# Laporan Tugas Besar 01

IF2211 Strategi Algoritma


Disusun oleh:

Kelompok 38 - ScoobyDude

| | |
|---|---|
| Danendra Shafi Athallah | (13523136) |
| Jovandra Otniel Pangalambok Siregar | (13523141) |
| Ardell Aghna Mahendra | (13523151) |

**PROGRAM STUDI TEKNIK INFORMATIKA**
**SEKOLAH TEKNIK ELEKTRO DAN INFORMATIKA**
**INSTITUT TEKNOLOGI BANDUNG**
**JL. GANESA 10, BANDUNG 40132**


**2025**

# BAB I
# DESKRIPSI TUGAS

## 1.1 Deskripsi Permainan Robocode TankRoyale

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

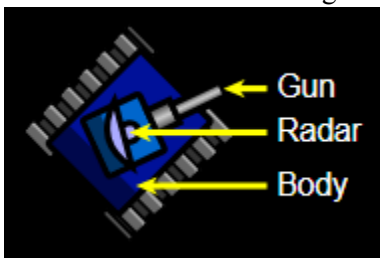
bot tidak bisa mengambil waktu sebanyak yang mereka inginkan untuk bergerak dan menyelesaikan turn saat ini. Setiap kali turn baru dimulai, penghitung waktu ulang diatur ulang dan mulai berjalan. Jika batas waktu tercapai dan bot tidak mengirimkan pergerakannya untuk turn tersebut, maka tidak ada perintah 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:
   
   

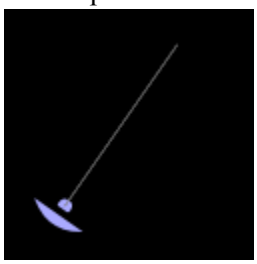
*Body* adalah bagian utama dari tank yang digunakan untuk menggerakkan tank. *Gun* digunakan untuk menembakkan peluru dan dapat berputar bersama *body* atau independen dari *body*. *Radar* digunakan untuk memindai posisi musuh dan dapat berputar bersama *body* atau independen dari *body*.

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.

## 1.2 Tujuan

1. Menerapkan Algoritma Greedy pada bot tank di permainan Robocode Tank Royale.
2. Meningkatkan peluang kemenangan bot tank pada permainan kompetisi Robocode Tank Royale.
3. Merancang Bot Tank pada permainan Robocode Tank Royale.

## 1.3 Spesifikasi Program

Pada tugas ini, kami mengimplementasikan *Greedy Algorithms* dalam permainan Robocode Tank Royale yang merupakan sebuah permainan pemrograman di mana bot berbentuk tank virtual bertarung di arena untuk menjadi yang terakhir bertahan, mirip dengan konsep permainan *Battle Royale*. Robocode Tank Royale ditulis dalam bahasa Java dan Kotlin, sedangkan *tank bot* yang menjadi player ditulis dalam bahasa C#.

Program yang kami kembangkan terdiri dari empat bot tank yang berbeda, di mana masing-masing diimplementasikan dengan strategi greedy yang unik dan menggunakan heuristik yang berbeda-beda. Setiap bot dibuat menggunakan bahasa pemrograman C# (.NET) dan mengimplementasikan API yang disediakan oleh Robocode Tank Royale. Dalam implementasinya, masing-masing bot memiliki kemampuan untuk bergerak secara otonom, melacak posisi musuh, menembak, dan mengambil keputusan berdasarkan kondisi pertempuran yang sedang berlangsung.

Bot utama yang kami kembangkan, ScoobyGodBot, menerapkan strategi greedy yang lebih kompleks dan dioptimasi untuk performa yang lebih baik di berbagai skenario pertempuran, sementara tiga bot alternatif (ScoobyLuicy, ScoobyShadowBot, dan ScoobyRoma3000) menggunakan pendekatan greedy dengan variasi yang berbeda untuk mengeksplorasi berbagai kemungkinan strategi. Semua bot didesain untuk beradaptasi dengan kondisi arena dan pola pertempuran, dengan tujuan memaksimalkan peluang kelangsungan hidup dan perolehan skor tertinggi.

# BAB II
# LANDASAN TEORI

## 2.1 Algoritma Greedy

Algoritma *greedy* adalah paradigma penyelesaian masalah yang mengutamakan pilihan optimum lokal pada setiap langkah dengan harapan mencapai solusi optimum global pada akhir proses (journalofinequalitiesandapplications.springeropen.com, ijsrp.org). Pada setiap iterasi, algoritma greedy memilih tindakan yang *paling menjanjikan* atau memberikan keuntungan terbesar saat itu, tanpa mempertimbangkan konsekuensi jangka panjang keputusan tersebut. Prinsip kerjanya bersifat *myopic*, yakni berfokus pada optimalitas langkah saat ini, dan algoritma ini tidak melakukan penelusuran mundur (backtracking) untuk mengoreksi keputusan sebelumnya. Konsekuensinya, meskipun seringkali *efisien dan sederhana* dalam implementasi, algoritma greedy tidak menjamin solusi yang diperoleh selalu optimal secara global untuk semua jenis masalah

Agar algoritma greedy menghasilkan solusi optimal, masalah yang dihadapi harus memiliki struktur optimasi tertentu, terutama memenuhi sifat pilihan rakus (*greedy choice property*) dan *optimal substructure*. *Optimal substructure* berarti solusi optimal global dapat dibangun dari solusi optimal sub-masalahnya. Sementara *greedy choice property* mensyaratkan bahwa pilihan lokal optimal akan mengarah pada solusi global optimal. Jika kedua kondisi ini terpenuhi, algoritma greedy akan menemukan solusi optimal dengan lebih cepat dibanding metode eksplorasi menyeluruh. Namun, jika kondisi tersebut tidak terpenuhi, solusi yang dihasilkan mungkin hanya mendekati optimum atau bahkan jauh dari optimum

