang terlihat. Ketika tidak terlihat musuh dari beberapa langkah yang sudah ditentukan (*lostTargetTurns*), bot akan bergerak menjadi acak untuk eksplorasi. Namun, ketika sudah ada musuh, bot akan lanjut menyerang musuh tersebut tanpa memikirkan langkah yang ada selanjutnya
b. Greedy dalam Penguncian Radar dan Target
   Saat terdeteksinya musuh, bot akan langsung mengunci target dan menghadapkan radar serta gun pada target secara konstan. Hal ini dilakukan setelah mendapat data posisi musuh terakhir yang ter-scan, tanpa memperhitungkan kecepatan dan arah gerak musuh yang dikunci.
c. Greedy dalam Penggunaan Energi
   Penggunaan energi diharapkan melakukan yang terbaik dengan memperhitungkan energi yang dimiliki oleh bot. Jika energi pada bot lebih dari 30, bot akan memutuskan untuk menggunakan peluru yang lebih besar. Jika dibawah itu, kekuatan peluru akan menyesuaikan
d. Greedy dalam Penembakan
   Ketika musuh berada dalam sudut ±10 derajat dari arah gun, bot segera menembak tanpa menunggu perhitungan prediktif posisi musuh. Bot tidak melakukan kalkulasi lebih dalam memperhitungkan posisi masa depan lawan, melainkan langsung memanfaatkan peluang akurasi yang tinggi saat ini. Pemilihan kekuatan tembakan juga bersifat greedy, ketika energi tinggi, bot langsung menggunakan tembakan kuat, sedangkan saat energi rendah, bot menyesuaikan untuk tetap dapat menembak sambil mempertahankan nyawa.

3.2.2.2 Analisis Strategi

Kelebihan strategi yang digunakan:

a. Responsif terhadap kondisi arena, hal ini karena memperhitungkan kehadiran musuh yang ada disekitar
b. Efisiensi waktu eksekusi, keputusan yang diambil bisa lebih cepat karena keringanan komputasi
c. Hemat energi berdasarkan situasi, penyesuaian energi yang digunakan untuk kekuatan
d. Fleksibel terhadap jumlah musuh, menyesuaikan pergerakan berdasarkan banyak-sedikitnya musuh

Kekurangan strategi yang digunakan:

a. Rentan terhadap bot yang prediktif, kesulitan menyerang pada bot yang membutuhkan prediksi penembakan yang tinggi
b. Eksplorasi dengan pergerakan acak yang kurang efisien, kurang mempertimbangkan posisi yang strategis
c. Ada tembakan yang masih terbuang, tembakan dilakukan segera ketika musuh berada di sudut ±10 derajat, tanpa mempertimbangkan kecepatan musuh atau estimasi posisi.

3.2.3 Strategi Greedy pada Tank "Dwifungsi"

3.2.3.1 Deskripsi Strategi

Heuristik yang dipakai dalam algoritma bot "Dwifungsi" yaitu menentukan kekuatan tembakkan berdasarkan jarak musuh agar efisien dalam menggunakan energi, memiliki gerakan yang dinamis dalam mengikuti musuh agar tidak gampang terkena peluru, respon cepat terhadap tabrakan, tembakan agar meminimalisir pengurangan energi, dan menabrak bot yang ditargetkan dengan harapan dapat menambahkan poin dan dapat menembakkan senjatanya dengan kekuatan maksimal agar dapat memaksimalkan poin.

a. Greedy dalam penargetan
   Bot akan langsung menarget musuh dengan cara menargetkan bot pertama yang dilihat, jika lepas dari pandangannya maka akan langsung menargetkan bot lain yang dilihat lagi. Ketika sudah menargetkan satu bot, maka Dwifungsi akan berusaha mendekati bot tersebut dan menabraknya.
b. Greedy dalam menyerang
   Bot akan berusaha untuk mengikuti dan mendekat untuk memberikan damage yang maksimum, sedangkan ketika jauh bot akan memberikan tembakan yang lebih kecil untuk menghemat energi dan juga agar tembakan peluru lebih cepat untuk mengenai musuh.
c. Greedy ketika diserang
   Ketika bot terkena tembakan maka akan langsung bereaksi dengan membelokan arahnya ke kanan sebesar 60 derajat.

