LAPORAN TUGAS BESAR I

IF2211 Strategi Algoritma

Pemanfaatan Algoritma Greedy dalam pembuatan bot permainan Robocode Tank Royale

Disusun oleh:

“JaMin”

Dzaky Aurelia Fawwaz 13523065

Rafif Farras 13523095

Ahmad Wicaksono 13523121

SEKOLAH TEKNIK ELEKTRO DAN INFORMATIKA

PROGRAM STUDI TEKNIK INFORMATIKA

INSTITUT TEKNOLOGI BANDUNG

2025

DAFTAR ISI

BAB I
DESKRIPSI TUGAS

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

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

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.

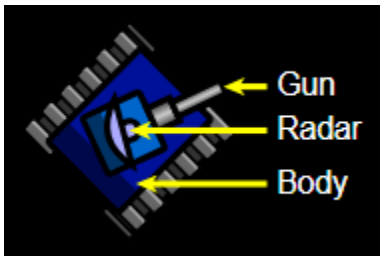
Seperti disebutkan sebelumnya, peluru yang lebih berat akan bergerak lebih lambat. Ini berarti akan membutuhkan waktu lebih lama untuk mencapai target, meningkatkan risiko peluru tidak mengenai sasaran. Sebaliknya, peluru yang lebih ringan bergerak lebih cepat, sehingga lebih mudah mengenai target, tetapi 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 Dasar Teori Algoritma Greedy

Algoritma greedy adalah metode pemecahan masalah yang dilakukan secara bertahap (step by step) dengan prinsip "ambil yang terbaik saat ini" (take what you can get now). Pada setiap langkah, algoritma ini memilih solusi terbaik yang tersedia saat itu tanpa mempertimbangkan dampaknya di masa depan, dengan harapan bahwa serangkaian keputusan optimum lokal akan menghasilkan solusi optimum global. Pendekatan ini bertujuan untuk mencapai hasil yang optimal dengan memilih langkah yang paling logis dan efisien pada setiap tahapnya.

2.2 Cara Kerja Program Algoritma Greedy pada Bot Robocode Tank Royale

2.2.1 Cara Bot Melakukan Aksi

Bot dalam Robocode Tank Royale dikembangkan menggunakan **Robocode Tank Royale API**, yang memungkinkan bot untuk berkomunikasi dengan game engine, membaca informasi lingkungan, dan mengambil keputusan berbasis algoritma greedy. API ini menyediakan berbagai event yang memungkinkan bot untuk mendeteksi lawan, menghindari tembok, dan menembak dengan strategi tertentu.

Struktur utama bot terdiri dari beberapa file penting yang digunakan dalam pengembangannya:

1. File .cs (C# Source Code)
   Berisi logika utama bot, termasuk strategi pergerakan, penembakan, dan mekanisme bertahan.
2. File .json (Konfigurasi Bot)
   Berisi metadata bot, seperti nama bot, warna, dan strategi yang diterapkan
3. File .csproj (Project Configuration File)
   Digunakan untuk mengelola dependensi proyek, termasuk versi .NET yang digunakan dan referensi API Robocode.
4. File .sh dan .cmd (Script Eksekusi Bot)
   Digunakan untuk menjalankan bot dalam Linux/MacOS (.sh) atau Windows (.cmd).

Melalui struktur file ini, bot dapat membaca input dari game engine, menentukan aksi berdasarkan kondisi permainan, dan mengambil keputusan berbasis greedy untuk memenangkan pertandingan.

2.2.2 Implementasi Algoritma Greedy ke Bot

Untuk mengimplementasikan bot dalam Robocode Tank Royale, langkah pertama adalah membuat direktori bot dan menyusun file yang diperlukan. Setiap bot harus memiliki file .json sebagai konfigurasi metadata, file .cs sebagai implementasi logika utama, serta file .csproj, .cmd, dan .sh untuk memastikan bot dapat dieksekusi dengan baik.

Bot dijalankan menggunakan API yang memungkinkan komunikasi dengan game engine, membaca kondisi permainan, dan mengambil keputusan secara greedy. Bot akan mengimplementasikan greedy dengan mengambil langkah terbaik dalam setiap kondisi permainan tanpa mempertimbangkan konsekuensi jangka panjang. Hal ini memastikan bot selalu memiliki keuntungan taktis dibandingkan lawannya.

2.2.3 Cara Menjalankan Bot

Untuk menjalankan permainan Robocode Tank Royale, ikuti langkah-langkah berikut:

1. Menjalankan Aplikasi GUI
   - Jalankan file GUI menggunakan perintah berikut di terminal:

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

   - Setelah GUI terbuka, klik tombol "Config" untuk melakukan setup awal.

2. Mengatur Direktori Bot
   - Klik tombol "Bot Root Directories".
   - Masukkan direktori tempat folder bot berada.

3. Menjalankan Battle
   - Klik tombol "Battle", lalu pilih "Start Battle".
   - Pada panel konfigurasi permainan:
     - Bot yang telah dikonfigurasi akan muncul di kotak kiri-atas.
     - Pilih bot yang ingin dimainkan dan klik "Boot →" untuk memuatnya.
     - Bot yang berhasil di-boot akan muncul di kotak kiri-bawah.
   - Tambahkan bot ke dalam permainan:
     - Pilih bot dari kotak kiri-bawah dan klik "Add →".
     - Jika ingin menambahkan semua bot, klik "Add All →".
     - Klik "Start Battle" untuk memulai permainan.

BAB III

APLIKASI STRATEGI GREEDY

3.1 Mapping

3.1.1 NabrakBelok - Greedy by Finding the Wall for Safety Place

1. Himpunan Kandidat: Semua aksi yang dapat dilakukan oleh AltBot1 dalam permainan, diantaranya bergerak maju sampai menabrak dinding/musuh, mundur dan berbelok setelah menabrak dinding/musuh, menyerang lawan berdasarkan jarak, serta menghindari musuh dengan bergerak mundur setelah tabrakan.
2. Himpunan Solusi: Langkah optimal yang diambil oleh AltBot1 untuk bertahan dan menyerang secara efisien. Bot bergerak maju dalam pola lurus, lalu berbelok setelah mencapai ujung arena. Jika menabrak dinding, bot akan mundur dan berbelok. Jika bertemu lawan, bot akan menembak dan mundur sebelum melanjutkan pergerakan.
3. Fungsi Solusi: Bot menggunakan pola pergerakan lurus panjang di arena, berbelok setelah menabrak dinding, dan menyesuaikan kekuatan tembakan berdasarkan jarak musuh. Saat bot menabrak lawan, ia akan mundur dan menyesuaikan posisi sebelum kembali bergerak.
4. Fungsi Seleksi: Pada setiap langkahnya, AltBot1 memilih aksi terbaik berdasarkan kondisi yang dihadapi. Jika tidak ada musuh atau hambatan, bot akan bergerak lurus dalam jarak panjang hingga menabrak dinding. Jika mendeteksi musuh dalam jangkauan, bot akan menyesuaikan kekuatan tembakan berdasarkan jarak dan menembak. Saat bertabrakan dengan bot lain, AltBot1 akan mundur, berbelok, lalu menembak musuh yang ada di hadapannya sebelum kembali ke pola gerakan awalnya.
5. Fungsi Kelayakan: Setiap aksi hanya dilakukan jika memungkinkan. Bot menembak hanya jika turret sejajar dan senjata tidak overheat, berbelok hanya setelah menabrak dinding atau bot lain, dan mundur hanya saat terjadi tabrakan, memastikan pergerakan tetap efisien tanpa berlebihan.
6. Fungsi Objektif: Tujuan utama AltBot1 adalah bertahan selama mungkin sambil menyerang secara efektif. Dengan terus bergerak dalam pola lurus dan berbelok setelah tabrakan, bot menghindari menjadi target statis. Serangan dilakukan hanya saat musuh dalam jangkauan optimal, memastikan energi digunakan secara efisien.