Algoritma greedy diterapkan luas di berbagai bidang optimasi dan ilmu komputer. Beberapa contoh penerapan mencakup:

- *Activity Selection Problem* (pemilihan sejumlah kegiatan dengan konflik jadwal minimum) dapat diselesaikan secara optimal dengan strategi greedy, yaitu selalu memilih kegiatan yang selesai paling awal terlebih dahulu. Begitu pula penjadwalan tugas pada sistem multiprosesor dan pemadatan jadwal kuliah sering memanfaatkan pendekatan greedy untuk memperoleh jadwal efisien dengan kompleksitas yang lebih rendah dibanding metode exact.
- Algoritma graf bersifat *greedy*, misalnya *Prim* dan *Kruskal* untuk mencari Minimum Spanning Tree, serta algoritma *Dijkstra* untuk rute terpendek pada graf berbobot non-negatif. Algoritma tersebut setiap kalinya memilih edge atau node terdekat yang paling optimal lokal hingga membangun solusi global. Dalam jaringan komputer, pendekatan greedy digunakan untuk routing heuristik dan pengalokasian kanal yang cepat meskipun tidak selalu optimal absolut.
- Algoritma *Huffman coding* dalam kompresi data menggunakan strategi greedy untuk membangun kode prefiks optimal berdasarkan frekuensi simbol. Di bidang lain seperti seleksi fitur machine learning atau alokasi investasi, heuristik *greedy* kerap dipakai untuk memperoleh solusi cukup baik secara cepat ketika metode eksak terlalu mahal secara komputasi.

## 2.2 Robocode TankRoyale

Robocode TankRoyale adalah sebuah platform *game simulasi* pertarungan tank virtual yang dirancang untuk tujuan edukasi pemrograman dan eksperimen strategi AI. Dalam game ini, pemain bukanlah manusia yang mengendalikan tank secara langsung, melainkan program (bot) yang ditulis oleh pemain tersebut. Setiap bot mengontrol sebuah tank dan bertarung melawan bot lain di arena secara real-time. Tugas programmer (*"Robocoder"*) adalah menulis kode yang menjadi *otak* tank, menentukan bagaimana tank bergerak, membidik, menembak, dan bereaksi terhadap situasi di medan pertempuran. Selama pertandingan, bot menjalankan programnya secara otonom tanpa intervensi langsung, dan keberhasilan bot diukur dari skor yang diperoleh (misal melalui hit pada musuh atau bertahan hidup paling akhir). Robocode TankRoyale dikembangkan sebagai versi baru dari game Robocode klasik dengan perbaikan pada fleksibilitas bahasa dan arsitektur sistem, sehingga cocok untuk eksperimen strategi AI yang lebih luas.

Robocode TankRoyale dibangun dengan arsitektur modular berbasis *client-server*. Komponen utama permainan terbagi menjadi:

(a) **Server** – menjalankan inti game dan menegakkan peraturan pertarungan,
(b) **GUI (Graphical User Interface)** – antarmuka visual untuk menampilkan arena dan mengontrol jalannya pertandingan, dan
(c) **Booter** – modul untuk meluncurkan bot dari mesin lokal dan menghubungkannya ke server. Seluruh komponen game ini ditulis di atas platform Java menggunakan bahasa Kotlin.

Robocode TankRoyale mengadopsi protokol komunikasi berbasis WebSocket antara server, bot, dan komponen observer (seperti GUI). Dengan pendekatan ini, bot dapat dijalankan sebagai proses terpisah di mana saja (bahkan di komputer lain atau cloud) asalkan dapat terhubung ke server melalui WebSocket. Hal ini merupakan peningkatan signifikan dari Robocode versi lama yang membatasi bot berjalan di satu JVM Java. Arsitektur terdistribusi ini memungkinkan *multi-platform support*: bot dapat ditulis dalam bahasa apa pun (Java, Kotlin, **C#**, Python, dll.) selama bahasa tersebut memiliki kemampuan WebSocket untuk komunikasi. Secara khusus, pengembang Robocode menyediakan Bot API untuk .NET (C#) dan Java agar programmer dapat dengan mudah membuat bot dalam bahasa tersebut tanpa harus mengimplementasi protokol komunikasi secara manual. Dengan desain modular, setiap komponen (misal GUI atau server) dapat diganti atau ditingkatkan oleh pengembang lain, sehingga platform ini extensible untuk pengembangan fitur lebih lanjut.

Robocode TankRoyale beroperasi dalam mode turn-based (giliran) yang berjalan sangat cepat sehingga menyerupai waktu nyata. Setiap *turn* mewakili satu siklus simulasi di mana bot menerima informasi sensorik dan mengirim perintah aksi. Berbeda dengan versi lama Robocode yang mengeksekusi bot di thread terpisah (sehingga urutan eksekusi bisa acak), TankRoyale mengumpulkan semua *perintah* (intents) bot per turn dan menerapkannya secara simultan untuk memperbarui keadaan game.

## 2.3 Perbandingan Heuristik Greedy dengan Metode Lain

Algoritma Greedy memiliki beberapa keunggulan dibandingkan dengan metode optimasi lainnya seperti Dynamic Programming, Brute Force, dan pendekatan lainnya. Dari segi efisiensi komputasi, Greedy biasanya memiliki kompleksitas waktu yang lebih rendah dibandingkan metode exhaustive search seperti Brute Force atau Branch and Bound, sehingga sangat cocok untuk skenario real-time seperti pada game Robocode TankRoyale. Selain itu, implementasi algoritma Greedy cenderung lebih sederhana dan membutuhkan memori yang lebih sedikit dibandingkan Dynamic Programming, yang memerlukan tabel

memoization untuk menyimpan solusi sub-masalah. Dalam konteks permainan bot, Greedy dapat membuat keputusan cepat yang sangat bernilai ketika waktu komputasi per turn sangat terbatas.