3.2.3.2 Analisis Strategi

Kelebihan strategi yang digunakan:

a. Efisiensi dalam pergerakan dan serangan.
b. Serangan agresif dan menabrak musuh yang dapat menjadi efektif ketika musuh hanya sedikit.
c. Respon cepat terhadap serangan dari musuh.

Kekurangan strategi yang digunakan:

a. Kurang adaptif terhadap berbagai strategi yang dimiliki musuh karena hanya memperhatikan tembok dan jumlah musuh.
b. Tidak memprediksi gerakan musuh.
c. Dapat menjadi target empuk ketika berhenti karena menempel di bot lawan.

3.2.4 Strategi Greedy pada Tank "Mayor Teddy"

3.2.4.1 Deskripsi Strategi

Heuristik yang dipakai dalam algoritma bot "Mayor Teddy" adalah penghindaran dinding, karena algoritma utamanya adalah untuk selalu menghindar dari peluru musuh maka

diharapkan dengan menghindari dinding bot ini dapat mengurangi kemungkinan energinya berkurang secara sia-sia karena menabrak tembok, selain itu bot ini juga memutarkan senjatanya secara konstan dan langsung menembakkan senjatanya ketika mendeteksi musuh. Dengan demikian diharapkan bahwa bot ini akan susah ditembak namun di saat yang bersamaan dapat langsung menyerang musuh dan menambahkan poin sebanyak mungkin.

3.2.4.2 Analisis Strategi

Kelebihan strategi yang digunakan:

a. Pergerakan acak untuk menghindari peluru musuh dan agar susah diprediksi.
b. Serangan yang cepat ketika mendeteksi musuh.
c. Bersifat defensif

Kekurangan strategi yang digunakan:

a. Pergerakan acak bisa malah membawa bot ke posisi yang lebih merugikan.
b. Dapat menjadi target yang empuk untuk musuh yang agresif jika terdapat musuh yang banyak.

3.3 Analisis Efisiensi dan Efektivitas

3.3.1 Analisis Tank "Dwifungsi"

Pada pertandingan *one-by-one*, Tank "Dwifungsi" menang:

Results for 10 rounds

Rank	Name	Total Score	Survival	Surv. Bonus	Bullet Dmg.	Bullet Bo...	Ram Dmg.	Ram Bonus	1sts	2nds	3rds
1	Dwifungsi 1.0	1993	450	90	852	124	383	94	9	1	0
2	Spin Bot 1.0	475	50	10	288	18	108	0	1	9	0

Pada pertandingan melawan tiga bot sampel, Tank "Dwifungsi" memang:

Results for 10 rounds

Rank	Name	Total Score	Survival	Surv. Bonus	Bullet Dmg.	Bullet Bo...	Ram Dmg.	Ram Bonus	1sts	2nds	3rds
1	Dwifungsi 1.0	4320	1150	180	1720	335	829	106	8	1	1
2	Velocity Bot 1.0	1487	600	60	536	41	250	0	2	1	5
3	Spin Bot 1.0	1434	550	30	656	48	133	17	0	5	1
4	Track Fire 1.0	971	550	0	396	24	0	0	0	3	3

Analisis:
Dwifungsi bisa memang karena strateginya yaitu mengikuti musuh yang ada. Selain itu, bot ini tidak hanya mengandalkan penembakan untuk mendapatkan poin tapi juga dengan cara menabrak musuh. Hal ini bisa membuat banyak senjata yang bisa digunakan oleh Dwifungsi sehingga lebih memungkinkan untuk memenangkan pertandingan.