3.1.2 SeptikTank - Greedy by Target Vulnerability

1. Himpunan Kandidat : Semua kemungkinan aksi yang dapat diambil bot, seperti menabrak musuh dengan energi rendah, menembak dari jarak optimal, bergerak maju/mundur, berputar, atau menentukan kekuatan tembakan berdasarkan jarak dan energi musuh.
2. Himpunan Solusi : Rangkaian keputusan yang dipilih oleh bot untuk menyerang secara optimal berdasarkan energi musuh, seperti menabrak musuh dengan energi rendah,

menjaga jarak optimal dengan musuh energi tinggi, dan menembak dengan kekuatan yang disesuaikan dengan jarak.

3. Fungsi Solusi : Mengoptimalkan strategi serangan berdasarkan perbandingan energi dengan musuh, memaksimalkan damage dengan menabrak musuh lemah dan menembak musuh kuat dari jarak optimal.
4. Fungsi Seleksi : Fungsi yang menentukan aksi terbaik berdasarkan energi musuh. Bot memilih untuk menabrak jika energi musuh rendah, menjaga jarak aman jika energi musuh tinggi, dan menembak dengan kekuatan yang sesuai dengan jarak (tinggi untuk jarak dekat, rendah untuk jarak jauh).
5. Fungsi Kelayakan : Memastikan setiap aksi yang dipilih dapat dieksekusi dengan baik dalam kondisi saat itu. Bot hanya menembak jika memiliki cukup energi, hanya menabrak jika energi musuh rendah, dan selalu menyesuaikan arah untuk menghadap target secara tepat sebelum melakukan aksi apapun.
6. Fungsi Objektif : Tujuan utama bot ini adalah memaksimalkan efisiensi serangan dengan pendekatan berbasis energi musuh, menghabisi musuh lemah melalui tabrakan langsung dan menyerang musuh kuat dari jarak aman dengan kekuatan tembakan optimal.

3.1.3 MajuMundurLock - Greedy by Movement and Shooting

1. Himpunan Kandidat: Himpunan aksi yang dapat diambil bot berdasarkan kondisi saat ini, seperti melanjutkan gerakan maju-mundur, menghindari tembok, menembak musuh yang terdeteksi, atau berhenti menembak saat energi rendah.
2. Himpunan Solusi: Langkah-langkah yang dipilih bot untuk bertahan hidup dan menyerang secara optimal, seperti menjaga pola gerakan konstan, menghindari tabrakan sebelum terjadi, menembak hanya saat peluang kena tinggi, dan berhenti menyerang jika energi terlalu rendah.
3. Fungsi Solusi: mengoptimalkan pergerakan bot agar sulit ditembak musuh, meningkatkan efisiensi tembakan dengan mengunci target, serta memastikan bot selalu berada dalam kondisi bertahan terbaik sesuai banyak energi.
4. Fungsi Seleksi: Fungsi yang menentukan aksi terbaik yang diambil bot dalam setiap kondisi. Bot akan memilih untuk bergerak maju-mundur jika tidak ada ancaman, langsung berbelok jika hampir menabrak tembok, menembak hanya jika musuh sejajar dengan turret, dan berhenti menembak jika energi kurang dari 30.
5. Fungsi Kelayakan: Memastikan bahwa setiap aksi yang diambil bot dapat dieksekusi sesuai kondisi saat ini. Bot hanya menembak jika gun sudah mengarah ke musuh dan GunHeat = 0, hanya berbelok jika hampir menabrak dinding, dan hanya menghindar jika terkena tembakan atau ada musuh terlalu dekat.
6. Fungsi Objektif: Tujuan utama bot adalah memaksimalkan survival dengan tetap memberikan serangan efektif. Bot menggunakan gerakan konstan untuk menghindari serangan, menembak dengan presisi menggunakan metode konci target, dan menghentikan serangan saat energi rendah untuk bertahan lebih lama di arena.

3.1.4 ZigiZaga - Greedy by ZigZag Movement

1. Himpunan kandidat : Semua kemungkinan aksi yang dapat diambil bot pada setiap saat, seperti melakukan pergerakan zigzag dengan berbagai variasi sudut dan jarak, menembak musuh dengan kekuatan berbeda berdasarkan jarak, menghindar saat terdeteksi tembakan musuh, berputar menjauh dari dinding, atau berhenti dan mengubah arah gerakan.
2. Himpunan Solusi : Rangkaian keputusan yang dipilih bot untuk bertahan dan menyerang secara efektif, seperti mempertahankan pola zigzag yang terus berubah untuk sulit ditembak, menembak musuh hanya saat senjata terarah dengan baik, langsung menghindar saat terdeteksi tembakan, dan mengubah arah gerakan saat menabrak dinding atau musuh.
3. Fungsi Solusi : Mengoptimalkan pola pergerakan zigzag dinamis yang membuat bot sulit ditembak, memaksimalkan efisiensi serangan dengan memilih kekuatan tembakan yang sesuai jarak, dan memaksimalkan kemampuan menghindar dengan mendeteksi tembakan musuh melalui perubahan energi.
4. Fungsi Seleksi : Fungsi yang menentukan aksi terbaik pada setiap situasi. Bot memilih untuk bergerak dalam pola zigzag dengan sudut (30 - (tickCounter % 60)) dan jarak (150 + (tickCounter % 100)) jika tidak ada ancaman, langsung mengubah arah saat mendeteksi tembakan musuh, menembak dengan kekuatan yang divariasikan berdasarkan jarak.
5. Fungsi Kelayakan : Memastikan setiap aksi yang dipilih dapat dieksekusi dengan baik. Bot hanya menembak jika senjata sudah terarah pada musuh, hanya mengubah arah jika terdeteksi musuh menembak , dan hanya melakukan manuver penghindaran khusus saat tertembak atau terancam.
6. Fungsi objektif : Tujuan utama bot adalah memaksimalkan survival time dengan pola zigzag dinamis yang sulit diprediksi, mengoptimalkan serangan dengan pemilihan kekuatan tembak berbasis jarak, dan meningkatkan tingkat penghindaran melalui deteksi tembakan musuh dan reaksi cepat terhadap ancaman.

3.2 Eksplorasi Alternatif Solusi Greedy

Tujuan dari permainan ini adalah untuk mendapatkan poin yang sebesar besarnya dengan menembak bot musuh, bertahan sampai bot musuh mati (survival), dan menabrak bot musuh. Untuk mencapai hal-hal tersebut, dilakukan beberapa strategi greedy yang berfokus pada survival dengan pergerakan untuk menghindari tembakan musuh, penguncian penembakan pada satu musuh, mencari space aman pada arena, pergerakan zigzag, hingga menyesuaikan dengan kondisi dan kelemahan musuh. Berikut penjelasan lebih detailnya:

3.2.1 NabrakBelok - Greedy by Finding the Wall for Safety Place

Bot ini menggunakan strategi greedy berbasis pergerakan konstan dengan pola maju panjang, belok setelah tabrakan, dan serangan jarak optimal. Bot bergerak lurus hingga menabrak dinding, lalu mundur dan berbelok 135° untuk menghindari terjebak. Saat mendeteksi musuh, bot menyesuaikan kekuatan tembakan berdasarkan jarak, memastikan

serangan yang efisien. Jika bertabrakan dengan bot lain, bot akan mundur, berbelok, dan menembak sebelum kembali ke pola gerakannya. Dengan strategi ini, bot ini memaksimalkan mobilitas untuk bertahan lebih lama sambil tetap memberikan tekanan kepada lawan melalui serangan yang terukur.

