

**Tugas Besar 1 IF2211 Strategi Algoritma
Pemanfaatan Algoritma Greedy dalam pembuatan bot
permainan Robocode Tank Royale**



Disusun Oleh:
Adiel Rum (10123004)
Muhammad Zakkiy (10122074)
Qodri Azkarayan (13523010)

**SEKOLAH TEKNIK ELEKTRO DAN INFORMATIKA
INSTITUT TEKNOLOGI BANDUNG
TAHUN AKADEMIK 2024/2025**

Daftar Isi

Daftar Isi.....	2
Bab I: Deskripsi Tugas.....	3
Bab 2: Landasan Teori.....	8
2.1 Dasar Teori Algoritma Greedy.....	8
2.2 Cara Kerja Program Robocode.....	9
Bab 3: Aplikasi Strategi Greedy.....	11
3.1 Persoalan Robocode Dalam Elemen Algoritma Greedy.....	11
3.2 Eksplorasi Alternatif Solusi Greedy.....	12
Bab 4: Implementasi dan Pengujian.....	14
4.1 Implementasi Solusi Greedy.....	14
4.2 Pengujian.....	21
Bab 5: Kesimpulan dan Saran.....	24
5.1 Kesimpulan.....	24
5.2 Saran.....	24
Lampiran.....	25
Daftar Pustaka.....	26

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 melewati turn tersebut. Jika bot melewati 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)

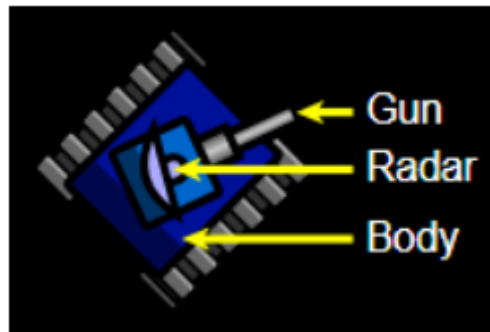
Saat menembakkan peluru, meriam akan menjadi panas. Peluru yang lebih berat menghasilkan lebih banyak panas dibandingkan peluru yang lebih ringan. Ketika meriam terlalu panas, bot tidak dapat menembak hingga suhu meriam turun ke nol. Selain itu, meriam juga sudah dalam keadaan panas di awal round dan 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:



Gambar 1. Anatomi Tank dalam Robocode

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.

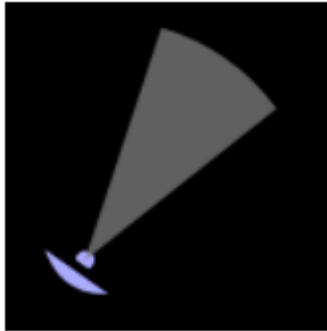
Bagian paling lambat adalah tubuh tank, yang dalam kondisi terbaik dapat berputar hingga 10 derajat per giliran. Namun, ini bergantung pada kecepatan bot saat ini. Semakin cepat bot bergerak, semakin lambat kemampuannya untuk berbelok.

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

10. Pemindaian

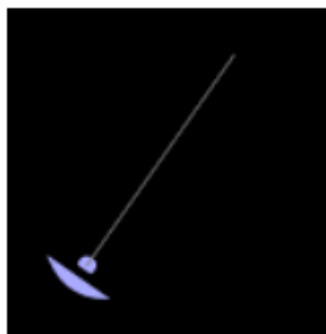
Aspek penting dalam Robocode adalah memindai bot musuh menggunakan radar. Radar dapat mendeteksi bot dalam jangkauan hingga 1200 piksel. Musuh yang berada lebih dari 1200 piksel dari bot tidak dapat terdeteksi atau dipindai oleh radar.

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.



Gambar 2. Sapuan Radar Bergerak

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.



Gambar 3. Sapuan Radar Ketika Tidak Bergerak

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 lainnya yang masih bertahan pada ronde tersebut mendapatkan 50 poin.
- Last Survival Bonus: Bot terakhir yang bertahan pada suatu ronde akan mendapatkan 10 poin dikali dengan banyaknya musuh.
- Ram Damage: Bot mendapatkan poin sebesar 2 kalinya damage yang dibuat kepada bot musuh dengan cara menabrak.
- Ram Damage Bonus: Apabila musuh terbunuh dengan cara ditabrak, bot mendapatkan poin sebesar 30% dari damage yang dibuat kepada musuh yang terbunuh.

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

Untuk informasi lebih lengkap, silahkan buka dokumentasi Tank Royale pada link [berikut](#).

Bab 2: Landasan Teori

2.1 Dasar Teori Algoritma Greedy

Algoritma greedy merupakan suatu teknik algoritmik yang mengikuti pendekatan penyelesaian masalah dengan memilih pilihan yang tampak terbaik pada saat itu. Algoritma ini bertujuan untuk membuat pilihan yang optimal secara lokal pada setiap langkah dengan harapan mencapai solusi yang optimal secara global. Keputusan yang telah diambil tidak akan pernah dipertimbangkan ulang, meskipun pilihan tersebut pada akhirnya mungkin salah. Inti dari algoritma greedy adalah optimasi jangka pendek; algoritma ini berfokus pada apa yang tampak sebagai pilihan terbaik saat ini tanpa mempertimbangkan dampak jangka panjang dari keputusannya. Pendekatan ini dapat menjadi keuntungan dalam hal kesederhanaan dan kecepatan, tetapi juga dapat menjadi kerugian karena berpotensi menghasilkan solusi yang kurang optimal. Dalam konteks Robocode, sebuah bot greedy mungkin memprioritaskan serangan langsung daripada penentuan posisi strategis, yang dapat merugikan dalam jangka panjang.

Karakteristik utama algoritma greedy meliputi pembuatan "pilihan greedy" pada setiap langkah yang layak dan membantu memecah masalah menjadi sub-masalah yang lebih kecil. Setelah pilihan dibuat, pilihan tersebut dianggap final dan tidak dapat diubah kembali, yang dikenal sebagai "properti greedy". Kelayakan suatu pilihan sangat penting. Dalam Robocode, misalnya, bot greedy mungkin "memilih" untuk menembak, tetapi pilihan ini hanya layak jika senjatanya tidak terlalu panas dan bot memiliki energi yang cukup.