3.3.2 Analisis Tank "Mayor Teddy"

Pada pertandingan *one-by-one*, Tank "Mayor Teddy" kalah:

Results for 10 rounds

Rank	Name	Total Score	Survival	Surv. Bonus	Bullet Dmg.	Bullet Bo...	Ram Dmg.	Ram Bonus	1sts	2nds	3rds
1	Spin Bot 1.0	483	200	40	192	3	48	0	5	1	0
2	Mayor Teddy 1.0	259	50	10	170	14	14	0	1	5	0

Pada pertandingan melawan tiga bot sampel, Tank “Mayor Teddy” mendapatkan posisi kedua:

Results for 10 rounds

Rank	Name	Total Score	Survival	Surv. Bonus	Bullet Dmg.	Bullet Bo...	Ram Dmg.	Ram Bonus	1sts	2nds	3rds
1	Spin Bot 1.0	1619	700	120	704	70	24	0	5	0	1
2	Mayor Teddy 1.0	1185	500	0	620	48	17	0	0	6	0
3	Velocity Bot 1.0	748	450	30	260	4	4	0	1	0	5
4	Track Fire 1.0	204	0	0	204	0	0	0	0	0	0

Analisis: Pada pertandingan *one-by-one* melawan Spin Bot, Mayor Teddy kalah karena strategi nya yang mirip dengan SpinBot, tetapi radar dan moncongnya selalu berputar, tidak hanya badannya saja. Selain itu, saat melawan musuh yang banyak, Mayor Teddy tidak diuntungkan karena posisinya yang tetap sama sehingga mudah ditembak terutama oleh bot yang memiliki fitur *locking*.

3.3.3 Analisis Tank “Aedes Aegypti”

Pada pertandingan *one-by-one*, Tank “Aedes Aegypti” menang:

Results for 10 rounds

Rank	Name	Total Score	Survival	Surv. Bonus	Bullet Dmg.	Bullet Bo...	Ram Dmg.	Ram Bonus	1sts	2nds	3rds
1	Aedes Aegypti 1.0	638	200	40	339	43	16	0	5	1	0
2	Spin Bot 1.0	236	50	10	144	13	18	0	1	5	0

Pada pertandingan melawan tiga bot sampel, Tank “Aedes Aegypti” memang:

Results for 10 rounds

Rank	Name	Total Score	Survival	Surv. Bonus	Bullet Dmg.	Bullet Bo...	Ram Dmg.	Ram Bonus	1sts	2nds	3rds
1	Aedes Aegypti 1.0	2093	800	150	1001	111	30	0	6	0	0
2	Spin Bot 1.0	946	400	0	496	32	18	0	0	5	1
3	Velocity Bot 1.0	569	400	0	140	6	22	0	0	1	4
4	Track Fire 1.0	180	50	0	130	0	0	0	0	0	1

Analisis:

Dalam dua jenis pertandingan tersebut, tank “Aedes Aegypti” memiliki keunggulan dari strategi greedy nya. Ketika *one-by-one*, tank “Aedes Aegypti” akan dengan mudah mengalahkan musuh dengan pergerakan yang tidak begitu kompleks karena sistem lock dan tracking dari tank ini sangat baik. Dalam permainan *battle royale*, keunggulan dari tank ini adalah karena dia memiliki strategi hit and run yang bagus, sehingga punya strategi penyerangan yang sangat baik sekaligus bertahan yang sangat baik.

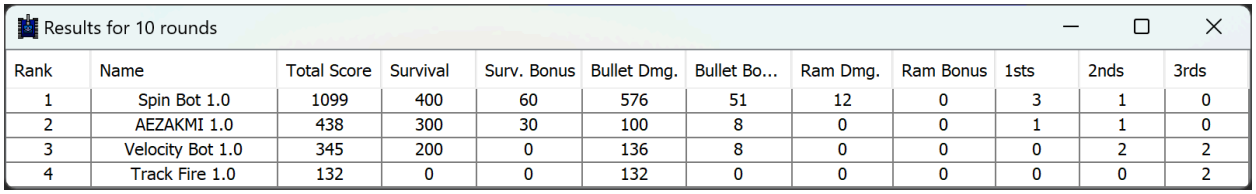
3.3.4 Analisis Tank “AEZAKMI”