3.2.2 SeptikTank - Greedy by Target Vulnerability

Algoritma ini bertujuan untuk memaksimalkan efisiensi serangan dengan selalu memilih strategi berdasarkan kelemahan musuh yang terdeteksi. Bot akan menganalisis level energi musuh dan mengambil keputusan yang berbeda berdasarkan kondisi tersebut. Untuk musuh dengan energi rendah (<10), bot akan langsung menabrak untuk memaksimalkan damage dan mempercepat eliminasi musuh yang sudah lemah. Untuk musuh dengan energi tinggi, bot akan mendekati hingga jarak optimal untuk menembak, dengan kekuatan serangan yang disesuaikan berdasarkan jarak (kekuatan tinggi untuk jarak dekat, kekuatan rendah untuk jarak jauh). Ketika menabrak musuh, bot akan menembak dengan kekuatan yang disesuaikan dengan energi musuh untuk memaksimalkan damage sambil tetap hemat energi.

3.2.3 MajuMundurLock - Greedy by Movement and Shooting

Algoritma ini bertujuan untuk mendapatkan poin sebanyak-banyaknya dengan mengunci target sembari mempertahankan gerakan maju-mundur untuk menghindari tembakan dari lawan. Pendekatan ini dipilih karena akumulasi poin terbesar berada pada bonus tembakan dan bonus survival. Di awal bot langsung scanning lawan sekitar untuk di lock, hal ini dikarenakan di awal lawan masih banyak, sehingga mengunci dan menembak akan memberikan poin maksimal di awal (Jika meleset ke satu target, bisa jadi mengenai target lainnya). Bot akan berusaha mencari *space* pada arena permainan yang membuat bot aman melakukan gerakan maju-mundur, yaitu cukup jauh dari tembok dan lawan pada saat itu. Ketika melakukan gerakan maju mundur, jika ada bot lain yang melewati garis tembakan, maka bot tersebut akan di lock dan dilakukan penembakan mengikuti pergerakan bot tersebut. Jika pada saat pergerakan maju mundur bot terkena peluru atau ditabrak, maka bot akan bergerak menjauh dari musuh untuk mencari *space* baru untuk melakukan pergerakan maju mundur sembari mengaktifkan scanner untuk mengunci penembakan pada suatu bot. Jika energi bot < 30, maka bot akan berfokus pada pergerakan yang menghindari peluru tanpa menembak dan mengunci lawan lagi.

3.2.4 ZigiZaga - Greedy By ZigZag Movement

Algoritma ini berfokus pada memaksimalkan kemampuan bertahan melalui pola pergerakan yang sulit diprediksi, sambil tetap mempertahankan kemampuan menyerang yang efektif. Bot bergerak dalam pola zigzag dinamis dengan sudut dan jarak yang selalu berubah berdasarkan nilai *ticker counter*, membuat bot sangat sulit untuk dibidik oleh musuh. Bot menggunakan deteksi tembakan dengan memantau perubahan energi musuh, dan langsung melakukan manuver penghindaraan saat mendeteksi tembakan. Saat tertembak atau

bertabrakan, bot akan mengeksekusi manuver penghindaran khusus dengan perubahan arah acak, membuat pola gerakan semakin tidak terprediksi.

3.3 Analisis Efisiensi dan Efektivitas dari 4 alternatif solusi

3.3.1 NabrakBelok - Greedy by Finding the Wall for Safety Place

Efisiensi algoritma ini dapat dianalisis berdasarkan kompleksitas waktu dari setiap aksi yang dilakukan bot. Pergerakan bot menggunakan operasi dasar seperti maju, mundur, dan berbelok yang dieksekusi dalam waktu konstan, sehingga memiliki kompleksitas O(1). Pemindaian musuh dilakukan setiap iterasi dengan kompleksitas O(1), sedangkan penembakan dan evaluasi kondisi musuh juga dilakukan dalam O(1). Secara keseluruhan, setiap langkah bot berjalan dalam kompleksitas waktu O(1) per iterasi. Namun, efektivitas strategi ini bergantung pada kondisi arena dan strategi lawan. Jika musuh memiliki pola gerakan yang sulit diprediksi atau dapat menghindari serangan dengan baik, bot ini mungkin kurang optimal. Meskipun begitu, pendekatan greedy dengan keputusan lokal optimal memastikan bahwa bot dapat bertahan lebih lama dan menyerang dengan efisien tanpa membuang sumber daya secara berlebihan.

3.3.2 SeptikTank - Greedy by Target Vulnerability

Pendekatan Septik Tank memiliki kompleksitas waktu O(n), di mana n adalah jumlah musuh yang terdeteksi dalam satu putaran pertandingan. Setiap musuh yang terdeteksi diproses secara individual untuk menentukan strategi terbaik (menabrak atau menembak), yang dilakukan dalam waktu konstan O(1) per musuh. Secara efektifitas, strategi ini cukup kuat dalam menghadapi musuh dengan energi rendah karena menabrak mereka langsung dapat mempercepat eliminasi. Namun, strategi ini memiliki kelemahan jika digunakan melawan musuh yang gesit atau memiliki strategi penghindaran yang baik, karena bot dapat kehilangan banyak energi akibat benturan yang tidak diperlukan. Bot ini bekerja sangat baik dalam pertandingan yang cepat dan agresif, tetapi dalam skenario pertarungan jarak jauh dengan musuh yang defensif, bot ini bisa kehilangan keunggulannya karena terlalu fokus pada menabrak musuh yang lemah tanpa mempertimbangkan kemungkinan melawan musuh yang lebih kuat di akhir permainan.

3.3.3 MajuMundurLock - Greedy by Movement and Shooting

Bot ini memiliki efisiensi cukup tinggi dengan kompleksitas waktu O(n), di mana n adalah jumlah aksi yang diambil bot selama permainan. Setiap langkah, bot melakukan pergerakan maju-mundur, menghindari tembok sebelum menabrak, dan menembak dengan mengunci target saat musuh terdeteksi, yang semuanya berjalan dalam waktu O(1) per aksi. Efektivitas strategi ini terletak pada kemampuannya bertahan hidup dengan menghindari serangan musuh dan menjaga posisi aman, serta menembak secara akurat saat peluang terbaik muncul. Namun, kelemahannya adalah saat energi rendah (<30), bot berhenti menyerang

dan hanya fokus bertahan, yang bisa membuatnya kalah jika musuh lebih agresif. Selain itu, jika lawan memiliki pola pergerakan yang lebih adaptif atau algoritma prediksi tembakan yang lebih baik, bot bisa kesulitan mempertahankan keunggulan. Meskipun demikian, kombinasi greedy movement dan TrackFire menjadikan strategi ini cukup optimal dalam menghadapi sebagian besar lawan.

3.3.4 ZigiZaga - Greedy by ZigZag Movement

Strategi ZigiZaga memiliki efisiensi tinggi karena kompleksitas waktunya O(n), di mana n adalah jumlah musuh yang terdeteksi. Bot secara konstan bergerak zigzag tanpa perhitungan berat, sehingga setiap keputusan dilakukan dalam waktu O(1). Gerakan zigzag yang dinamis memastikan bot tetap bergerak tanpa perlu kalkulasi arah yang rumit, menjadikannya responsif terhadap serangan lawan. Dari sisi efektivitas, strategi ini sangat kuat dalam menghindari tembakan karena pergerakannya tidak mudah diprediksi, membuat lawan kesulitan dalam membidik dengan akurat. Selain itu, bot tetap menyerang dengan menyesuaikan kekuatan tembakan berdasarkan jarak musuh, yang meningkatkan peluang damage maksimal. Namun, kelemahannya adalah bot bisa tetap terkena serangan jika lawan menggunakan tembakan prediktif (predictive shooting) atau jika bot terjebak di area sempit yang menghambat pergerakan zigzag. Dengan demikian, strategi ini sangat efektif dalam pertempuran jangka panjang untuk bertahan hidup lebih lama, tetapi memerlukan kombinasi dengan strategi lain agar lebih optimal dalam menghadapi musuh yang lebih adaptif.