Namun demikian, pendekatan Greedy juga memiliki keterbatasan yang perlu dipertimbangkan. Tidak seperti teknik seperti Reinforcement Learning atau Neural Networks yang mampu belajar dan beradaptasi dari pengalaman, algoritma Greedy tidak memiliki kemampuan pembelajaran dan hanya bergantung pada heuristik yang telah didefinisikan sebelumnya. Ketika dibandingkan dengan Genetic Algorithm yang mampu mengeksplorasi berbagai solusi secara paralel, Greedy cenderung terjebak pada optimum lokal karena tidak memiliki mekanisme untuk "melompat" ke area solusi yang berbeda. Pada kasus seperti Robocode TankRoyale, kombinasi antara pendekatan Greedy untuk keputusan jangka pendek dengan metode lain untuk perencanaan strategis jangka panjang dapat menghasilkan performa yang lebih optimal.

# BAB III
# APLIKASI STRATEGI GREEDY

## 3.1 Pemetaan Persoalan ke Elemen Algoritma Greedy

Dalam konteks permainan Robocode Tank Royale, implementasi algoritma greedy memerlukan pemetaan yang tepat dari elemen-elemen permainan ke komponen-komponen algoritma greedy. Himpunan kandidat dalam permainan ini mencakup seluruh kemungkinan aksi yang dapat dilakukan oleh bot pada setiap giliran, meliputi pergerakan tank (maju, mundur, berhenti), arah pergerakan (berputar dengan sudut tertentu), arah dan daya penembakan, strategi pemindaian, serta berbagai reaksi terhadap peristiwa seperti terkena tembakan atau tabrakan. Setiap aksi ini menjadi kandidat yang akan dievaluasi untuk memberikan keuntungan maksimal dalam jangka pendek.

Himpunan solusi dalam konteks ini merupakan rangkaian aksi yang telah dipilih pada setiap turn, yang secara keseluruhan membentuk strategi permainan bot. Solusi optimal adalah kombinasi aksi yang memaksimalkan skor akhir melalui berbagai komponen seperti Bullet Damage, Bullet Damage Bonus, Survival Score, Last Survival Bonus, Ram Damage, dan Ram Damage Bonus. Setiap komponen skor ini menjadi pertimbangan dalam pengambilan keputusan greedy pada setiap langkah permainan.

Fungsi solusi mengidentifikasi apakah urutan aksi yang telah dipilih membentuk solusi akhir yang valid, yang terpenuhi ketika bot mencapai akhir pertandingan, seluruh aksi telah dilaksanakan dengan benar, dan total skor telah dihitung. Sementara itu, fungsi seleksi berperan penting dalam memilih kandidat terbaik untuk setiap langkah, yang dalam implementasi bot kami terlihat pada pengambilan keputusan seperti prioritas target, pemilihan daya tembak, dan keputusan pergerakan berdasarkan kondisi pertempuran saat itu.

Fungsi kelayakan dalam permainan ini berfungsi menilai apakah kandidat yang dipilih masih mempertahankan solusi yang mungkin, dengan mengevaluasi faktor-faktor seperti kecukupan energi untuk aksi tertentu, potensi tabrakan dengan dinding, status panas meriam, dan jangkauan penembakan. Sementara fungsi objektif yang menjadi tujuan utama adalah memaksimalkan skor akhir, yang dapat dicapai melalui strategi seperti memaksimalkan damage yang diberikan, meminimalkan damage yang diterima, bertahan hidup selama mungkin, dan mengoptimalkan penggunaan energi untuk berbagai aksi.

## 3.2 Eksplorasi Alternatif Solusi Greedy

Dalam pengembangan bot untuk permainan Robocode Tank Royale, kami mengeksplorasi empat alternatif solusi greedy dengan heuristik yang berbeda. ScoobyGodBot mengimplementasikan pendekatan greedy berdasarkan damage maksimum yang disesuaikan dengan kondisi energi bot. Bot ini secara dinamis mengatur tingkat agresivitas berdasarkan level energi saat ini, dengan strategi menyerang agresif saat energi tinggi, seimbang pada energi menengah, dan defensif saat energi rendah. Pengaturan daya tembak juga disesuaikan dengan jarak target, yang mencerminkan pendekatan greedy untuk memaksimalkan damage output dengan mempertimbangkan efisiensi energi.

Beranjak ke *alternative bot*, ScoobyLuicy menerapkan strategi *greedy* berdasarkan target terdekat, dengan prioritas utama pada efisiensi penembakan dan *target acquisition*. Bot ini secara konsisten melacak posisi target terdekat dan berusaha melakukan penembakan dengan daya maksimal untuk menghasilkan damage tertinggi dalam jangka pendek. Pendekatan ini mengutamakan reaksi cepat terhadap target terdekat tanpa mempertimbangkan efisiensi energi jangka panjang, yang merupakan karakteristik khas dari algoritma greedy.

ScoobyShadowBot mengadopsi strategi greedy berbasis evasion dan scanning, mengoptimalkan kelangsungan hidup melalui pola gerakan zigzag yang dirancang untuk menghindar dari tembakan musuh sekaligus memaksimalkan area pemindaian. Daya tembak bot ini bervariasi berdasarkan jarak target, dengan fokus pada survival dan kemampuan untuk mendeteksi musuh secara efektif. Pendekatan ini mencerminkan strategi greedy yang memprioritaskan informasi dan kelangsungan hidup sebagai dasar pengambilan keputusan.

ScoobyRoma3000 mengimplementasikan solusi greedy yang sangat agresif, dengan pengejaran target dan penembakan full power untuk memaksimalkan damage output dalam waktu singkat. Bot ini memiliki sedikit pertimbangan defensif, dengan fokus utama pada serangan dan ramming untuk memaksimalkan damage. Strategi ini mencerminkan pendekatan greedy yang memilih keuntungan maksimal jangka pendek tanpa memperhatikan konsekuensi jangka panjang seperti efisiensi energi atau survivability.