Pada pertandingan *one-by-one*, Tank “AEZAKMI” kalah:

Results for 10 rounds

Rank	Name	Total Score	Survival	Surv. Bonus	Bullet Dmg.	Bullet Bo...	Ram Dmg.	Ram Bonus	1sts	2nds	3rds
1	Spin Bot 1.0	956	300	60	496	38	17	44	7	0	0
2	AEZAKMI 1.0	191	0	0	180	0	11	0	0	7	0

Pada pertandingan melawan tiga bot sampel, Tank “AEZAKMI” mendapatkan posisi kedua:



Rank	Name	Total Score	Survival	Surv. Bonus	Bullet Dmg.	Bullet Bo...	Ram Dmg.	Ram Bonus	1sts	2nds	3rds
1	Spin Bot 1.0	1099	400	60	576	51	12	0	3	1	0
2	AEZAKMI 1.0	438	300	30	100	8	0	0	1	1	0
3	Velocity Bot 1.0	345	200	0	136	8	0	0	0	2	2
4	Track Fire 1.0	132	0	0	132	0	0	0	0	0	2

Analisis:
AEZAKMI memiliki algoritma untuk menjauhkan diri dari musuh, sehingga lebih memungkinkan untuk mencapai akhir dari tiap ronde. Bot akan menjauh dari musuh yang terdeteksi sambil menembakkan peluru dengan kekuatan konstan. Selain itu, bot akan selalu melakukan pemindaian sehingga dapat melihat keseluruhan situasi di arena dan bertindak sesuai musuh yang ada.

3.3.5 Analisis Keseluruhan

3.4 Strategi Greedy Terpilih

Strategi greedy yang kami pilih dari keempat opsi alternatif bot yang telah dibuat adalah tank bernama “Aedes Aegypti”. Kami telah melakukan pertandingan sebanyak tujuh kali dengan menandingkan seluruh alternatif bot yang kami buat dan mendapatkan hasil kemenangan terbanyak dari bot “Aedes Aegypti”. Tank yang kami pilih memiliki metode penyerangan yang sangat efektif untuk melawan banyak musuh sehingga bisa mendapatkan poin yang sangat banyak. Bot ini juga memiliki strategi bertahan yang baik. Bot ini dapat bertahan sangat lama (bahkan hingga akhir) karena pergerakannya yang sangat aktif, cepat, dan tidak monoton.

BAB IV
IMPLEMENTASI DAN PENGUJIAN

4.1 Implementasi Solusi

4.1.1 Implementasi Tank “Mayor Teddy”

Pseudocode:

```
Initialize enemies ← EnemyCount
Initialize movingForward ← true
Initialize enemyDistance ← 0
Initialize random ← Random()

function Main()
   create MayorTeddy Instance
   Start()

function Run()
   BodyColor, TurretColor, RadarColor, BulletColor, ScanColor, TracksColor, GunColor ←
White
   enemies ← EnemyCount
   movingForward ← true
   GunTurnRate ← 15
   AdjustGunForBodyTurn ← true

   while (IsRunning) do
      WaitFor(TurnCompleteCondition)

      wallMargin ← 100
      battlefieldWidth ← ArenaWidth
      battlefieldHeight ← ArenaHeight

      distanceToLeftWall ← X
      distanceToRightWall ← battlefieldWidth - X
      distanceToTopWall ← battlefieldHeight - Y
      distanceToBottomWall ← Y

      turnAngle ← 0

      if (distanceToLeftWall < wallMargin and CalcBearing(180) < 60) then
         turnAngle ← 90
      else if (distanceToRightWall < wallMargin and CalcBearing(0) < 60) then
         turnAngle ← -90
      else if (distanceToTopWall < wallMargin and CalcBearing(90) < 60) then
         turnAngle ← 90
```

```
        else if (distanceToBottomWall < wallMargin and CalcBearing(270) < 60) then
            turnAngle ← -90
        else
            turnAngle ← random(-180, 180)

        SetForward(100)
        SetTurnRight(turnAngle)

function OnScannedBot(event e)
    enemyX ← e.X
    enemyY ← e.Y
    SetFire(3)
    WaitFor(TurnCompleteCondition)

function OnHitWall(event e)
    if (movingForward = true) then
        Back(100)
        TurnRight(90)
        movingForward ← false
    else
        Forward(100)
        TurnRight(-90)
        movingForward ← true
    WaitFor(TurnCompleteCondition)

Class TurnCompleteCondition Extends Condition
    PRIVATE bot

    function TurnCompleteCondition(bot)
        this.bot ← bot

    function Test()
        → bot.TurnRemaining = 0
```