3.4 Strategi Greedy yang Dipilih

Setelah dijalankan pada Robocode Tank Royale, didapat bahwa strategi yang cenderung stabil meraih posisi teratas adalah MajuMundurLock (Terlampir). Strategi ini menerapkan pendekatan greedy berbasis survival, di mana bot selalu memilih keputusan yang memaksimalkan peluang bertahan dalam setiap langkahnya sembari mencari peluang untuk menyerang. Berbeda dengan ZigiZaga yang menggunakan pola zigzag untuk menghindari tembakan, MajuMundurLock lebih efisien karena hanya mengubah pergerakannya saat benar-benar dibutuhkan, yaitu ketika terkena tembakan atau ketika tertabrak. Hal ini membuat penggunaan energi lebih efektif, tetapi tetap sulit diprediksi oleh lawan.

Selain itu, MajuMundurLock menggunakan greedy dalam optimalisasi tembakan, yang hanya menembak dan mengunci jika musuh berada di garis tembakan atau dalam mode pemindaian ketika mencari posisi baru untuk bergerak maju mundur. Pendekatan ini cenderung lebih efektif dibandingkan Septik Tank, yang menembak atau menabrak musuh secara langsung. Dengan membatasi serangan pada momen tertentu, bot ini menghindari pemborosan energi dan memastikan setiap tembakan memiliki dampak maksimal. Sementara NabrakBelok terus bergerak tanpa strategi penghindaran, MajuMundurLock memilih posisi aman dengan tetap memiliki strategi penghindaran untuk menghindari serangan sekaligus mempertahankan potensi menyerang.

Keunggulan utama dari MajuMundurLock sebagai strategi greedy adalah pemilihan aksi di masa kini yang akan memberikan keuntungan terbesar dalam jangka panjang. Jika bot lain terlalu agresif atau terlalu bergantung pada pola gerakan tertentu, MajuMundurLock hanya akan mengubah strategi ketika benar-benar diperlukan, menjadikannya lebih fleksibel dan adaptif terhadap situasi pertandingan. Dengan memastikan bahwa setiap keputusan selalu membawa bot lebih dekat ke kondisi optimal, baik dalam menghindari bahaya maupun menyerang saat tepat, MajuMundurLock memiliki nilai tersendiri yang kuat sebagai implementasi greedy yang paling efektif di antara 3 strategi lain yang diuji.

BAB IV

IMPLEMENTASI DAN PENGUJIAN

4.1 Implementasi Algoritma Greedy

4.1.1 NabrakBelok - Greedy by Finding the Wall for Safety Place

Prosedur utama yang menjalankan bot

```
procedure Run()
{ Bot mulai bergerak dengan strategi greedy }

Deklarasi
    Tidak ada variabel tambahan yang perlu dideklarasikan di sini

Algoritma
    SetBodyColor(Blue)
    SetTurretColor(Black)
    SetRadarColor(Red)
    SetBulletColor(Green)
    SetScanColor(Yellow)

    moveAmount <- Max(ArenaWidth, ArenaHeight)
    movingForward <- true
    peek <- false

    TurnRight(Direction % 90)
    Forward(moveAmount)
    peek <- true
    TurnGunRight(0)
    TurnRight(135)

    while IsRunning do
        peek <- true
        Forward(moveAmount)
        peek <- false
        TurnRight(135)
    end while
end procedure
```

Prosedur tabrakan dinding

```
procedure OnHitWall(e: HitWallEvent)
{ Menghindari tabrakan dengan dinding }

Deklarasi:
    Tidak ada variabel tambahan

Algoritma:
    Print("Nabrak dinding, ya belok")
    Back(50)
    TurnRight(135)
    movingForward <- not movingForward
end procedure
```

Prosedur deteksi musuh

```
procedure OnScannedBot(e: ScannedBotEvent)
```

```
{ Menyerang musuh yang terdeteksi berdasarkan jarak }

Deklarasi:
    distance: double
    firepower: double

Algoritma:
    distance <- DistanceTo(e.X, e.Y)
    if distance < 200 then
        firepower <- 3
    else
        firepower <- 1
    end if

    Fire(firepower)

    if peek then
        Rescan()
    end if
end procedure
```

Prosedur ram/tabrakan dengan bot lain

```
procedure OnHitBot(e: HitBotEvent)
{ Menangani tabrakan dengan bot lain }

Deklarasi:
    Tidak ada variabel tambahan

Algoritma:
    if e.IsRammed then
        Print("Nabrak bot lain, mundur trus serang!")
        Back(50)
        TurnRight(45)
        Fire(2)
    end if
end procedure
```

4.1.2 SeptikTank - Greedy by Target Vulnerability

Prosedur utama yang menjalankan bot

```
procedure Run()
{ Bot mencari dan menyerang musuh berdasarkan kerentanan (energi) }

Deklarasi:
    turnDirection: integer

Algoritma:
    SetBodyColor(Green)
    SetTurretColor(DarkGreen)
    SetRadarColor(LightGreen)
    SetBulletColor(Yellow)
    turnDirection <- 1
    while IsRunning do
        TurnLeft(5 * turnDirection)
    end while
end procedure
```

Prosedur deteksi musuh

```
procedure OnScannedBot(e: ScannedBotEvent)
{ Menentukan strategi berdasarkan energi musuh yang terdeteksi }

Deklarasi:
    distance: double
    firepower: double

Algoritma:
    TurnToFaceTarget(e.X, e.Y)
    distance <- DistanceTo(e.X, e.Y)

    if e.Energy < 10 then
        Forward(distance + 5) { Tabrak musuh lemah }
    else if distance > 200 then
        Forward(distance - 150) { Dekati musuh hingga jarak optimal }
    else
        { Tembak dengan kekuatan sesuai jarak }
        if distance < 100 then
            firepower <- 3
        else if distance > 300 then
            firepower <- 1
        else
            firepower <- 3 - (((distance - 100)/200) * 2)
        end if

        if Energy > 5 then
            Fire(firepower)
        end if
    end if

    Rescan()
end procedure
```

Procedure ketika menabrak bot

```
procedure OnHitBot(e: HitBotEvent)
{ Tembak dengan kekuatan sesuai energi musuh saat tabrakan }

Deklarasi:
    Tidak ada variabel tambahan

Algoritma:
    TurnToFaceTarget(e.X, e.Y)

    if e.Energy > 16 then
        Fire(3)
    else if e.Energy > 10 then
        Fire(2)
    else if e.Energy > 4 then
        Fire(1)
    else if e.Energy > 2 then
        Fire(0.5)
    else if e.Energy > 0.4 then
        Fire(0.1)
    end if

    Forward(40) { Tabrak lagi }
end procedure
```

Prosedur ketika menabrak wall

```
procedure OnHitWall(e: HitWallEvent)
{ Menangani tabrakan dengan dinding }

Deklarasi:
    Tidak ada variabel tambahan

Algoritma:
    Back(50)
    TurnRight(90)
end procedure
```

Prosedur memutar bot

```
procedure TurnToFaceTarget(x: double, y: double)
{ Memutar bot untuk menghadap target }

Deklarasi:
    bearing: double

Algoritma:
    bearing <- BearingTo(x, y)

    if bearing >= 0 then
        turnDirection <- 1
    else
        turnDirection <- -1
    end if

    TurnLeft(bearing)
end procedure
```

Prosedur menghindar dari musuh

```
procedure Evade()
{ Manuver menghindar dengan pola acak }

Deklarasi:
    Tidak ada variabel tambahan

Algoritma:
    SetTurnRight(90 - (tickCounter % 180)) { Sudut putar acak}
    if tickCounter % 2 = 0 then { 50% kemungkinan ganti arah }
        movingForward <- not movingForward
    end if

    if movingForward then
        SetForward(150 + (tickCounter % 50))
    else
        SetBack(150 + (tickCounter % 50))
    end if
end procedure
```