## 3.3 Analisis Efisiensi dan Efektivitas Alternatif Solusi Greedy

Dari perspektif efisiensi komputasi, keempat bot yang kami kembangkan memiliki kompleksitas waktu yang relatif serupa, yaitu O(1) per turn, karena keputusan diambil berdasarkan informasi yang tersedia secara langsung tanpa kalkulasi yang kompleks. Namun, terdapat perbedaan dalam jumlah kondisional yang dievaluasi, dengan ScoobyGodBot dan ScoobyRoma3000 memiliki kompleksitas komputasi sedang, sementara ScoobyLuicy dan ScoobyShadowBot memiliki kompleksitas komputasi yang lebih rendah karena menggunakan pendekatan yang lebih sederhana.

Dalam hal penggunaan energi dan efektivitas penembakan, ScoobyGodBot menunjukkan efisiensi tertinggi dengan adaptasi berbasis level energi dan penyesuaian daya tembak berdasarkan jarak. Strategi ini memungkinkan keseimbangan yang baik antara damage output dan konservasi energi. Sebaliknya, ScoobyLuicy dan ScoobyRoma3000 cenderung kurang efisien dalam penggunaan energi karena sering menembak dengan daya penuh pada semua target. ScoobyShadowBot berada di tengah spektrum, dengan efisiensi energi moderat dan variasi daya tembak berdasarkan jarak.

Adaptabilitas terhadap kondisi pertempuran juga bervariasi antar bot, dengan ScoobyGodBot menunjukkan adaptabilitas tertinggi karena kemampuannya untuk menyesuaikan perilaku berdasarkan level energi. ScoobyShadowBot juga menunjukkan adaptabilitas yang baik dalam situasi defensive, sementara ScoobyLuicy memiliki adaptabilitas sedang dengan fokus pada pengejaran target terdekat. ScoobyRoma3000 memiliki adaptabilitas terendah dengan strategi yang sangat agresif tanpa banyak pertimbangan defensive.

## 3.4 Strategi Greedy Terpilih

Berdasarkan analisis komprehensif terhadap keempat alternatif solusi greedy, kami memilih ScoobyGodBot sebagai bot utama untuk implementasi. Pemilihan ini didasarkan pada beberapa pertimbangan penting. Pertama, ScoobyGodBot menunjukkan adaptabilitas superior dengan kemampuan untuk menyesuaikan strategi berdasarkan level energi, memungkinkan bot untuk bertahan lebih lama sekaligus memaksimalkan damage output ketika kondisi memungkinkan. Kedua, bot ini menawarkan keseimbangan yang sangat baik antara kemampuan menyerang dan bertahan, dengan pertimbangan jarak target untuk efisiensi penembakan dan penyesuaian strategi berbasis energi.

Ketiga, ScoobyGodBot mendemonstrasikan penggunaan energi yang lebih efisien dibandingkan pendekatan yang selalu agresif, yang meningkatkan peluang survival jangka panjang. Keempat, bot ini dirancang untuk performa optimal dalam berbagai skenario pertempuran, tidak seperti bot lain yang mungkin unggul dalam kondisi spesifik namun kurang optimal dalam kondisi lainnya. Dan kelima, pendekatan adaptif ScoobyGodBot lebih selaras dengan fungsi objektif memaksimalkan skor akhir, dengan pertimbangan berbagai komponen skor seperti damage, survival, dan efisiensi.

Meskipun bot alternatif kami memiliki keunggulan dalam aspek spesifik, ScoobyGodBot menawarkan solusi yang lebih komprehensif dan seimbang untuk memaksimalkan performa dalam berbagai kondisi pertempuran Robocode Tank Royale. Keseimbangan antara agresivitas dan defensif, serta kemampuan adaptasi terhadap perubahan kondisi pertempuran, menjadikannya kandidat terbaik untuk implementasi utama dalam kompetisi. Pendekatan greedy yang adaptif ini memungkinkan bot untuk mengambil keputusan optimal pada setiap langkah sambil tetap mempertimbangkan konsekuensi jangka menengah, yang merupakan pengembangan dari pendekatan greedy murni yang hanya fokus pada keuntungan langsung.

# BAB IV
# IMPLEMENTASI DAN PENGUJIAN

## 4.1 Implementasi Solusi

### 4.1.1 ScoobyGodBot (Bot Utama)

Bot utama kami, ScoobyGodBot, mengimplementasikan strategi greedy adaptif berdasarkan kondisi energi dengan fokus pada keseimbangan antara damage output dan survival. Berikut adalah pseudocode detailnya:

```
1. FUNCTION Main()
2.     Inisialisasi ScoobyGodBot
3.     CALL Start()
4. END FUNCTION
5.
6. CONSTRUCTOR ScoobyGodBot()
7.     Load konfigurasi dari file "ScoobyGodBot.json"
8. END CONSTRUCTOR
9.
10.   FUNCTION Run()
11.       // Set warna bot
12.       BodyColor = Color.Brown
13.       TurretColor = Color.Red
14.       BulletColor = Color.Purple
15.
16.       // Loop utama selama bot berjalan
17.       WHILE IsRunning DO
18.           CALL AdaptiveMovement()
19.           SetTurnGunRight(10)
20.       END WHILE
21.   END FUNCTION
22.
23.   FUNCTION AdaptiveMovement()
24.       // Strategi greedy dengan adaptasi berdasarkan energi
25.       IF Energy > 70 THEN
26.           // Strategi agresif saat energi tinggi
27.           SetTurnRight(30)
28.           Forward(200)
29.       ELSE IF Energy > 30 THEN
30.           // Strategi seimbang saat energi menengah
```

```
            SetTurnRight(60)
            Forward(100)
        ELSE
            // Strategi defensif saat energi rendah
            SetTurnLeft(90)
            Back(150)
        END IF
    END FUNCTION

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

        // Strategi greedy untuk pemilihan power tembakan
    berdasarkan jarak
        IF distance < 150 THEN
            // Jarak dekat: full power untuk damage maksimum
            Fire(3)
        ELSE IF distance < 500 THEN
            // Jarak menengah: power sedang untuk efisiensi
            Fire(2)
        ELSE
            // Jarak jauh: full power kembali untuk damage yang
    signifikan
            Fire(3)
        END IF
    END FUNCTION

    FUNCTION OnHitBot(e)
        // Strategi greedy untuk mengoptimalkan ramming damage
        IF e.IsRammed AND Energy > 20 THEN
            Fire(3)
            Forward(50)
        END IF
    END FUNCTION

    FUNCTION OnHitWall(e)
        // Menghindari dinding untuk mengurangi damage
        Back(150)
        SetTurnRight(120)
    END FUNCTION
```