Solusi optimal secara global dapat dicapai dengan memilih pilihan optimal pada setiap langkah tanpa perlu mempertimbangkan kembali langkah-langkah sebelumnya. Properti ini sangat penting agar algoritma greedy dapat menjamin solusi yang optimal. Namun, banyak masalah, termasuk dinamika kompleks dalam Robocode, mungkin tidak memenuhi properti ini. Contoh jalur dengan jumlah terbesar dalam dengan jelas menunjukkan bagaimana pilihan optimal secara lokal mencegah tercapainya solusi optimal secara global. Hal ini mengindikasikan bahwa pendekatan greedy murni dalam Robocode mungkin melewatkan peluang untuk hasil jangka panjang yang lebih baik.

Solusi optimal untuk seluruh masalah mengandung solusi optimal untuk sub-masalahnya. Meskipun Robocode melibatkan pengambilan keputusan pada setiap giliran (sub-masalah), apakah urutan tindakan optimal untuk seluruh pertempuran terdiri dari tindakan optimal pada setiap giliran individu perlu dipertimbangkan dengan cermat. Pendekatan greedy mengasumsikan bahwa hal ini benar. Misalnya, mundur pada satu giliran (tampak sub-optimal untuk serangan langsung) mungkin optimal untuk kelangsungan hidup dalam jangka panjang, tetapi bot greedy murni mungkin tidak menyadari hal ini.

Algoritma greedy cukup berhasil dalam beberapa masalah seperti pengkodean Huffman dan algoritma Dijkstra. Namun, algoritma ini sering gagal dalam masalah lain (misalnya, menemukan jalur terpanjang, masalah knapsack tanpa item fraksional). Keberhasilan

algoritma greedy sangat bergantung pada masalah yang dihadapi. Penting untuk menganalisis karakteristik masalah Robocode Tank Royale untuk menilai potensi efektivitas pendekatan greedy.

Algoritma ini sering kali lebih mudah dijelaskan dan diimplementasikan dibandingkan dengan pendekatan yang lebih kompleks seperti dynamic programming. Algoritma ini juga dapat berkinerja lebih baik dalam hal kecepatan dan penggunaan sumber daya, karena biasanya tidak menjelajahi Kekurangan utama adalah algoritma ini tidak selalu menghasilkan solusi yang optimal. Algoritma ini dapat terjebak dalam optimum lokal, tanpa menyadari solusi keseluruhan yang lebih baik.

2.2 Cara Kerja Program Robocode

Robocode adalah permainan pemrograman di mana tujuannya adalah membuat kode bot (tank) untuk bersaing dengan bot lain di arena pertempuran virtual. "Pemain" adalah pemrogram yang menulis "otak" bot. Tujuannya adalah untuk mengembangkan tank yang dapat bertahan dan mengalahkan tank musuh untuk mencapai skor dan peringkat tertinggi. Pertempuran terjadi di medan perang hingga hanya satu bot yang tersisa, mirip dengan permainan Battle Royale. Sistem penilaian dalam Robocode Tank Royale menghargai kerusakan yang ditimbulkan, kelangsungan hidup, dan potensi tabrakan. Strategi greedy yang berhasil perlu mempertimbangkan faktor-faktor ini dalam fungsi tujuannya.

Sebuah pertempuran terdiri dari beberapa ronde, dan setiap ronde dibagi menjadi giliran. Setiap bot memiliki waktu terbatas (turn timeout, biasanya 30-50 ms) untuk memutuskan dan mengeksekusi tindakannya pada setiap giliran. Sebuah bot terdiri dari badan untuk bergerak, turret dengan senjata untuk menembak, dan radar untuk memindai. Komponen-komponen ini dapat berputar secara independen. Medan perang dibatasi oleh dinding. Bot kehilangan energi ketika terkena peluru musuh, ditabrak musuh, atau ketika bertabrakan dengan dinding atau bot lain. Energi diperoleh dengan mengenai lawan dengan peluru. Pengelolaan energi sangat penting untuk kelangsungan hidup dan kemampuan menyerang. Bot greedy perlu menyeimbangkan tindakan ofensif (menembak) dengan manuver defensif (menghindari kerusakan) untuk menghemat energi.

Aksi Bot dalam Robocode:

Pergerakan: Bot dapat bergerak maju dan mundur dengan akselerasi maksimum 1 unit/giliran dan deselerasi 2 unit/giliran. Kecepatan maksimum adalah 8 unit/giliran. Tingkat putaran dibatasi oleh kecepatan (maksimum 10 derajat/giliran saat diam, berkurang pada kecepatan lebih tinggi). Fisika pergerakan membatasi pilihan langsung. Bot greedy tidak dapat langsung mengubah arah atau kecepatan. Ia perlu merencanakan pergerakan dalam giliran diskrit dengan mempertimbangkan batasan-batasan ini.

Pemindaian: Radar memiliki radius pemindaian 1200 piksel dan dapat berputar hingga 45 derajat per giliran. Sebuah *ScannedBotEvent* dihasilkan ketika musuh terdeteksi dalam busur

pemindaian. Pergerakan radar yang berkelanjutan sangat penting untuk mendeteksi musuh. Pemindaian memberikan informasi penting tentang lingkungan. Pengambilan keputusan bot greedy sangat bergantung pada informasi yang dikumpulkan melalui pemindaian pada giliran saat ini dan mungkin giliran sebelumnya.

Penembakan: Senjata dapat menembakkan peluru dengan daya tembak antara 0,1 (minimum) dan 3 (maksimum). Kerusakan peluru adalah $4 * \text{daya tembak}$ (ditambah bonus jika $\text{daya tembak} > 1$). Kecepatan peluru adalah $20 - 3 * \text{daya tembak}$. Penembakan menghasilkan panas senjata, dan senjata tidak dapat menembak lagi sampai mendingin. Pilihan daya tembak melibatkan pertukaran antara kerusakan, kecepatan peluru, dan panas senjata. Bot greedy membutuhkan heuristik untuk menentukan daya tembak yang sesuai berdasarkan situasi (misalnya, jarak ke target, energi musuh).