4.1.3 MajuMundurLock - Greedy by Movement and Shooting

Prosedur utama yang menjalankan bot

```
procedure Run()
{ Bot mulai bergerak dengan strategi greedy }

Deklarasi
    Tidak ada variabel tambahan yang perlu dideklarasikan di sini

Algoritma
    SetTurnGunLeft(INFINITY) { [TRACKFIRE] Memutar turret untuk scanning musuh
}

    while IsRunning do
        if scanningActive then
            SetTurnRadarRight(INFINITY) { Scanner aktif saat mencari posisi
baru }

        if IsNearWall() then
            AvoidWall() { Hindari tabrakan dengan dinding }
        else
            MoveBackAndForth() { Gerakan maju-mundur untuk survival }
        end if
    end while
end procedure
```

Prosedur pergerakan maju mundur

```
procedure MoveBackAndForth()
{ Bot terus bergerak maju dan mundur untuk menghindari tembakan musuh }

Deklarasi:
    Tidak ada variabel tambahan

Algoritma:
    if movingForward then
        SetForward(100)
    else
        SetBack(100)
    end if

    WaitFor(MoveCompleteCondition(this))
    movingForward ← not movingForward { Balik arah setelah satu pergerakan
selesai }
end procedure
```

Fungsi pengecekan posisi terhadap dinding

```
function IsNearWall() -> boolean
{ Menentukan apakah bot hampir menabrak dinding }
Deklarasi:
    Tidak ada variabel tambahan

Algoritma:
    return (X < WALL_MARGIN or X > ArenaWidth - WALL_MARGIN or
            Y < WALL_MARGIN or Y > ArenaHeight - WALL_MARGIN)
```

```
end function
```

Prosedur menjauh dari dinding sebelum menabraknya

```
procedure AvoidWall()
{ Jika bot hampir menabrak dinding, ubah arah sebelum terjadi tabrakan }
Deklarasi:
    Tidak ada variabel tambahan

Algoritma:
    SetTurnRight(90 + random.Next(-30, 30)) { Putar dengan sudut acak }
    WaitFor(TurnCompleteCondition(this))
    SetForward(200) { Bergerak menjauh dari tembok }
    WaitFor(MoveCompleteCondition(this))
end procedure
```

Prosedur menjauh dari bot untuk mencari posisi aman

```
procedure AvoidWall()
{ Jika bot hampir menabrak dinding, ubah arah sebelum terjadi tabrakan }
Deklarasi:
    Tidak ada variabel tambahan

Algoritma:
    SetTurnRight(90 + random.Next(-30, 30)) { Putar dengan sudut acak }
    WaitFor(TurnCompleteCondition(this))
    SetForward(200) { Bergerak menjauh dari tembok }
    WaitFor(MoveCompleteCondition(this))
end procedure
```

Prosedur mendeteksi musuh dan mengunci untuk ditembak

```
procedure AvoidWall()
{ Jika bot hampir menabrak dinding, ubah arah sebelum terjadi tabrakan }
Deklarasi:
   Tidak ada variabel tambahan

Algoritma:
   SetTurnRight(90 + random.Next(-30, 30)) { Putar dengan sudut acak }
   WaitFor(TurnCompleteCondition(this))
   SetForward(200) { Bergerak menjauh dari tembok }
   WaitFor(MoveCompleteCondition(this))
end procedure
```

Prosedur pergerakan setelah terkena tembakan

```
procedure OnHitByBullet(e: HitByBulletEvent)
{ Jika terkena tembakan, lakukan manuver menghindar }
Deklarasi:
    Tidak ada variabel tambahan

Algoritma:
    scanningActive ← true { Aktifkan scanner sementara }

    SetTurnRight(random.Next(30, 60)) { Putar bot ke arah acak }
    WaitFor(TurnCompleteCondition(this))
    SetForward(50) { Bergerak menjauh dari posisi sebelumnya }
```

```
    WaitFor(MoveCompleteCondition(this))

    scanningActive ← false { Matikan scanner setelah berpindah posisi }
end procedure
```

Prosedur pergerakan setelah menabrak dinding

```
procedure OnHitWall(e: HitWallEvent)
{ Jika bot menabrak dinding, segera berbelok menjauh }
Deklarasi:
    Tidak ada variabel tambahan

Algoritma:
    AvoidWall()
end procedure
```

Prosedur pergerakan setelah bertabrakan

```
procedure OnHitBot(e: HitBotEvent)
{ Jika bot menabrak bot lain, segera menjauh }
Deklarasi:
    Tidak ada variabel tambahan

Algoritma:
    AvoidBot()
end procedure
```

Fungsi mencari musuh terdekat untuk di kunci pergerakannya

```
function FindNearestEnemy() -> (double, double)?
{ Mencari musuh dengan jarak terdekat dari daftar musuh yang terdeteksi }
Deklarasi:
    minDistance : double
    nearestEnemy : (double, double)?

Algoritma:
    minDistance ← INFINITY
    nearestEnemy ← NULL

    for each enemy in enemies do
        distance ← DistanceTo(enemy.X, enemy.Y)
        if distance < minDistance then
            minDistance ← distance
            nearestEnemy ← enemy
        end if
    end for

    return nearestEnemy
end function
```

Class menunggu pergerakan selesai

```
class MoveCompleteCondition extends Condition
{ Menunggu sampai bot selesai bergerak sebelum melanjutkan aksi berikutnya }
Deklarasi:
    bot : Bot
```

```
Algoritma:
    constructor(bot: Bot)
        this.bot ← bot
    end constructor
    function Test() -> boolean
        return bot.DistanceRemaining = 0
    end function
end class
```

Class menunggu belokan selesai

```
class TurnCompleteCondition extends Condition
{ Menunggu sampai bot selesai berbelok sebelum melanjutkan aksi berikutnya }
Deklarasi:
    bot : Bot

Algoritma:
    constructor(bot: Bot)
        this.bot ← bot
    end constructor
    function Test() -> boolean
        return bot.TurnRemaining = 0
    end function
end class
```

4.1.4 ZigiZaga - Greedy by ZigZag Movement

Prosedur utama yang menjalankan bot

```
procedure Run()
{ Bot menjalankan pola zigzag yang sulit diprediksi }

Deklarasi:
    tickCounter: integer
    movingForward: boolean
    lastEnemyEnergy: double

Algoritma:
    tickCounter ←   0
    movingForward ←  true
    while IsRunning do
        tickCounter ←  tickCounter + 1
        SetTurnRight(30 - (tickCounter % 60)) { Sudut zigzag dinamis }
        if movingForward then
            SetForward(150 + (tickCounter % 100)) { Jarak zigzag dinamis }
        else
            SetBack(150 + (tickCounter % 100))
        end if
        if tickCounter % 50 = 0 then
            SetTurnRadarRight(360) { Scan radar periodik }
        end if
        if tickCounter % 100 = 0 then
            movingForward ← not movingForward { Balik arah periodik }
        end if
        Go()
    end while
end procedure
```

Prosedur deteksi musuh

```
procedure OnScannedBot(e: ScannedBotEvent)
{ Mendeteksi dan menembak musuh, serta mendeteksi tembakan musuh }

Deklarasi:
    distance: double
    firepower: double
    gunBearing: double
    radarBearing: double

Algoritma:
    radarBearing <- RadarBearingTo(e.X, e.Y)
    SetTurnRadarRight(radarBearing * 1.5) { Lock radar ke musuh }
    gunBearing <- GunBearingTo(e.X, e.Y)
    SetTurnGunRight(gunBearing) { Arahkan senjata }
    if Abs(gunBearing) < 10 then { Jika sudah terarah dengan baik }
        distance <- DistanceTo(e.X, e.Y)
        if distance < 100 then
            firepower <- 3.0
        else if distance < 200 and Energy > 30 then
            firepower <- 2.0
        else
            firepower <- 1.0
        end if
        SetFire(firepower)
    end if

    if lastEnemyEnergy > e.Energy and lastEnemyEnergy - e.Energy <= 3.0 then
        Evade() { Menghindar jika musuh menembak }
    end if

    lastEnemyEnergy <- e.Energy
end procedure
```