Algoritma greedy dalam ScoobyGodBot terlihat pada adaptasi strategi berdasarkan level energi saat ini. Saat energi tinggi (>70), bot bergerak agresif dan maju jauh untuk mencari musuh. Saat energi menengah

(>30), bot bergerak dengan lebih seimbang. Saat energi rendah (<30), bot beralih ke mode defensif, mundur dan menghindari konfrontasi langsung.

Pemilihan daya tembak juga mengikuti pendekatan greedy dengan optimasi berdasarkan jarak. Untuk target jarak dekat dan jauh, bot menggunakan daya maksimum untuk memaksimalkan damage, sementara untuk target jarak menengah, bot menggunakan daya sedang untuk keseimbangan antara damage dan efisiensi energi.

### 4.1.2 ScoobyShadowBot (Bot Alternatif 1)

ScoobyShadowBot mengimplementasikan strategi greedy berbasis evasion dan scan dengan pola pergerakan zigzag. Berikut pseudocodenya:

```
1. FUNCTION Main()
2.     Inisialisasi ScoobyShadowBot
3.     CALL Start()
4. END FUNCTION
5.
6. CONSTRUCTOR ScoobyShadowBot()
7.     Load konfigurasi dari file "ScoobyShadowBot.json"
8. END CONSTRUCTOR
9.
10.   FUNCTION Run()
11.       // Set warna bot
12.       BodyColor = Color.Black
13.       GunColor = Color.Red
14.       RadarColor = Color.Purple
15.       BulletColor = Color.Blue
16.
17.       // Loop utama selama bot berjalan
18.       WHILE IsRunning DO
19.           CALL EvadeAndScan()
20.       END WHILE
21.   END FUNCTION
22.
23.   FUNCTION EvadeAndScan()
24.       // Strategi greedy: pola zigzag untuk evasion dan
   scanning optimal
25.       SetTurnRight(45)
26.       Forward(160)
27.       SetTurnLeft(90)
28.       Forward(160)
29.       SetTurnGunRight(45)
30.   END FUNCTION
```

```
31.
32.   FUNCTION OnScannedBot(e)
33.       distance = DistanceTo(e.X, e.Y)
34.
35.       // Strategi greedy untuk pemilihan power tembakan
   berdasarkan jarak
36.       IF distance > 400 THEN
37.           // Jarak jauh: full power hanya jika meriam tidak
   panas
38.           IF GunHeat == 0 THEN
39.               Fire(3)
40.           END IF
41.       ELSE IF distance > 200 THEN
42.           // Jarak menengah: power sedang
43.           Fire(2)
44.       ELSE
45.           // Jarak dekat: hindari dan full power
46.           SetTurnRight(90)
47.           Back(100)
48.           Fire(3)
49.       END IF
50.   END FUNCTION
51.
52.   FUNCTION OnHitByBullet(e)
53.       // Strategi evasive: belok dan maju untuk menjauh
54.       SetTurnRight(60)
55.       Forward(150)
56.   END FUNCTION
57.
58.   FUNCTION OnHitWall(e)
59.       // Menghindari dinding
60.       Back(180)
61.       SetTurnRight(130)
62.   END FUNCTION
```

Algoritma greedy dalam ScoobyShadowBot terlihat pada pola pergerakan zigzag untuk maksimalisasi scanning sambil menghindar, serta pemilihan daya tembak yang disesuaikan dengan jarak. Bot ini menggunakan pendekatan konservatif untuk tembakan jarak jauh, hanya menembak dengan daya penuh jika meriam tidak panas, yang mencerminkan optimasi penggunaan energi.