4.1.2 Implementasi Tank "Dwifungsi"

Pseudocode:

```
Initialize random ← Random()

function Main()
    create Dwifungsi Instance
    Start()

function Run()
    GunColor, TurretColor, ScanColor, BulletColor, TracksColor ← Aquamarine
```

```
    BodyColor, RadarColor ← White

    while (IsRunning) do
        SetForward(100)
        SetTurnLeft(10)
        SetTurnGunLeft(10)
        Go()

function OnScannedBot(event e)
    Interruptible ← true
    turnDirection ← BearingTo(e.X, e.Y)
    bearingFromGun ← GunBearingTo(e.X, e.Y)

    output("Turning:", e.Direction)
    SetTurnGunLeft(bearingFromGun / 2)
    SetTurnLeft(turnDirection)
    output("Turn direction:", turnDirection, "Current Direction:", Direction)
    SetForward(100)
    output(“Forward")
    SetFire(FiringPower(DistanceTo(e.X, e.Y)))

    if (bearingFromGun = 0) then
            Rescan()

function OnHitBot(event e)
    if (e.IsRammed) then
        Fire(3)
    else
        Back(50)
        TurnLeft(90 + random(45))

function OnHitByBullet(event e)
    TurnRight(60)

function OnHitWall(event e)
    Back(50 + random(30))
    TurnLeft(90 + random(45))

function FiringPower(distance)
    if (distance > 200) then
        → 1
    else if (distance > 100) then
        → 2
    else
        → 3
```

```
Class TurnCompleteCondition Extends Condition
   PRIVATE bot

   function TurnCompleteCondition(bot)
      this.bot ← bot

   function Test()
      → bot.TurnRemaining = 0
```

4.1.3 Implementasi Tank "Aedes Aegypti"

Pseudocode:

```
Initialize rng ← Random()
Initialize targetVisible ← false
Initialize turnsSinceLastScan ← 0
Initialize moveCooldown ← 0

function MaxLostTurns()
   if (EnemyCount < 4) then // Jika musuh < 4, lebih lock target
      → 40
   else
      → 15

function Run()
   SET BodyColor ← White
   SET TurretColor ← Black
   SET RadarColor ← Red
   SET BulletColor ← Red
   SET ScanColor ← Red
   SET TracksColor ← Black
   SET GunColor ← Red

function OnTick(event e)
   if (targetVisible = false) then
      if (moveCooldown <= 0) then
         SetForward(random(100, 300))
         SetTurnRight(random(-90, 90))
         moveCooldown ← random(20, 40)
      moveCooldown ← moveCooldown - 1
      SetTurnGunRight(45)  // Scan sambil bergerak
   else
      // Bot sedang mengunci musuh
      SetForward(20)
      SetTurnRight(5)
      SetTurnGunRight(20)  // Terus track musuh
```

```
        turnsSinceLastScan ← turnsSinceLastScan + 1
        if (turnsSinceLastScan > MaxLostTurns()) then
            targetVisible ← false
            turnsSinceLastScan ← 0

function OnScannedBot(event e)
    targetVisible ← true
    turnsSinceLastScan ← 0

    bearingFromGun ← GunBearingTo(e.X, e.Y)
    SetTurnGunLeft(bearingFromGun)

    // Algoritma menembak
    if (abs(bearingFromGun) <= 10 and GunHeat = 0) then
        distance ← DistanceBetween(e.X, e.Y)
        if (Energy > 30) then
            power ← 2.5
        else
            power ← 1.0
        power ← min(3.0, power)
        Fire(power)

    if (abs(bearingFromGun) < 0.01) then
        Rescan()

function OnHitWall(event e)
    Back(50)
    TurnLeft(random(90, 180))

function OnHitBot(event e)
    Back(40)
    Fire(3)

function OnHitByBullet(event e)
    TurnLeft(random(-90, 90))
    Forward(random(40, 100))

// Mengukur jarak musuh
function DistanceBetween(x, y)
    dx ← x - X
    dy ← y - Y
    → sqrt(dx² + dy²)
```