Bot biasanya diprogram menggunakan Java atau .NET, memanfaatkan Bot API yang disediakan untuk komunikasi dengan server game melalui WebSockets. Logika inti biasanya diimplementasikan dalam metode `run()`, yang dieksekusi secara terus menerus. Bot bereaksi terhadap peristiwa game dengan menimpa metode penanganan peristiwa seperti *OnScannedBot*, *OnHitByBullet*, dll.. Arsitektur berbasis peristiwa mengharuskan bot greedy untuk membuat keputusan berdasarkan informasi terbaru yang diterima melalui peristiwa dan untuk mengeksekusi tindakan dalam batas waktu giliran. Logika yang kompleks harus dikelola dengan hati-hati untuk menghindari kehilangan giliran.

Bot greedy akan terus menerus merasakan lingkungannya (terutama melalui pemindaian) dan, berdasarkan heuristik yang ditentukan, memilih tindakan yang tampak paling menguntungkan pada saat itu. Proses pengambilan keputusan ini kemungkinan besar akan terjadi dalam loop `run()` atau dalam penanganan peristiwa. Tindakan yang dipilih (pergerakan, putaran, penembakan) kemudian akan dieksekusi menggunakan metode yang disediakan oleh Bot API. Kunci untuk mengimplementasikan algoritma greedy terletak pada pendefinisian heuristik yang jelas dan efektif yang memandu pilihan lokal bot dalam setiap situasi. Kode bot yang disediakan menawarkan contoh heuristik yang berbeda.

Pastikan Java 11 atau yang lebih baru telah terinstal. Unduh aplikasi GUI Robocode Tank Royale dari halaman rilis di GitHub. Buat atau unduh kode bot (misalnya, bot sampel) dan tempatkan di direktori yang sesuai. Luncurkan aplikasi GUI (misalnya, melalui baris perintah: `java -jar robocode-tankroyale-gui-x.y.z.jar`). Konfigurasi GUI untuk mengenali direktori bot (Config -> Bot Root Directories). Buat pertempuran baru (Battle -> Start Battle), pilih bot yang diinginkan (termasuk bot greedy Anda dan lawan), dan mulai pertempuran. Keakraban dengan menjalankan pertempuran Robocode sangat penting untuk menguji dan mengamati perilaku strategi greedy yang diimplementasikan.

Bab 3: Aplikasi Strategi Greedy

3.1 Persoalan Robocode Dalam Elemen Algoritma Greedy

Elemen-Elemen Algoritma Greedy:

- **Himpunan Kandidat:** Kumpulan tindakan yang mungkin dipertimbangkan oleh bot pada giliran tertentu. Ini dapat mencakup:
 - Berbagai kemungkinan pergerakan (misalnya, bergerak maju dengan jarak tertentu, bergerak mundur, berputar ke kiri dengan sudut tertentu, berputar ke kanan). Tingkat granularitas opsi ini akan mempengaruhi kompleksitas dan potensi efektivitas.
 - Tingkat daya tembak yang berbeda untuk digunakan saat menembakkan senjata (misalnya, minimum, maksimum, atau nilai berdasarkan jarak).
 - Tindakan terkait kontrol radar (misalnya, putar radar ke kiri, putar radar ke kanan dengan kenaikan tertentu). Pilihan granularitas untuk himpunan kandidat melibatkan pertukaran. Granularitas yang lebih halus memungkinkan kontrol yang lebih tepat tetapi meningkatkan ruang pencarian untuk pilihan greedy.
- **Himpunan Solusi:** Urutan lengkap tindakan yang diambil oleh bot selama pertempuran atau sebagian besar pertempuran. "Solusi" dalam konteks ini adalah semua keputusan perilaku robot yang mungkin dan hasil interaksi bot dengan lingkungan dan bot lain.
- **Fungsi Solusi:** Fungsi (implisit atau eksplisit dalam logika bot) yang menentukan apakah keadaan saat ini atau urutan tindakan itu "baik" atau mengarah pada hasil yang diinginkan (misalnya, bertahan hidup, menimbulkan kerusakan, memenangkan ronde).
- **Fungsi Seleksi:** Inti dari algoritma greedy. Ini adalah heuristik yang digunakan bot untuk mengevaluasi tindakan kandidat yang tersedia dan memilih tindakan yang tampak terbaik pada saat itu. Contoh dalam Robocode dapat berupa:
 - Memilih musuh terdekat untuk diserang.
 - Memilih sudut tembak yang memaksimalkan probabilitas langsung mengenai musuh yang dipindai.
 - Memilih arah gerakan yang memaksimalkan jarak dari ancaman terdekat. Efektivitas seluruh strategi greedy bergantung pada desain fungsi seleksi (heuristik) ini. Heuristik ini perlu selaras dengan tujuan keseluruhan.
- **Fungsi Kelayakan:** Batasan yang diberlakukan oleh aturan permainan dan keadaan bot saat ini yang menentukan tindakan kandidat mana yang diizinkan. Ini termasuk:
 - Tingkat energi bot saat ini (tidak dapat menembak jika energi tidak mencukupi).
 - Panas senjata saat ini (tidak dapat menembak jika terlalu panas).
 - Batasan fisik pergerakan (kecepatan maksimum, tingkat putaran).
- **Fungsi Objektif:** Tujuan keseluruhan yang ingin dicapai oleh bot. Dalam Robocode Tank Royale, ini biasanya memaksimalkan skor, yang dipengaruhi oleh kelangsungan hidup, kerusakan yang ditimbulkan, dan faktor-faktor lain. Namun, algoritma greedy membuat keputusan berdasarkan tujuan lokal yang lebih langsung

yang diturunkan dari tujuan keseluruhan ini.