### 4.1.3 ScoobyLuicy (Bot Alternatif 2)

ScoobyLuicy mengimplementasikan strategi greedy berbasis target terdekat, dengan prioritas pada serangan agresif. Berikut adalah pseudocodenya:

```
1. FUNCTION Main()
2.     Inisialisasi ScoobyLuicy
3.     CALL Start()
4. END FUNCTION
5.
6. CONSTRUCTOR ScoobyLuicy()
7.     Inisialisasi random number generator
8.     targetX = -1
9.     targetY = -1
10.       targetDistance = double.MaxValue
11.       lastFireTurn = 0
12.       lastTargetTurn = -100
13.       Load konfigurasi dari file "ScoobyLuicy.json"
14.   END CONSTRUCTOR
15.
16.   FUNCTION Run()
17.       // Set warna bot
18.       BodyColor = Color.Black
19.       GunColor = Color.Red
20.       RadarColor = Color.Purple
21.       BulletColor = Color.Blue
22.       ScanColor = Color.Yellow
23.       MaxSpeed = 8
24.
25.       // Loop utama selama bot berjalan
26.       WHILE IsRunning DO
27.           // Strategi greedy: fokus pada target terakhir jika
   baru terdeteksi
28.           IF TurnNumber - lastTargetTurn < 2 THEN
29.               // Arahkan bot dan meriam ke target terakhir
30.               angleToTarget = GunBearingTo(targetX, targetY)
31.               TurnLeft(angleToTarget)
32.               gunTurn = GunBearingTo(targetX, targetY)
33.               TurnGunLeft(gunTurn)
34.
35.               // Tembak dengan power maksimum jika interval
```

```
   terpenuhi
36.              IF TurnNumber - lastFireTurn >= 1 THEN
37.                  Fire(3)
38.                  lastFireTurn = TurnNumber
39.              END IF
40.
41.              // Strategi greedy untuk pergerakan: maju lebih
   cepat jika target dekat
42.              distance = DistanceTo(targetX, targetY)
43.              IF distance < 100 THEN
44.                  Forward(200)
45.              ELSE
46.                  Forward(MIN(150, distance))
47.              END IF
48.          ELSE
49.              // Jika tidak ada target baru, putar dan maju
   untuk scanning
50.              TurnLeft(99999)
51.              Forward(99999)
52.          END IF
53.      END WHILE
54.  END FUNCTION
55.
56.  FUNCTION OnScannedBot(e)
57.      distance = DistanceTo(e.X, e.Y)
58.
59.      // Strategi greedy: update target jika lebih dekat dari
   target sebelumnya
60.      IF distance < targetDistance THEN
61.          targetX = e.X
62.          targetY = e.Y
63.          targetDistance = distance
64.          lastTargetTurn = TurnNumber
65.      END IF
66.
67.      // Tembak target yang terdeteksi secara langsung
68.      gunTurn = GunBearingTo(e.X, e.Y)
69.      TurnGunLeft(gunTurn)
70.      IF TurnNumber - lastFireTurn >= 1 THEN
71.          Fire(3)
72.          lastFireTurn = TurnNumber
73.      END IF
74.  END FUNCTION
```

```
75.
76.   FUNCTION OnHitByBullet(e)
77.       // Respon minimal terhadap tembakan
78.       TurnRight(random.Next(10, 30))
79.       Forward(50)
80.   END FUNCTION
81.
82.   FUNCTION OnHitBot(e)
83.       // Strategi greedy untuk memaksimalkan ramming damage
84.       gunTurn = GunBearingTo(e.X, e.Y)
85.       TurnGunLeft(gunTurn)
86.       Fire(3)
87.       Forward(100)
88.   END FUNCTION
89.
90.   FUNCTION OnHitWall(e)
91.       // Menghindari dinding
92.       Back(180)
93.       SetTurnRight(130)
94.   END FUNCTION
```

Algoritma greedy dalam ScoobyLuicy terlihat pada pemilihan target berdasarkan jarak terdekat dan fokus pada target tersebut selama dua turn setelah deteksi. Bot ini selalu menggunakan daya tembak maksimum (3) untuk memaksimalkan damage output jangka pendek tanpa mempertimbangkan efisiensi energi.

### 4.1.4 ScoobyRoma3000 (Bot Alternatif 3)

ScoobyRoma3000 mengimplementasikan strategi greedy agresif dengan fokus pada pengejaran target dan ramming. Berikut pseudocodenya:

```
1. FUNCTION Main()
2.     Inisialisasi ScoobyRoma3000
3.     CALL Start()
4. END FUNCTION
5.
6. CONSTRUCTOR ScoobyRoma3000()
7.     Inisialisasi random number generator
8.     FIRE_INTERVAL = 1
9.     lastFireTurn = 0
10.       targetX = -1
11.       targetY = -1
12.       targetDistance = double.MaxValue
13.       lastTargetTurn = -100
```

```
14.       Load konfigurasi dari file "ScoobyRoma3000.json"
15.   END CONSTRUCTOR
16.
17.   FUNCTION DistanceTo(x, y)
18.       dx = x - X
19.       dy = y - Y
20.       RETURN sqrt(dx^2 + dy^2)
21.   END FUNCTION
22.
23.   FUNCTION Run()
24.       // Set warna bot
25.       BodyColor = Color.Red
26.       GunColor = Color.Black
27.       RadarColor = Color.Yellow
28.       BulletColor = Color.Magenta
29.       ScanColor = Color.Yellow
30.
31.       MaxSpeed = 8
32.
33.       // Loop utama selama bot berjalan
34.       WHILE IsRunning DO
35.           // Strategi greedy: fokus pada target terakhir jika
   baru terdeteksi
36.           IF TurnNumber - lastTargetTurn < 2 THEN
37.               // Arahkan bot dan meriam ke target
38.               angleToTarget = GunBearingTo(targetX, targetY)
39.               TurnLeft(angleToTarget)
40.               gunTurn = GunBearingTo(targetX, targetY)
41.               TurnGunLeft(gunTurn)
42.
43.               // Tembak dengan power maksimum jika interval
   terpenuhi
44.               IF TurnNumber - lastFireTurn >= FIRE_INTERVAL
   THEN
45.                   Fire(3)
46.                   lastFireTurn = TurnNumber
47.               END IF
48.
49.               // Strategi agresif: maju lebih cepat jika target
   dekat untuk ramming
50.               d = DistanceTo(targetX, targetY)
51.               IF d < 100 THEN
52.                   Forward(200)
```

```
            ELSE
                Forward(MIN(150, d))
            END IF
        ELSE
            // Scanning agresif jika tidak ada target
            TurnLeft(99999)
            Forward(99999)
        END IF
    END WHILE
END FUNCTION

FUNCTION OnScannedBot(e)
    d = DistanceTo(e.X, e.Y)

    // Strategi greedy: update target jika lebih dekat
    IF d < targetDistance THEN
        targetX = e.X
        targetY = e.Y
        targetDistance = d
        lastTargetTurn = TurnNumber
    END IF

    // Tembak dengan power maksimum
    gunTurn = GunBearingTo(e.X, e.Y)
    TurnGunLeft(gunTurn)
    IF TurnNumber - lastFireTurn >= FIRE_INTERVAL THEN
        Fire(3)
        lastFireTurn = TurnNumber
    END IF
END FUNCTION

FUNCTION OnHitByBullet(e)
    // Respon minimal, tetap agresif
    TurnRight(random.Next(10, 30))
    Forward(50)
END FUNCTION

FUNCTION OnHitBot(e)
    // Strategi ramming agresif
    gunTurn = GunBearingTo(e.X, e.Y)
    TurnGunLeft(gunTurn)
    Fire(3)
    Forward(100)
```

```
96.   END FUNCTION
97.
98.   FUNCTION OnHitWall(e)
99.       // Menghindari dinding
100.      TurnLeft(90)
101.      Forward(100)
102.  END FUNCTION
```