4.1.4 Implementasi Tank "AEZAKMI"

Pseudocode:

```
Initialize battlefieldWidth ← ArenaWidth
Initialize battlefieldHeight ← ArenaHeight
Initialize lastScannedX, lastScannedY ← 0
Initialize SafeMargin ← 100

function Run()
   BodyColor ← DarkBlue
   battlefieldWidth ← ArenaWidth
   battlefieldHeight ← ArenaHeight

   while IsRunning do
      MoveGreedy()

// Pergerakan untuk menghindari musuh
function MoveGreedy()
   bestAngle ← 0
   maxDistance ← 0

   for i from 0 to 360 step 45 do
      newX ← X + cos(DegreesToRadians(i)) * 100
      newY ← Y + sin(DegreesToRadians(i)) * 100

      if IsSafePosition(newX, newY) then
         distance ← DistanceTo(lastScannedX, lastScannedY)
         if distance > maxDistance then
            maxDistance ← distance
            bestAngle ← i

   SetTurnRight(bestAngle)
   Forward(100)

function IsSafePosition(x, y)
   → (x > SafeMargin and x < battlefieldWidth - SafeMargin) and (y > SafeMargin and y <
battlefieldHeight - SafeMargin)

//Fungsi saat musuh terpindai
function OnScannedBot(event e)
   lastScannedX ← e.X
   lastScannedY ← e.Y
   LockRadar(e)
   PredictiveFire(e)

function LockRadar(event e)
```

```
    angleToEnemy ← BearingTo(e.X, e.Y)
    radarTurn ← NormalizeBearing(angleToEnemy - GunDirection)
    SetTurnGunRight(radarTurn)

function PredictiveFire(event e)
    bulletSpeed ← 20 - (3 * 2)  // Power 2 bullet
    distance ← DistanceTo(e.X, e.Y)
    time ← distance / bulletSpeed
    futureX ← e.X + cos(DegreesToRadians(e.Direction)) * e.Speed * time
    futureY ← e.Y + sin(DegreesToRadians(e.Direction)) * e.Speed * time
    fireAngle ← BearingTo(futureX, futureY)

    SetTurnGunRight(NormalizeBearing(fireAngle - GunDirection))
    Fire(2)

function DistanceTo(x, y)
    → sqrt((x - X)² + (y - Y)²)

function BearingTo(x, y)
    → atan2(y - Y, x - X) * (180 / π)

function NormalizeBearing(angle)
    while (angle > 180) do
        angle ← angle - 360
    while (angle < -180) do
        angle ← angle + 360
    → angle

function OnHitWall(event e)
    Back(50)
    TurnRight(random(90, 180))
```

4.2 Pengujian

4.2.1 Hasil Pengujian

Dilakukan percobaan sebanyak 7 kali pertandingan, hasil pertandingannya sebagai berikut:

Percobaan 1:

Results for 10 rounds

Rank	Name	Total Score	Survival	Surv. Bonus	Bullet Dmg.	Bullet Bo...	Ram Dmg.	Ram Bonus	1sts	2nds	3rds
1	Aedes Aegypti 1.0	1246	500	90	536	66	54	0	4	0	1
2	Mayor Teddy 1.0	1120	400	60	560	73	12	15	2	0	2
3	Dwifungsi 1.0	973	350	0	402	37	139	45	0	5	0
4	AEZAKMI 1.0	395	250	0	120	0	25	0	0	1	3

Percobaan 2:

Results for 10 rounds

Rank	Name	Total Score	Survival	Surv. Bonus	Bullet Dmg.	Bullet Bo...	Ram Dmg.	Ram Bonus	1sts	2nds	3rds
1	Dwifungsi 1.0	1388	350	0	546	57	319	115	4	2	0
2	Aedes Aegypti 1.0	1278	650	60	480	41	47	0	2	2	2
3	AEZAKMI 1.0	899	600	90	110	8	91	0	0	2	4
4	Mayor Teddy 1.0	234	50	0	144	0	40	0	0	0	0

Percobaan 3:

Results for 10 rounds

Rank	Name	Total Score	Survival	Surv. Bonus	Bullet Dmg.	Bullet Bo...	Ram Dmg.	Ram Bonus	1sts	2nds	3rds
1	Aedes Aegypti 1.0	1241	550	60	544	55	31	0	3	2	1
2	Dwifungsi 1.0	1178	400	60	460	84	173	0	2	1	0
3	AEZAKMI 1.0	776	450	30	200	55	86	0	1	2	3
4	Mayor Teddy 1.0	485	250	0	208	12	14	0	0	1	2

Percobaan 4:

Results for 10 rounds

Rank	Name	Total Score	Survival	Surv. Bonus	Bullet Dmg.	Bullet Bo...	Ram Dmg.	Ram Bonus	1sts	2nds	3rds
1	Aedes Aegypti 1.0	1953	750	120	913	80	89	0	5	2	1
2	Dwifungsi 1.0	1850	500	30	740	77	410	93	2	5	0
3	Mayor Teddy 1.0	895	550	30	272	28	14	0	1	1	5
4	AEZAKMI 1.0	652	300	30	130	6	186	0	0	0	2

Percobaan 5:

Results for 10 rounds

Rank	Name	Total Score	Survival	Surv. Bonus	Bullet Dmg.	Bullet Bo...	Ram Dmg.	Ram Bonus	1sts	2nds	3rds
1	Aedes Aegypti 1.0	1606	750	90	683	51	31	0	3	2	2
2	Dwifungsi 1.0	1417	450	30	570	59	268	40	3	2	1
3	AEZAKMI 1.0	792	500	30	170	8	84	0	1	2	2
4	Mayor Teddy 1.0	606	250	30	288	6	26	5	0	1	2

Percobaan 6:

Results for 10 rounds

Rank	Name	Total Score	Survival	Surv. Bonus	Bullet Dmg.	Bullet Bo...	Ram Dmg.	Ram Bonus	1sts	2nds	3rds
1	Aedes Aegypti 1.0	2168	950	150	915	108	44	0	5	2	0
2	Dwifungsi 1.0	1278	400	0	556	64	214	44	1	3	1
3	Mayor Teddy 1.0	760	450	30	256	9	14	0	1	0	5
4	AEZAKMI 1.0	553	250	0	190	0	107	6	0	2	1

Percobaan 7:

Results for 10 rounds

Rank	Name	Total Score	Survival	Surv. Bonus	Bullet Dmg.	Bullet Bo...	Ram Dmg.	Ram Bonus	1sts	2nds	3rds
1	Aedes Aegypti 1.0	1660	700	120	690	113	36	0	4	2	0
2	Dwifungsi 1.0	1453	400	0	602	68	305	78	1	5	0
3	Mayor Teddy 1.0	663	300	30	272	9	31	20	2	0	2
4	AEZAKMI 1.0	563	400	30	110	0	23	0	0	0	5

Dari tujuh kali percobaan, Tank “Aedes Aegypti” memenangkan pertandingan sebanyak 6 kali, dan Tank “Dwifungsi” menang sebanyak 1 kali dengan selisih poin yang tidak terlalu jauh. Oleh karena itu, Tank “Aedes Aegypti” dipilih sebagai *Main-Bot*.