3.2 Eksplorasi Alternatif Solusi Greedy

Alternatif 1: Penghindar, Tembakan Prediktif (Analisis Bot "Sniper"):

- **Heuristik:** Bergerak di sepanjang dinding arena. Ketika musuh dipindai, hentikan pergerakan, prediksi posisi masa depannya berdasarkan kecepatan dan arah saat ini, dan tembak ke lokasi yang diprediksi.
- **Himpunan Kandidat:** Tindakan: Bergerak di sepanjang dinding, berhenti, memprediksi posisi musuh, membidik dan menembak.
- **Himpunan Solusi:** Himpunan solusi terdiri dari semua penembakan yang diambil jika telah terdapat robot musuh pada scanner
- **Fungsi Seleksi:** Terus bergerak di sepanjang dinding. Setelah memindai musuh, prediksi lokasinya dan tembak. Strategi ini menggabungkan bentuk prediksi, membuat pilihan "terbaik" berdasarkan perkiraan langsung dari keadaan masa depan.
- **Fungsi Kelayakan:** Tembakan dilakukan jika robot memiliki energi yang cukup, dan ketika menembak robot berhenti agar tembakan lebih akurat
- **Fungsi Objektif:** Memaksimalkan perolehan poin pada game

Alternatif 2: Pemburu Bot Lemah (Analisis Bot "AgroRammer"):

- **Heuristik:** Bergerak berputar-putar sampai menemukan bot dengan energi lebih kecil. Jika menemukan bot dengan energi yang lebih kecil, maka tabrak bot tersebut.
- **Himpunan Kandidat:** Semua bot pada arena.
- **Himpunan Solusi:** Semua bot yang terpilih.
- **Fungsi Solusi:** Memeriksa apakah masih ada bot pada arena.
- **Fungsi Seleksi:** Pilih bot dengan jumlah energi yang kurang dari energi bot yang kita miliki.
- **Fungsi Kelayakan:** Memastikan tabrakan aman dan tidak membahayakan bot.
- **Fungsi Objektif:** Memaksimalkan perolehan poin pada game.

Alternatif 3: Pengemudi agresif (Analisis Bot "Drift"):

- **Heuristic:** Bot bergerak terus ke depan kanan sejauh mungkin sambil berbelok jika berada di dekat batas arena, menembak setiap kali mendeteksi musuh, dan menyesuaikan arah jika bertabrakan dengan lawan atau dinding.
- **Himpunan Kandidat:** Kandidat tindakan yang dapat diambil bot mencakup Navigasi: Bergerak maju dalam garis lurus, berbelok saat mencapai batas arena, atau berbelok jika bertabrakan; Serangan: Menembak dengan kekuatan tertentu berdasarkan deteksi musuh dan posisi relatifnya; Reaksi terhadap tabrakan: Menyesuaikan arah dengan sedikit berbelok jika bertabrakan dengan bot lain atau dinding.

- **Himpunan Solusi:** Himpunan solusi terdiri dari semua jalur gerakan dan keputusan serangan yang diambil selama pertandingan. Solusi yang optimal adalah strategi yang mempertahankan mobilitas tinggi, menghindari dinding dan musuh secara efektif, serta memberikan jumlah tembakan yang maksimal dengan akurasi tinggi.
- **Fungsi Seleksi:** Jika bot berada dalam batas arena, ia terus bergerak maju, tetapi jika mendekati batas arena, ia berbelok sedikit untuk menghindari tabrakan. Saat mendeteksi musuh, bot langsung menembak dengan kekuatan penuh, sementara jika bertabrakan dengan musuh, ia menembak jika lawan berada dalam sudut kecil di depannya dan berbelok untuk menghindari benturan lebih lanjut. Jika menabrak dinding, bot segera berbelok dan kembali menjalankan strategi awalnya.
- **Fungsi Kelayakan:** Aksi navigasi hanya dilakukan jika bot masih dalam arena, sementara tembakan hanya dilakukan jika bot memiliki cukup energi. Selain itu, aksi berbelok dilakukan jika bot berada dekat dengan dinding atau bertabrakan dengan musuh.
- **Fungsi objektif:** Perolehan poin pada permainan

Alternatif 4: Tabrak dan Lari (Analisis bot “HitnRun”):

- **Heuristic:** Bot akan terus memindai lapangan permainan untuk musuh, jika ditemukan bot musuh dengan energi yang lebih rendah, maka bot akan menyerang. Jika ditemukan bot musuh dengan energi yang lebih tinggi, maka bot akan kabur
- **Himpunan Kandidat:** Himpunan kandidat berisi semua keputusan gerakan yang dapat diambil oleh bot, yaitu menyerang jika menemukan musuh dengan energi rendah, dan kabur jika menemukan musuh dengan energi yang lebih tinggi.
- **Himpunan Solusi:** Himpunan solusi berisi robot apa saja yang dipindai, lalu aksi yang diambil apakah menyerang atau kabur
- **Fungsi Seleksi:** Bandingkan energi bot musuh yang dipindai dengan energi sendiri, jika energi bot musuh lebih kecil, maka menyerang bot musuh dengan cara ram/mendekatinya, jika energi bot musuh lebih besar, maka bergerak ke arah lawannya.
- **Fungsi Kelayakan:** Memastikan bahwa bot tidak akan mati ketika menyerang karena bot yang diserang adalah bot dengan energi yang lebih rendah
- **Fungsi Objektif:** Memaksimalkan poin pada permainan

Bab 4: Implementasi dan Pengujian

4.1 Implementasi Solusi Greedy

Berikut adalah pseudocode dari implementasi masing-masing alternatif solusi greedy