Algoritma greedy dalam ScoobyRoma3000 terlihat pada fokus agresif untuk pengejaran target terdekat dan penggunaan daya tembak maksimum secara konsisten. Bot ini memiliki respons minimal terhadap serangan, memprioritaskan posisi ofensif daripada defensif, yang mencerminkan pendekatan greedy untuk memaksimalkan damage output jangka pendek.

## 4.2 Struktur Program

### 4.2.1 Struktur Data

Bot utama kami, ScoobyGodBot, beserta 3 bot alternatif lainnya menerapkan paradigma *Object Oriented Programming* dalam rancangan bot ini. Kelas ScoobyGodBot merupakan warisan (*inheritance*) dari kelas Bot. Dalam Kelas ScoobyGodBot, terdapat beberapa atribut (variabel) dan metode (fungsi/prosedur) yang terkandung dalam kelas tersebut, di antaranya:

- BodyColor (Color): Variabel yang menyimpan warna tank pada bot.
- TurretColor (Color): Variabel yang menyimpan warna laras tank pada bot.
- BulletColor (Color): Variabel yang menyimpan warna peluru yang ditembak bot tersebut.
- IsRunning (boolean): Menentukan apakah bot sedang bergerak ketika turnamen.
- Energy (double): Variabel yang menyatakan besaran energi yang dimiliki bot tersebut ketika turnamen.
- distance (double): Variabel yang menyatakan jarak bot tersebut dengan bot lain yang ter-*scan*.

### 4.2.2 Fungsi dan Prosedur Utama

Bot utama kami, ScoobyGodBot, menggunakan beberapa fungsi dan prosedur utama untuk mengimplementasikan strategi greedy adaptif diantaranya adalah sebagai berikut:

- Run(): Fungsi utama yang dijalankan saat bot dimulai. Menetapkan warna bot dan menjalankan loop utama yang memanggil fungsi-fungsi pergerakan dan pemutaran meriam.
- AdaptiveMovement(): Prosedur yang mengimplementasikan strategi pergerakan adaptif berdasarkan level energi. Ini adalah komponen utama strategi greedy, di mana bot membuat keputusan pergerakan berdasarkan kondisi energi saat ini.
- OnScannedBot(ScannedBotEvent e): Handler event yang dipanggil saat radar mendeteksi bot musuh. Mengimplementasikan strategi greedy untuk pemilihan daya tembak berdasarkan jarak target.

- OnHitBot(HitBotEvent e): Handler event yang dipanggil saat bot bertabrakan dengan bot lain. Mengimplementasikan strategi ofensif untuk mengoptimalkan ram damage jika kondisi memungkinkan.
- OnHitWall(HitWallEvent e): Handler event yang dipanggil saat bot menabrak dinding. Mengimplementasikan strategi evasive untuk menghindari damage berlebih dari dinding.

Semua fungsi ini dirancang untuk bekerja bersama dalam mengimplementasikan strategi *greedy* adaptif yang menyeimbangkan kebutuhan ofensif dan defensif berdasarkan kondisi pertempuran.

# 4.3 Pengujian

## 4.3.1 Skenario Pengujian

Kami melakukan empat skenario pengujian untuk mengevaluasi performa ScoobyGodBot:

1. **Pengujian 1 - Skenario Duel**: ScoobyGodBot vs Bot Alternatif (1v1) dengan pengaturan default arena.
2. **Pengujian 2 - Skenario Battle Royal**: ScoobyGodBot  vs Bot Alternatif (1 vs 1 vs 1 vs 1) dengan pengaturan default arena.
3. **Pengujian 3 - Skenario Multi-Target**: ScoobyGodBot vs 3 Bot Asisten dengan pengaturan default arena.
4. **Pengujian 4 - Skenario Battle Royal**: 4 Bot Kelompok ScoobyDude vs 4 Bot Asisten dengan pengaturan default arena.

## 4.3.2 Hasil dan Analisis Pengujian

**Pengujian 1 - Skenario Duel**

**Hasil:**

Results for 10 rounds

| Rank | Name | Total Score | Survival | Surv. Bonus | Bullet D... | Bullet Bo... | Ram Dmg. | Ram Bonus | 1sts | 2nds | 3rds |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | ScoobyGodBot 1.0 | 454 | 200 | 40 | 104 | 0 | 80 | 29 | 4 | 0 | 0 |
| 2 | ScoobyShadowBot 1.0 | 190 | 0 | 0 | 110 | 0 | 80 | 0 | 0 | 4 | 0 |

Results for 10 rounds

| Rank | Name | Total Score | Survival | Surv. Bonus | Bullet D... | Bullet Bo... | Ram Dmg. | Ram Bonus | 1sts | 2nds | 3rds |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | ScoobyGodBot 1.0 | 602 | 150 | 30 | 270 | 47 | 104 | 0 | 4 | 1 | 0 |
| 2 | ScoobyLuicy 1.0 | 270 | 50 | 10 | 144 | 15 | 50 | 0 | 1 | 4 | 0 |

Results for 10 rounds

| Rank | Name | Total Score | Survival | Surv. Bonus | Bullet D... | Bullet Bo... | Ram Dmg. | Ram Bonus | 1sts | 2nds | 3rds |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | ScoobyGodBot 1.0 | 374 | 100 | 20 | 204 | 0 | 50 | 0 | 3 | 1 | 0 |
| 2 | ScoobyRoma3000 1.0 | 211 | 50 | 10 | 128 | 13 | 10 | 0 | 1 | 3 | 0 |

**Analisis:**

- Dalam pertandingan duel 1v1 melawan bot alternatif, ScoobyGodBot menunjukkan performa yang sangat baik dengan memanfaatkan strategi adaptif berdasarkan level energi. ScoobyGodBot berhasil meraih kemenangan melawan 3 bot alternatif yang menandakan bahwa strategi *greedy* ScoobyGodBot berhasil mendapatkan nilai optimal.