4.2.2 Analisis Hasil Pengujian

Dilakukan tujuh kali percobaan dan Tank “Aedes Aegypti” mendapatkan kemenangan terbanyak yaitu enam dari tujuh pertandingan. Ada beberapa faktor mengapa Tank “Aedes Aegypti” dapat menjadi juara :

1. Bot menggunakan pergerakan keliling dan berputar, yang mana dapat meningkatkan visibilitas dan kemungkinan untuk memindai musuh dengan baik.
2. Bot lebih hemat energi yang mana menyesuaikan situasi. Energi yang dikeluarkan disesuaikan untuk kekuatan menembak.
3. Bot lebih fleksibel terhadap jumlah musuh, yaitu menyesuaikan pergerakan berdasarkan banyak-sedikitnya musuh yang ada di arena.
4. Untuk menghadapi musuh yang banyak, bot “Aedes Aegypti” diuntungkan karena pergerakannya yang acak dan sistem *lock* untuk target sehingga dapat mengenai target dan mempertahankan energi yang dimiliki.

Walaupun Tank “Aedes Aegypti” lebih banyak meraih kemenangan, tetapi Tank “Dwifungsi” memiliki poin yang tidak berbeda jauh. Hal ini karena algoritma “Dwifungsi” yang menerapkan *following target* sehingga dapat membunuh musuh lebih cepat dan memulihkan energi. Akan tetapi, strategi ini tidak cukup efisien jika melawan banyak musuh sehingga yang menjadi juara adalah “Aedes Aegypti”.

BAB V
KESIMPULAN DAN SARAN

5.1 Kesimpulan

Penggunaan strategi greedy pada permainan *Robocode Tank Royale* terbukti dapat menjadi pendekatan yang baik melalui bagaimana kita dapat mengatur strategi-strategi yang diterapkan pada bot sehingga mendapat strategi yang optimal dalam permainan yang memiliki situasi dinamis. Perancangan bot yang ada memiliki harapan agar bot dapat mempertimbangkan situasi aktual pada permainan sehingga dapat menghasilkan keputusan terbaik pada langkah tersebut tanpa memikirkan efek jangka panjang dari pergerakan yang bot lakukan.

Dalam implementasinya, strategi greedy diterapkan di berbagai macam situasi, antara lain greedy digunakan untuk pemilihan arah yang dapat menghindari musuh atau dinding, penggunaan greedy untuk penguncian target musuh berdasarkan radar, pemilihan kekuatan tembakan berdasarkan jarak antara bot dengan musuh, hingga penggunaan greedy pada keputusan yang berubah tergantung jumlah musuh yang tersisa. Setiap implementasi ini menunjukan bagaimana konsep dari algoritma greedy itu sendiri, terutama bagaimana menghasilkan langkah-langkah terbaik agar mendapatkan solusi yang optimal.

Dari tugas besar ini, dapat dipahami bahwa algoritma greedy yang sederhana dan efisien tentu mempunyai keterbatasan dalam pengaplikasiannya. Diantaranya adalah tidak adanya prediksi perlakuan situasi di masa depan, kurangnya pertimbangan jangka panjang, serta tidak adanya koordinasi antara langkah satu dengan langkah lainnya. Meskipun demikian, strategi greedy tetap merupakan pilihan yang baik untuk permainan seperti *Robocode Tank Royale* karena memberikan hasil yang kompetitif dengan pertimbangan kompleksitas yang relatif rendah.

5.2 Saran

Secara keseluruhan pengerjaan Tugas Besar 1 ini sudah berjalan cukup lancar, namun sedikit saran atau keluhan yaitu mungkin pembuatan program dapat menggunakan bahasa Java karena referensi API lebih banyak yang terdapat di internet dan ketika ketika ingin dicoba hasilnya menjadi cukup berbeda dengan API yang ditulis di C#.

LAMPIRAN

Repository github:
https://github.com/ahsuunn/Tubes1_YaMainLastWarLah

Link video:
https://www.youtube.com/watch?v=IEFiw2Y830E

DAFTAR PUSTAKA

Robocode Developers. 2025. *Beyond the Basics - Robocode Tank Royale Tutorial*. Diakses pada https://robocode-dev.github.io/tank-royale/tutorial/beyond-the-basics.html.

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