4.1.1 Robot “Sniper”

```
ALGORITMA Sniper
Deklarasi:
peek : boolean
(Menunjukkan apakah bot sedang dalam mode "mengintip")
moveAmount : real
(Jarak untuk bergerak sepanjang dinding, diatur ke dimensi arena maksimum)
botInfo : BotInfo
(Konfigurasi bot yang di-load dari file "Sniper.json")

Fungsi Utama: Main()
Fungsi Main():
    Buat instance Sniper dengan konfigurasi dari file "Sniper.json"
    Panggil method Start() untuk memulai bot

Fungsi Konstruktor: Sniper(botInfo)
Fungsi Sniper(botInfo: BotInfo):
    Inisialisasi bot dengan botInfo

Fungsi Run():
    // Atur warna untuk identifikasi visual
    Set BodyColor menjadi hitam
    Set TurretColor menjadi hitam
    Set RadarColor menjadi oranye
    Set BulletColor menjadi cyan
    Set ScanColor menjadi cyan

    // Atur jarak gerak sepanjang dinding
    moveAmount ← nilai maksimum antara ArenaWidth dan ArenaHeight
    peek ← false

    // Awal: Berputar untuk menghadap dinding
    Putar ke kanan sebesar (Direction mod 90) derajat
    Majukan bot ke depan sejauh moveAmount unit
```

```

// Persiapan mengintip dan patroli dinding
peek ← true
Putar senjata (turret) ke kanan sebesar 90 derajat
Putar bot ke kanan sebesar 90 derajat

// Loop utama: Patroli di perimeter arena
Selama IsRunning bernilai true:
    peek ← true    // Aktifkan mode mengintip
    Majukan bot ke depan sejauh moveAmount unit
    peek ← false   // Nonaktifkan mode mengintip
    Putar bot ke kanan sebesar 90 derajat

Fungsi OnScannedBot(event: ScannedBotEvent)
Fungsi OnScannedBot(event):
    // Hentikan gerakan untuk memungkinkan tembakan yang
    akurat
    Stop()

    // --- Perhitungan Tembakan Prediksi ---
    bulletPower ← 2
    bulletSpeed ← 20 - 3 * bulletPower    // Rumus kecepatan
    peluru

    // Hitung jarak ke musuh
    distance ← DistanceTo(event.X, event.Y)

    // Perkirakan waktu yang diperlukan peluru untuk mencapai
    target
    time ← distance / bulletSpeed

    // Prediksi posisi musuh di masa depan (asumsi kecepatan
    dan arah konstan)
    enemyDirectionRad ← ToRadians(event.Direction)
    predictedX ← event.X + event.Speed *
    Cos(enemyDirectionRad) * time
    predictedY ← event.Y + event.Speed *
    Sin(enemyDirectionRad) * time

    // Tentukan sudut dari posisi bot ke posisi prediksi
    musuh
    angleToPredicted ← Atan2(predictedY - Y, predictedX - X)
    * (180 / π)

```

```

    // Hitung sudut minimal untuk memutar senjata berdasarkan
posisi saat ini
    gunTurn ← NormalizeAngle(angleToPredicted - GunDirection)
    Putar senjata (turret) ke kanan sebesar gunTurn derajat

    // Tembakkan peluru dengan kekuatan bulletPower
    Fire(bulletPower)

    // Jika mode mengintip aktif, lakukan pemindaian ulang
segera
    Jika peek bernilai true:
        Rescan()

    // Lanjutkan patroli dengan melanjutkan gerakan
    Resume()

Fungsi OnHitBot(event: HitBotEvent)
Fungsi OnHitBot(event):
    bearing ← BearingTo(event.X, event.Y)
    Jika (bearing > -90) dan (bearing < 90) maka:
        Mundur sejauh 100 unit
    Jika tidak:
        Majukan ke depan sejauh 100 unit

Fungsi Pembantu: ToRadians(degrees)
Fungsi ToRadians(degrees):
    Kembalikan degrees *  $\pi$  / 180.0

Fungsi Pembantu: NormalizeAngle(angle)
Fungsi NormalizeAngle(angle):
    Selama angle > 180:
        Kurangi angle dengan 360
    Selama angle < -180:
        Tambahkan angle dengan 360
    Kembalikan angle

```

4.1.2 Robot “AgroRammer”

```

ALGORITMA AgroRammer
Deklarasi:
    botInfo : BotInfo

```


Deskripsi:

```
// Muat konfigurasi dari file "AgroRammer.json"
bot.Start()
```

Fungsi AgroRammer(botInfo : BotInfo)

```
Inisialisasi bot dengan informasi botInfo
Set warna tubuh, turret, radar, peluru, jejak, dan
senjata
```

```
Selama bot masih berjalan (IsRunning):
    Putar turret ke kanan (10.000 derajat)
    Majukan bot ke depan (10.000 unit)
```

Fungsi OnScannedBot(event : ScannedBotEvent)

```
Jika energi musuh < energi bot maka
    Hitung arah ke musuh
    Putar bot ke arah musuh
    Maju ke arah musuh untuk menabrak
    Tembak musuh dengan kekuatan optimal
Jika tidak
    Hitung arah turret ke musuh
    Tembak musuh dengan kekuatan optimal
```

Fungsi OnHitByBullet(event : HitBulleBytEvent)

```
Putar ke kanan sebesar 90 derajat
Maju ke depan sebesar 120 unit
```

Fungsi OnHitWall(Event : HitWallEvent)

```
Putar ke kanan sebesar 90 derajat
Maju ke depan sebesar 120 unit
```

Fungsi OnHitBot(event : HitBotEvent)

```
Jika musuh yang menabrak kita maka
    Jika musuh memiliki energi < 10 maka
        Maju ke arah musuh
        Tembak dengan kekuatan maksimal
    Jika tidak
        Putar ke kanan
        Mundur untuk menghindari serangan
Jika tidak // Jika kita yang menabrak musuh
    Mundur sedikit
    Tembak dengan kekuatan sedang
```

4.1.3 Robot “Drift:

ALGORITMA DriftBot

Deklarasi:

botInfo : BotInfo

Deskripsi:

// Muat konfigurasi dari file "Drift.json"

bot.Start()

Fungsi Drift(botInfo : BotInfo)

Inisialisasi bot dengan informasi botInfo

Set warna tubuh, turret, radar, peluru, jejak, dan senjata

Selama bot masih berjalan (IsRunning):

Putar turret ke kanan (10.000 derajat)

Majukan bot ke depan (10.000 unit)

Jika bot mendekati batas area ($X < 100$, $X > 700$, $Y < 100$, $Y > 500$):

Putar sedikit ke kanan (10 derajat)

Fungsi OnScannedBot(event : ScannedBotEvent)

Tembak dengan kekuatan 3

Fungsi OnHitBot(event : HitBotEvent)

Hitung bearing (arah) antara bot dan tembak jika bearing < 10

Jika bot ditabrak oleh bot lain (rammed):

Putar turret ke kanan (10 derajat)

Fungsi OnHitWall(event : HitWallEvent)

Jika bot menabrak dinding:

Putar turret 20 derajat

Panggil metode Run untuk melanjutkan pergerakan bot

4.1.4 Robot "HitnRun"

ALGORITMA HitnRun

Deklarasi:

turnDirection: integer

(Nilai 1: berlawanan arah jarum jam; -1: searah jarum jam)

botInfo: BotInfo

Fungsi Main():