Procedure terkena tembakan

```
procedure OnHitByBullet(e: HitByBulletEvent)
{ Menghindar saat terkena tembakan }

Deklarasi:
    Tidak ada variabel tambahan

Algoritma:
    Evade() { Lakukan manuver menghindar }
    tickCounter <- tickCounter + 25 { Reset pola gerakan }
end procedure
```

Prosedur ketika menabrak wall

```
 procedure OnHitWall(e: HitWallEvent)
{ Menangani tabrakan dengan dinding }
Deklarasi:
    Tidak ada variabel tambahan
  Algoritma:
    Stop()
    SetTurnRight(120)
    movingForward <- not movingForward
    if movingForward then
   SetForward(150)
    else
```

```
        SetBack(150)
    end if
    Go()
end procedure
```

Prosedur ketika tabrakan dengan bot lain

```
procedure OnHitBot(e: HitBotEvent)
{ Menangani tabrakan dengan bot lain }

Deklarasi:
    Tidak ada variabel tambahan

Algoritma:
    if Energy > e.Energy then { Jika energi kita lebih tinggi }
        SetTurnGunRight(GunBearingTo(e.X, e.Y))
        SetFire(3.0) { Tembak dengan kekuatan maksimal }
        Evade() { Lalu menghindar }
    else
        Evade() { Langsung menghindar jika energi lebih rendah }
    end if
end procedure
```

Prosedur menghindar dari musuh

```
procedure Evade()
{ Manuver menghindar dengan pola acak }

Deklarasi:
    Tidak ada variabel tambahan

Algoritma:
    SetTurnRight(90 - (tickCounter % 180)) { Sudut putar acak}
    if tickCounter % 2 = 0 then { 50% kemungkinan ganti arah }
        movingForward <- not movingForward
    end if

    if movingForward then
        SetForward(150 + (tickCounter % 50))
    else
        SetBack(150 + (tickCounter % 50))
    end if
end procedure
```

4.2. Struktur Data Program

4.2.1 NabrakBelok - Greedy by Finding the Wall for Safety Place

1. movingForward() – Menentukan apakah bot sedang bergerak maju atau mundur.
2. moveAmount() – Menentukan jarak maksimum yang akan ditempuh bot dalam satu pergerakan.
3. peek() – Menentukan apakah bot dalam mode pengintaian saat mendeteksi musuh.
4. Run() – Prosedur utama yang menjalankan bot, mengatur pergerakan, scanning, dan strategi menyerang.

5. OnHitWall(e) – Dipanggil ketika bot menabrak dinding, menyebabkan bot mundur dan berbelok untuk menghindari tabrakan berulang.
6. OnScannedBot(e**)** – Menentukan jarak musuh menggunakan DistanceTo(x, y), lalu menembak menggunakan Fire(power) sesuai dengan jarak target.
7. OnHitBot(e): Dipanggil ketika bot bertabrakan dengan lawan, menyebabkan bot mundur dan menyerang dengan tembakan jika memungkinkan.
8. TurnRight(): Memutar bot ke kanan sesuai sudut yang ditentukan.
9. Forward(): Menggerakkan bot maju sejauh jarak yang ditentukan.
   Back(): Menggerakkan bot mundur sejauh jarak yang ditentukan.
10. Fire(): Menembakkan peluru dengan kekuatan tertentu berdasarkan jarak musuh.
11. Rescan(): Mengulangi proses scanning setelah menembak untuk mencari target baru
12. DistanceTo(x, y): Menghitung jarak bot dengan koordinat musuh untuk menentukan strategi serangan yang optimal.

4.2.2 SeptikTank - Greedy by Target Vulnerability

1. turnDirection: Menentukan arah putaran bot (1 atau -1).
2. Run(): Mengatur pergerakan dasar dengan berputar terus menerus mencari musuh.
3. OnScannedBot(): Menentukan strategi berdasarkan energi musuh.
4. TurnToFaceTarget(): Menghadapkan bot langsung ke arah musuh terdeteksi.
5. DistanceTo(): Menghitung jarak ke musuh untuk penentuan strategi.
6. Forward(): Maju untuk menabrak musuh energi rendah atau mencapai jarak optimal.
7. Fire(): Menembak dengan kekuatan berdasarkan jarak ke musuh.
8. OnHitBot(): Menembak dengan kekuatan berdasarkan energi musuh saat tabrakan.
9. OnHitWall(): Mundur dan berputar saat menabrak dinding.
10. BearingTo(): Menghitung sudut yang diperlukan untuk menghadap target.
11. Rescan(): Mempertahankan kontak dengan musuh yang terdeteksi.
12. TurnLeft(): Berputar untuk menghadap target dengan sudut terpendek.

4.2.3 MajuMundurLock - Greedy by Movement and Shooting

1. random : Menghasilkan angka acak untuk variasi dalam gerakan dan penghindaran.
2. WALL_MARGIN : Menentukan jarak aman dari dinding agar bot tidak menabrak.
3. movingForward : Menunjukkan apakah bot sedang maju atau mundur.
4. scanningActive : Menunjukkan apakah radar aktif untuk mencari musuh.
5. enemies : Dictionary yang menyimpan ID musuh dan posisinya untuk tracking.
6. Run() : Memulai bot dan mengatur pergerakan serta scanning musuh.
7. MoveBackAndForth() : Menjaga pola gerakan maju-mundur untuk menghindari tembakan.
8. IsNearWall() : Mengecek apakah bot hampir menabrak dinding.
9. AvoidWall() : Menghindari tabrakan dengan tembok sebelum terjadi.
10. AvoidBot() : Menghindari bot lain jika terdeteksi terlalu dekat.

11. OnScannedBot(e) : Menggunakan TrackFire untuk mengunci target dan menembak secara akurat.
12. OnHitByBullet(e) : Menghindari setelah terkena tembakan untuk mencegah serangan beruntun.
13. OnHitWall(e) : Langsung berbelok jika menabrak tembok.
14. OnHitBot(e) : Menghindari setelah bertabrakan dengan bot lain.

4.2.4 ZigiZaga - Greedy by ZigZag Movement

1. tickCounter: Menghitung iterasi untuk menghasilkan variasi dinamis dalam pola zigzag.
2. movingForward: Menunjukkan arah pergerakan bot (maju atau mundur).
3. lastEnemyEnergy: Menyimpan energi musuh terakhir untuk mendeteksi tembakan.
4. Run(): Mengatur pergerakan zigzag dengan sudut dan jarak yang selalu berubah.
5. SetTurnRight(): Mengatur sudut berputar berdasarkan nilai tickCounter.
6. SetForward()/SetBack(): Mengatur jarak pergerakan yang bervariasi berdasarkan tickCounter.
7. OnScannedBot(): Mengunci radar pada musuh dan menentukan kekuatan tembakan berdasarkan jarak.
8. GunBearingTo(): Menghitung sudut untuk mengarahkan senjata ke musuh.
9. DistanceTo(): Menghitung jarak ke musuh untuk menentukan strategi serangan.
10. Evade(): Metode khusus untuk manuver menghindar dengan pola zigzag acak.
11. OnHitByBullet(): Mengaktifkan manuver penghindaran dan mengubah pola gerakan.
12. OnHitWall(): Membalik arah dan berputar untuk menghindari dinding.
13. OnHitBot(): Menembak jika energi lebih tinggi atau langsung menghindar jika energi lebih rendah.

4.3 Pengujian

Pada tahap pengujian, dilakukan 10 kali pertandingan antara ke 4 bot dengan sistem 1 vs 1 vs 1 vs 1 dan didapatkan hasil sebagaimana terlampir pada lampiran. Dari hasil tersebut, MajuMundurLock berhasil meraih 8 kali peringkat 1, dan 2 kali peringkat 2. Hasil ini tentunya menunjukkan bahwa dari ke 4 strategi yang ada, MajuMundurLock punya penerapan yang paling efektif, efisien dan stabil dalam meraih poin maksimal dalam permainan Robocode Tank Royale.

Tabel 1. Statistik Hasil 10 Kali Pengujian

Nama Bot	Peringkat 1	Peringkat 2	Peringkat 3	Peringkat 4
MajuMundurLock	8 kali	2 kali	-	-
ZigiZaga	-	1 kali	5 kali	4 kali
NabrakBelok	1 kali	2 kali	2 kali	5 kali

SeptikTank	1 kali	5 kali	3 kali	1 kali

4.4 Analisis Hasil Pengujian

Dari hasil pengujian yang dilakukan sebanyak 10 pertandingan, bot MajuMundurLock menunjukkan efektivitas tinggi dalam menerapkan strategi greedy, dengan memenangkan 8 pertandingan dan meraih posisi kedua dalam 2 pertandingan lainnya. Hasil ini mengindikasikan bahwa strategi greedy berbasis survival dan optimalisasi tembakan yang diterapkan berhasil dalam mayoritas kondisi pertempuran. Bot ini mampu memaksimalkan peluang bertahan hidup, menghindari serangan lawan secara efisien, dan hanya menembak ketika ada peluang yang benar-benar menguntungkan. Konsistensi dalam meraih poin tinggi menunjukkan bahwa pendekatan **greedy lokal** yang diterapkan hampir selalu mengarah pada **solusi global** yang optimal, yakni bertahan hingga akhir pertandingan dan mengeliminasi lawan secara efektif.

Namun, strategi greedy MajuMundurLock tidak selalu berhasil mendapatkan nilai optimal, sebagaimana terlihat dari 2 pertandingan di mana bot ini hanya meraih peringkat kedua. Salah satu kelemahan utama yang menyebabkan hal ini adalah ketika pergerakan bot terhambat, misalnya jika bot terjebak di sudut arena atau tertabrak oleh lawan dalam posisi yang tidak menguntungkan. Dalam situasi ini, strategi maju-mundur menjadi kurang efektif karena ruang gerak yang terbatas mengurangi efektivitas dalam menghindari tembakan. Selain itu, strategi penghindaran yang berbasis reaksi terhadap serangan bisa menjadi tidak cukup cepat jika musuh menggunakan pola serangan yang lebih agresif atau tembakan prediktif.

Dari hasil ini, dapat disimpulkan bahwa strategi greedy yang diterapkan sudah cukup optimal dalam sebagian besar pertandingan, tetapi tetap memiliki keterbatasan dalam situasi tertentu, seperti ruang gerak yang sempit atau ketika menghadapi lawan yang dapat membaca pola pergerakan bot. Untuk meningkatkan performa lebih lanjut, bot ini dapat dikembangkan dengan sistem deteksi posisi yang lebih adaptif, sehingga mampu menyesuaikan pola gerakannya saat menghadapi medan yang lebih sulit dan sulit bergerak. Meskipun demikian, dengan tingkat kemenangan 80% dan selalu berada di posisi 2 besar, MajuMundurLock tetap menjadi strategi greedy paling efektif dan efisien dibandingkan ketiga strategi lainnya dalam Robocode Tank Royale.

BAB V

KESIMPULAN DAN SARAN

5.1 Kesimpulan

Algoritma greedy adalah metode pengambilan keputusan lokal yang optimal pada setiap langkah dengan harapan dapat menghasilkan solusi global yang terbaik. Dalam Robocode Tank Royale, pendekatan greedy diterapkan dalam berbagai strategi, seperti penghindaran serangan, optimalisasi pergerakan, pertahanan, dan lainnya. Dari empat strategi MajuMundurLock, Septik Tank, ZigiZaga, dan NabrakBelok, masing-masing memiliki keunggulan dan kelemahan dalam mengimplementasikan greedy. Dari hasil pengujian, MajuMundurLock menjadi strategi yang paling stabil dan efektif. Strategi ini menerapkan greedy dalam survival, di mana bot selalu bergerak maju-mundur untuk menghindari serangan dan hanya menyerang pada momen yang menguntungkan. Meski demikian, strategi ini masih memiliki kelemahan, seperti kurangnya fleksibilitas saat terjebak di area sempit atau menghadapi musuh dengan tembakan prediktif.

5.2 Saran

Terdapat beberapa saran untuk pengembangan bot lebih lanjut

1. Membuat kode yang lebih efisien dengan struktur kode yang lebih readable.
2. Memperbanyak eksperimen dalam penentuan strategi greedy.
3. Membuat variasi greedy yang lebih adaptif terhadap berbagai kondisi.
4. Memahami API lebih dalam agar strategi yang sudah dibuat dapat di implementasikan dengan lebih detail.

LAMPIRAN

NO	Poin	Ya	Tidak
1	Bot dapat dijalankan pada engine yang sudah dimodifikasi asisten	☑	
2	Membuat 4 solusi greedy dengan heuristic yang berbeda	☑	
3	Membuat laporan sesuai dengan spesifikasi	☑	
4	Membuat Video bonus dan diunggah di youtube	☑	

Tautan Repository:

https://github.com/rafifrs/Tubes1_JaMin

Tautan Video Youtube:

https://youtu.be/8BdYBYdUatE?si=JdeK4Uj6R3zgUQhA

HASIL PENGUJIAN

Results for 10 rounds

Rank	Name	Total Score	Survival	Surv. Bonus	Bullet Dmg.	Bullet Bon...	Ram Dmg.	Ram Bonus	1sts	2nds	3rds
1	MajuMundurLock ...	1212	450	60	616	77	8	0	3	1	0
2	Septik Tank 1.0	848	300	30	466	20	31	0	1	2	1
3	AltBot1 1.0	843	400	30	352	41	19	0	1	2	1
4	ZigiZaga 1.0	317	200	0	56	0	61	0	0	0	3

Results for 10 rounds

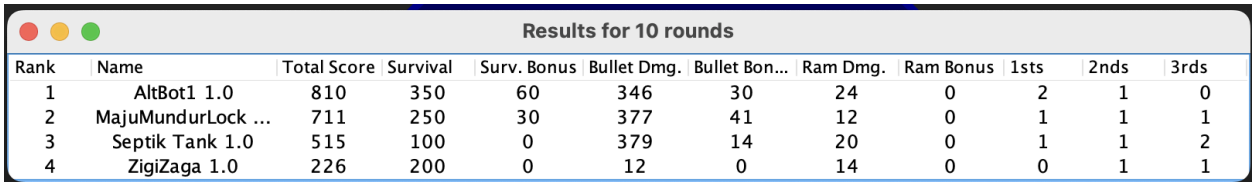
Rank	Name	Total Score	Survival	Surv. Bonus	Bullet Dmg.	Bullet Bon...	Ram Dmg.	Ram Bonus	1sts	2nds	3rds
1	MajuMundurLock ...	1267	450	90	630	80	17	0	3	0	0
2	ZigiZaga 1.0	550	450	30	50	0	20	0	1	2	1
3	Septik Tank 1.0	442	100	0	300	0	14	28	0	2	1
4	AltBot1 1.0	353	200	0	128	10	14	0	0	0	2

Results for 10 rounds

Rank	Name	Total Score	Survival	Surv. Bonus	Bullet Dmg.	Bullet Bon...	Ram Dmg.	Ram Bonus	1sts	2nds	3rds
1	MajuMundurLock ...	874	300	60	450	56	8	0	2	0	0
2	Septik Tank 1.0	787	250	0	461	61	14	0	0	4	0
3	ZigiZaga 1.0	606	450	30	102	10	13	0	1	0	3
4	AltBot1 1.0	454	200	30	204	6	13	0	1	0	1