Buat instance HitnRun dengan konfigurasi dari file "HitnRun.json"

Panggil method Start() untuk memulai bot

Fungsi HitnRun(botInfo: BotInfo):
 Inisialisasi bot dengan botInfo
 Set turnDirection = 1

Fungsi Run():
 Set BodyColor menjadi abu-abu terang (contoh: RGB(0x99, 0x99, 0x99))
 Set TurretColor menjadi abu-abu (contoh: RGB(0x88, 0x88, 0x88))
 Set RadarColor menjadi abu-abu gelap (contoh: RGB(0x66, 0x66, 0x66))

 Selama bot masih berjalan (IsRunning):
 Putar bot ke kiri sebesar (5 * turnDirection) derajat

Fungsi OnScannedBot(event):
 Jika (event.Energy <= Energy) maka:
 // Strategi menyerang:
 Panggil TurnToFaceTarget(event.X, event.Y)
 distance ← DistanceTo(event.X, event.Y)
 Majukan bot ke depan sejauh (distance + 5) unit
 Jika tidak:
 // Strategi menghindar:
 Panggil TurnAway(event.X, event.Y)
 distance ← DistanceTo(event.X, event.Y)
 Majukan bot ke depan sejauh (distance + 5) unit

 Panggil Rescan() untuk melanjutkan pemindaian bot lain

Fungsi OnHitBot(event):
 Mundur sejauh 40 unit

Fungsi TurnToFaceTarget(x, y):
 bearing ← BearingTo(x, y)
 Jika (bearing >= 0) maka:
 Set turnDirection = 1
 Jika tidak:
 Set turnDirection = -1
 Putar bot ke kiri sebesar bearing derajat

```

Fungsi TurnAway(x, y):
    bearing ← BearingTo(x, y)
    Putar bot ke kiri sebesar (bearing + 180) derajat

```

Dari keempat alternatif solusi greedy yang kami buat, kami memutuskan untuk menggunakan bot **AgroRammer** sebagai bot perwakilan kelompok kami (penjelasan lebih lanjut tercantum pada subbab 4.2 Pengujian), berikut adalah analisis lebih lanjut dari robot “AgroRammer”.

Penjelasan struktur data, fungsi, dan prosedur yang digunakan pada bot dengan solusi greedy ArgoRammer

Struktur Data

Struktur data yang digunakan dalam bot ini terdiri dari variabel global yang menyimpan status bot serta informasi musuh.

- movingForward (boolean) → Menyimpan status apakah bot sedang bergerak maju.
- BodyColor, TurretColor, RadarColor, BulletColor, ScanColor, GunColor → Variabel untuk mengatur warna bot, hanya berfungsi sebagai estetika.

Struktur data ini digunakan untuk menentukan pergerakan bot dan bagaimana bot merespons lingkungan sekitarnya (misalnya, musuh, tembakan, atau dinding).

Prosedur Run()

Prosedur ini bertindak sebagai loop utama bot, yang terus berjalan selama bot masih hidup. Prosedur ini bertujuan untuk mengatur pola pergerakan bot agar terus berputar ke kiri dalam lingkaran besar

Penjelasan:

- SetTurnLeft(10_000); berfungsi untuk Memerintahkan bot untuk terus berputar ke kiri.
- MaxSpeed = 5; berfungsi untuk membatasi kecepatan maksimum bot untuk mengontrol pergerakan lebih baik.
- Forward(10_000); berfungsi Memerintahkan bot untuk terus maju dalam jarak besar sambil berputar.

Prosedur OnScannedBot(e)

Prosedur ini dipanggil setiap kali bot mendeteksi musuh melalui radar. Prosedur ini bertujuan untuk menyerang musuh dengan strategi yang disesuaikan dengan kondisinya.

Penjelasan:

- Jika energi musuh lebih kecil dari bot, bot akan menabraknya untuk memberikan damage tambahan.
- Jika musuh lebih kuat, bot hanya akan menembak dari jarak aman.
- Tembakan menggunakan $\text{Math.Min}(3 - \text{Math.Abs}(\text{bearingFromGun}), \text{Energy} - .1)$, yang memastikan daya tembakan tetap optimal sesuai dengan kondisi energi bot.

Prosedur OnHitByBullet(e)

Prosedur ini dijalankan saat bot terkena tembakan dari musuh. Prosedur ini bertujuan untuk Menghindari serangan musuh dengan segera bergerak ke posisi yang lebih aman.

Penjelasan:

- SetTurnRight(90); Berfungsi untuk memerintahkan bot akan berbelok 90° ke kanan untuk mengubah jalur gerakan.

- SetForward(120); → Berfungsi untuk memerintahkan bot akan maju sejauh 120 unit agar lebih sulit ditembak lagi.

Prosedur OnHitWall(e)

Prosedur ini dijalankan saat bot menabrak dinding di arena. Prosedur ini bertujuan untuk menghindari bot terjebak di sudut arena.

Penjelasan:

- SetTurnRight(90); Berfungsi untuk memerintahkan bot akan berbelok ke kanan untuk menjauhi dinding.
- SetForward(150); Berfungsi untuk memerintahkan bot akan bergerak menjauh dari dinding agar tetap aktif di arena.

Prosedur OnHitBot(e)

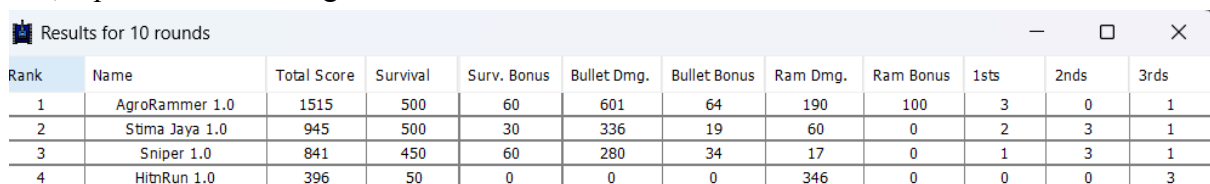
Prosedur ini dipanggil saat bot menabrak atau ditabrak oleh bot lain. Prosedur ini bertujuan untuk menyesuaikan strategi berdasarkan kondisi musuh setelah tabrakan.

Penjelasan:

- Jika musuh lebih lemah ($\text{energi} < 10$), bot akan terus menekan dan menembak dengan kekuatan penuh.
- Jika musuh lebih kuat, bot akan menghindar dengan mundur dan berbelok.
- Jika bot yang menabrak duluan, ia akan mundur sedikit lalu menembak agar tetap dalam posisi menyerang.

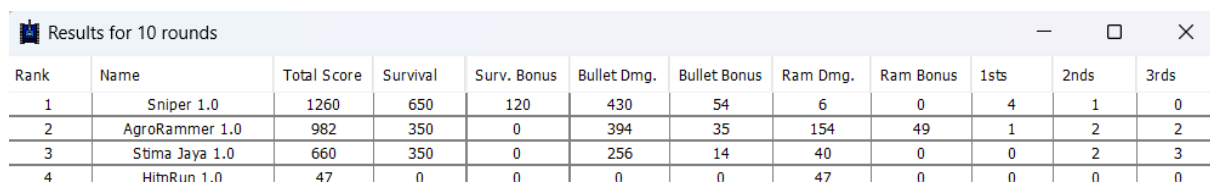
4.2 Pengujian

Untuk menentukan mana dari keempat alternatif solusi yang akan dijadikan bot utama dalam kompetisi, keempat alternatif solusi yang telah dibuat ditandingkan satu sama lain dalam mode *battle royale* sebanyak lima kali. Untuk setiap iterasi *battle royale*, diberlakukan sistem *scoring* seperti pada kontestasi balap Formula One “F1”, yaitu peringkat pertama mendapatkan poin terbesar (dalam hal ini, 4 poin karena dari 4 robot), diikuti peringkat kedua mendapatkan 2 poin, dan seterusnya. Setelah menjalankan simulasi *battle royale* sebanyak 5 kali, diperoleh hasil sebagai berikut:



Rank	Name	Total Score	Survival	Surv. Bonus	Bullet Dmg.	Bullet Bonus	Ram Dmg.	Ram Bonus	1sts	2nds	3rds
1	AgroRammer 1.0	1515	500	60	601	64	190	100	3	0	1
2	Stima Jaya 1.0	945	500	30	336	19	60	0	2	3	1
3	Sniper 1.0	841	450	60	280	34	17	0	1	3	1
4	HitnRun 1.0	396	50	0	0	0	346	0	0	0	3

Gambar 4. Hasil Simulasi Pertama



Rank	Name	Total Score	Survival	Surv. Bonus	Bullet Dmg.	Bullet Bonus	Ram Dmg.	Ram Bonus	1sts	2nds	3rds
1	Sniper 1.0	1260	650	120	430	54	6	0	4	1	0
2	AgroRammer 1.0	982	350	0	394	35	154	49	1	2	2
3	Stima Jaya 1.0	660	350	0	256	14	40	0	0	2	3
4	HitnRun 1.0	47	0	0	0	0	47	0	0	0	0

Gambar 5. Hasil Simulasi Kedua

Rank	Name	Total Score	Survival	Surv. Bonus	Bullet Dmg.	Bullet Bonus	Ram Dmg.	Ram Bonus	1sts	2nds	3rds
1	Stima Jaya 1.0	1070	450	60	480	45	35	0	2	1	1
2	AgroRammer 1.0	767	300	0	349	39	58	21	1	2	1
3	Sniper 1.0	697	350	30	280	12	25	0	1	1	1
4	HitnRun 1.0	82	50	0	0	0	32	0	0	0	1

Gambar 6. Hasil Simulasi Ketiga

Rank	Name	Total Score	Survival	Surv. Bonus	Bullet Dmg.	Bullet Bonus	Ram Dmg.	Ram Bonus	1sts	2nds	3rds
1	AgroRammer 1.0	1073	400	30	380	49	172	42	2	1	1
2	Stima Jaya 1.0	794	400	30	320	10	34	0	1	2	1
3	Sniper 1.0	612	300	30	260	10	12	0	1	1	1
4	HitnRun 1.0	144	50	0	0	0	94	0	0	0	1

Gambar 7. Hasil Simulasi Keempat

Rank	Name	Total Score	Survival	Surv. Bonus	Bullet Dmg.	Bullet Bonus	Ram Dmg.	Ram Bonus	1sts	2nds	3rds
1	AgroRammer 1.0	1046	350	30	452	24	143	46	2	0	2
2	Stima Jaya 1.0	742	400	30	256	16	40	0	1	2	1
3	Sniper 1.0	615	350	30	210	14	11	0	1	2	0
4	HitnRun 1.0	164	50	0	0	0	114	0	0	0	1

Gambar 8. Hasil Simulasi Kelima

Maka, Berdasarkan hasil simulasi tersebut, diperoleh perolehan poin sebagai berikut

Robot	Sim I	Sim II	Sim III	Sim IV	Sim V	Total
AgroRammer	4	3	3	4	4	18
Stima Jaya	3	2	4	3	3	15
Sniper	2	4	2	2	2	12
HitnRun	1	1	1	1	1	5

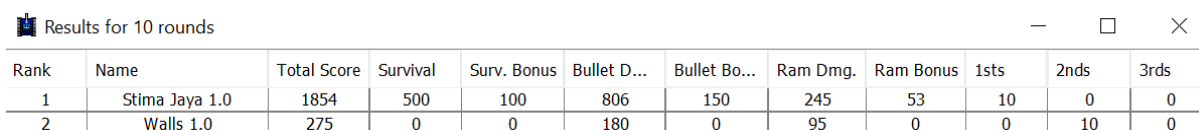
Sehingga berdasarkan hasil perolehan poin, dipilih bot **AgroRammer** sebagai bot perwakilan kelompok kami. Selanjutnya untuk menguji performa dari bot yang kami pilih, kami melakukan pengujian 1v1 dengan beberapa template robot yang diberikan. Dari 3 kali pengujian didapatkan hasil sebagai berikut:

Rank	Name	Total Score	Survival	Surv. Bonus	Bullet D...	Bullet Bo...	Ram Dmg.	Ram Bonus	1sts	2nds	3rds
1	Stima Jaya 1.0	2188	500	100	957	116	349	165	10	0	0
2	Corners 1.0	419	0	0	418	0	1	0	0	10	0

Gambar 9. Hasil Simulasi Keenam

Rank	Name	Total Score	Survival	Surv. Bonus	Bullet D...	Bullet Bo...	Ram Dmg.	Ram Bonus	1sts	2nds	3rds
1	Stima Jaya 1.0	1096	250	50	535	108	152	0	5	4	0
2	Spin Bot 1.0	680	150	30	384	40	76	0	4	5	0

Gambar 10. Hasil Simulasi Ketujuh



Rank	Name	Total Score	Survival	Surv. Bonus	Bullet D...	Bullet Bo...	Ram Dmg.	Ram Bonus	1sts	2nds	3rds
1	Stima Jaya 1.0	1854	500	100	806	150	245	53	10	0	0
2	Walls 1.0	275	0	0	180	0	95	0	0	10	0

Gambar 11. Hasil Simulasi Kedelapan

Terbukti bahwa strategi dari bot AgroRammer berhasil untuk mengalahkan bot lainnya. Dari simulasi keenam dan kedelapan, AgroRammer berhasil untuk menang di setiap babak. Namun, saat melawan Spin Bot, AgroRammer hanya menang sebanyak 5 kali. Hal ini terjadi karena Spin Bot berhasil menembak AgroRammer sehingga mendapatkan poin Bullet Damage yang cukup besar.

Bab 5: Kesimpulan dan Saran

5.1 Kesimpulan

Berdasarkan hasil implementasi dan pengujian, dapat disimpulkan bahwa penerapan algoritma greedy dalam pembuatan bot pada permainan Robocode Tank Royale memberikan hasil yang memuaskan. Empat solusi greedy dengan heuristic yang berbeda telah berhasil diimplementasikan dan dievaluasi melalui serangkaian pengujian, dan dipilih bot dengan performa terbaik yaitu bot “AgroRammer”. Meskipun solusi berbasis algoritma greedy menunjukkan performa yang baik dan efisiensi waktu komputasi, terdapat beberapa keterbatasan, seperti ketidakmampuan bot untuk beradaptasi dengan perubahan strategi lawan secara optimal dan keterbatasan dalam memprediksi pergerakan musuh pada situasi kompleks. Hal ini mengindikasikan bahwa, meskipun algoritma greedy efektif dalam konteks tertentu, pengembangan strategi yang lebih adaptif mungkin diperlukan untuk menghadapi skenario permainan yang lebih bervariasi.

5.2 Saran

1. Menentukan heuristic yang lebih optimal
2. Menggabungkan beberapa pendekatan ke dalam satu bot
3. Melakukan pengujian yang lebih ekstensif
4. Saat melawan bot dengan strategi tembakan seperti Spin Bot, AgroRammer mengalami kesulitan. Dapat dipertimbangkan untuk menambahkan strategi penghindaran tembakan untuk mengurangi damage yang diterima.
5. strategi penyerangan dapat dioptimalkan dengan menyesuaikan pola gerak agar lebih agresif dalam mendekati lawan sehingga dapat memaksimalkan *ram bonus*.

Lampiran

Tautan Github: https://github.com/qodriazka/Tubes1_Stima-Jaya.git

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

Daftar Pustaka

GitHub Pages. (n.d.). *Robocode Tank Royale Docs*. Diakses pada 24 Maret 2025, dari <https://robocode-dev.github.io/tank-royale/>

YouTube. (n.d.). *The Greedy Algorithm: A Quick Overview*. Diakses pada 24 Maret 2025, dari <https://www.youtube.com/watch?v=UoRLg-rfb2U>

NuGet Gallery. (n.d.). *Robocode.TankRoyale.BotApi 0.30.0*. Diakses pada 24 Maret 2025, dari <https://www.nuget.org/packages/Robocode.TankRoyale.BotApi>

SourceForge. (n.d.). *Robot (Robocode 1.9.5.0 API)*. Diakses pada 24 Maret 2025, dari <https://robocode.sourceforge.io/docs/robocode/robocode/Robot.html>

Random Article. (n.d.). *Robocode Lesson: Basic Targeting*. Diakses pada 24 Maret 2025, dari http://mark.random-article.com/robocode/basic_targeting.html

SourceForge. (n.d.). *ScannedRobotEvent (Robocode 1.9.5.0 API)*. Diakses pada 24 Maret 2025, dari <https://robocode.sourceforge.io/docs/robocode/robocode/ScannedRobotEvent.html>

GitHub. (n.d.). *talk on the Walls bot's 'wrong' behaviour (with blocking API calls) · robocode-dev tank-royale · Discussion #80*. Diakses pada 24 Maret 2025, dari <https://github.com/robocode-dev/tank-royale/discussions/80>

Chalmers University. (n.d.). *Wall Avoidance - RoboWiki*. Diakses pada 24 Maret 2025, dari https://www.cse.chalmers.se/~bergert/robowiki-mirror/RoboWiki/robowiki.net/wiki/Wall_Avoidance.html

Stack Overflow. (n.d.). *Greedy Algorithm: The Robot*. Diakses pada 24 Maret 2025, dari <https://stackoverflow.com/questions/40577833/greedy-algorithm-the-robot>

YouTube. (n.d.). *RoboCode Tutorial 8 - Targeting Part 1*. Diakses pada 24 Maret 2025, dari <https://www.youtube.com/watch?v=75sU-AJxZm0>

GitLab. (n.d.). *X-Mas Lecture: Robocode - Matworx*. Diakses pada 24 Maret 2025, dari <https://gitlabio.z6.web.core.windows.net/inf-fpk/assets/15-vorlesung/15-vorlesung.pdf>

ETHGlobal. (n.d.). *Greedy Bots*. Diakses pada 24 Maret 2025, dari <https://ethglobal.com/showcase/greedy-bots-sou6t>

GitHub Gist. (n.d.). *A simple yet extremely effective Robocode robot*. Diakses pada 24 Maret 2025, dari <https://gist.github.com/rogerluan/d62a26039d9bcb9395f5d391fb1a17ae>