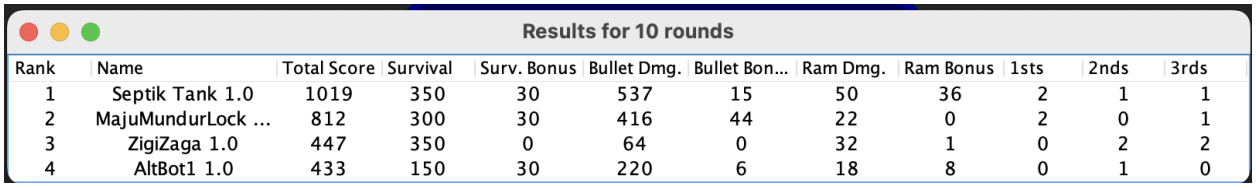
Results for 10 rounds

Rank	Name	Total Score	Survival	Surv. Bonus	Bullet Dmg.	Bullet Bon...	Ram Dmg.	Ram Bonus	1sts	2nds	3rds
1	MajuMundurLock ...	889	400	60	362	45	22	0	2	1	1
2	Septik Tank 1.0	651	250	30	330	3	12	26	1	0	2
3	ZigiZaga 1.0	320	250	0	44	0	26	0	0	2	1
4	AltBot1 1.0	313	150	0	152	0	11	0	1	1	0



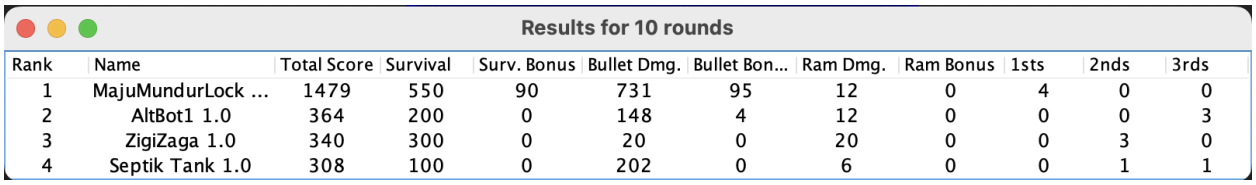
Results for 10 rounds

Rank	Name	Total Score	Survival	Surv. Bonus	Bullet Dmg.	Bullet Bon...	Ram Dmg.	Ram Bonus	1sts	2nds	3rds
1	AltBot1 1.0	810	350	60	346	30	24	0	2	1	0
2	MajuMundurLock ...	711	250	30	377	41	12	0	1	1	1
3	Septik Tank 1.0	515	100	0	379	14	20	0	1	1	2
4	ZigiZaga 1.0	226	200	0	12	0	14	0	0	1	1



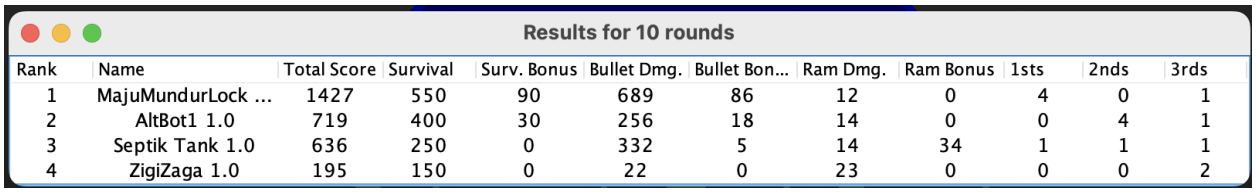
Results for 10 rounds

Rank	Name	Total Score	Survival	Surv. Bonus	Bullet Dmg.	Bullet Bon...	Ram Dmg.	Ram Bonus	1sts	2nds	3rds
1	Septik Tank 1.0	1019	350	30	537	15	50	36	2	1	1
2	MajuMundurLock ...	812	300	30	416	44	22	0	2	0	1
3	ZigiZaga 1.0	447	350	0	64	0	32	1	0	2	2
4	AltBot1 1.0	433	150	30	220	6	18	8	0	1	0



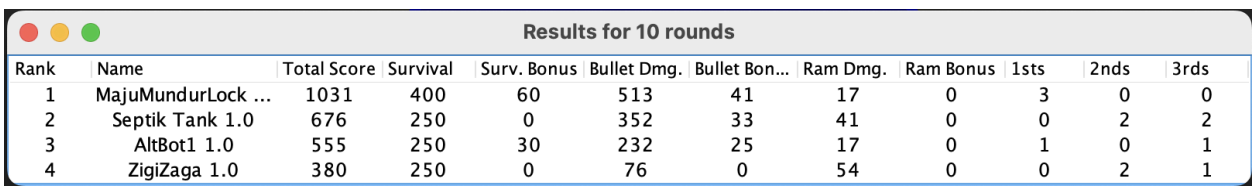
Results for 10 rounds

Rank	Name	Total Score	Survival	Surv. Bonus	Bullet Dmg.	Bullet Bon...	Ram Dmg.	Ram Bonus	1sts	2nds	3rds
1	MajuMundurLock ...	1479	550	90	731	95	12	0	4	0	0
2	AltBot1 1.0	364	200	0	148	4	12	0	0	0	3
3	ZigiZaga 1.0	340	300	0	20	0	20	0	0	3	0
4	Septik Tank 1.0	308	100	0	202	0	6	0	0	1	1



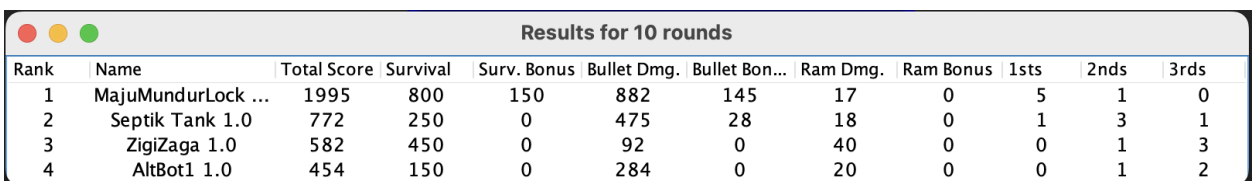
Results for 10 rounds

Rank	Name	Total Score	Survival	Surv. Bonus	Bullet Dmg.	Bullet Bon...	Ram Dmg.	Ram Bonus	1sts	2nds	3rds
1	MajuMundurLock ...	1427	550	90	689	86	12	0	4	0	1
2	AltBot1 1.0	719	400	30	256	18	14	0	0	4	1
3	Septik Tank 1.0	636	250	0	332	5	14	34	1	1	1
4	ZigiZaga 1.0	195	150	0	22	0	23	0	0	0	2



Results for 10 rounds

Rank	Name	Total Score	Survival	Surv. Bonus	Bullet Dmg.	Bullet Bon...	Ram Dmg.	Ram Bonus	1sts	2nds	3rds
1	MajuMundurLock ...	1031	400	60	513	41	17	0	3	0	0
2	Septik Tank 1.0	676	250	0	352	33	41	0	0	2	2
3	AltBot1 1.0	555	250	30	232	25	17	0	1	0	1
4	ZigiZaga 1.0	380	250	0	76	0	54	0	0	2	1



Results for 10 rounds

Rank	Name	Total Score	Survival	Surv. Bonus	Bullet Dmg.	Bullet Bon...	Ram Dmg.	Ram Bonus	1sts	2nds	3rds
1	MajuMundurLock ...	1995	800	150	882	145	17	0	5	1	0
2	Septik Tank 1.0	772	250	0	475	28	18	0	1	3	1
3	ZigiZaga 1.0	582	450	0	92	0	40	0	0	1	3
4	AltBot1 1.0	454	150	0	284	0	20	0	0	1	2

DAFTAR PUSTAKA

https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2024-2025/04-Algoritma-Greedy-(2025)-Bag1.pdf
(Terakhir di akses : 23 Maret 2025, 22.33)
https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2024-2025/05-Algoritma-Greedy-(2025)-Bag2.pdf
(Terakhir di akses : 23 Maret 2025, 23.14)
https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2024-2025/06-Algoritma-Greedy-(2025)-Bag3.pdf
(Terakhir di akses : 23 Maret 2025, 23.14)