**Pengujian 2 - Skenario Battle Royal**

**Hasil:**

Results for 10 rounds

| Rank | Name | Total Score | Survival | Surv. Bonus | Bullet D... | Bullet Bo... | Ram Dmg. | Ram Bonus | 1sts | 2nds | 3rds |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | ScoobyGodBot 1.0 | 844 | 350 | 60 | 272 | 8 | 154 | 0 | 2 | 1 | 0 |
| 2 | ScoobyShadowBot 1.0 | 697 | 250 | 30 | 332 | 13 | 71 | 0 | 1 | 1 | 1 |
| 3 | ScoobyLuicy 1.0 | 694 | 350 | 0 | 256 | 12 | 76 | 0 | 0 | 2 | 2 |
| 4 | ScoobyRoma3000 1.0 | 497 | 250 | 30 | 176 | 16 | 25 | 0 | 1 | 0 | 1 |

Results for 10 rounds

| Rank | Name | Total Score | Survival | Surv. Bonus | Bullet D... | Bullet Bo... | Ram Dmg. | Ram Bonus | 1sts | 2nds | 3rds |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | ScoobyGodBot 1.0 | 1067 | 450 | 60 | 400 | 32 | 113 | 12 | 3 | 0 | 1 |
| 2 | ScoobyShadowBot 1.0 | 553 | 250 | 0 | 216 | 24 | 62 | 0 | 0 | 3 | 0 |
| 3 | ScoobyLuicy 1.0 | 544 | 350 | 30 | 144 | 6 | 13 | 0 | 1 | 0 | 3 |
| 4 | ScoobyRoma3000 1.0 | 231 | 100 | 0 | 112 | 0 | 19 | 0 | 0 | 1 | 0 |

Results for 10 rounds

| Rank | Name | Total Score | Survival | Surv. Bonus | Bullet D... | Bullet Bo... | Ram Dmg. | Ram Bonus | 1sts | 2nds | 3rds |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | ScoobyGodBot 1.0 | 1018 | 400 | 60 | 386 | 31 | 140 | 0 | 3 | 1 | 0 |
| 2 | ScoobyShadowBot 1.0 | 795 | 350 | 30 | 288 | 18 | 109 | 0 | 1 | 2 | 2 |
| 3 | ScoobyRoma3000 1.0 | 593 | 200 | 0 | 288 | 14 | 91 | 0 | 1 | 1 | 2 |
| 4 | ScoobyLuicy 1.0 | 406 | 250 | 30 | 96 | 0 | 30 | 0 | 0 | 1 | 1 |

**Analisis:**

- ScoobyGodBot berhasil meraih peringkat pertama 3 kali berturut-turut dari 3 turnamen. Poin terbanyak didapat dari survival, yang menunjukkan keberhasilan implementasi strategi adaptif yang menyesuaikan perilaku berdasarkan kondisi energi. Dan dari 10 ronde, ScoobyDude meraih posisi pertama “1st” sebanyak 3 kali pada pertandingan 2 dan 3 dan pada pertandingan 1 sebanyak 2 kali. Kemenangan ini menunjukkan bahwa strategi *greedy* ScoobyGodBot berhasil mendapatkan nilai optimal.

**Pengujian 3 - Skenario Multi-Target**

**Hasil:**

Results for 10 rounds

| Rank | Name | Total Score | Survival | Surv. Bonus | Bullet D... | Bullet Bo... | Ram Dmg. | Ram Bonus | 1sts | 2nds | 3rds |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | Spin Bot 1.0 | 916 | 300 | 0 | 496 | 38 | 60 | 22 | 2 | 2 | 0 |
| 2 | ScoobyGodBot 1.0 | 819 | 350 | 60 | 308 | 12 | 89 | 0 | 2 | 1 | 1 |
| 3 | Crazy 1.0 | 535 | 350 | 30 | 136 | 0 | 19 | 0 | 0 | 1 | 3 |
| 4 | Corners 1.0 | 98 | 50 | 0 | 48 | 0 | 0 | 0 | 0 | 0 | 0 |

Results for 10 rounds

| Rank | Name | Total Score | Survival | Surv. Bonus | Bullet D... | Bullet Bo... | Ram Dmg. | Ram Bonus | 1sts | 2nds | 3rds |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | ScoobyGodBot 1.0 | 817 | 400 | 60 | 270 | 25 | 61 | 0 | 3 | 1 | 0 |
| 2 | Spin Bot 1.0 | 705 | 200 | 30 | 432 | 19 | 23 | 0 | 1 | 1 | 2 |
| 3 | Crazy 1.0 | 458 | 250 | 0 | 176 | 0 | 28 | 4 | 0 | 1 | 2 |
| 4 | Fire 1.0 | 142 | 50 | 0 | 92 | 0 | 0 | 0 | 0 | 1 | 0 |

Results for 10 rounds

| Rank | Name | Total Score | Survival | Surv. Bonus | Bullet D... | Bullet Bo... | Ram Dmg. | Ram Bonus | 1sts | 2nds | 3rds |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | ScoobyGodBot 1.0 | 1203 | 450 | 90 | 516 | 73 | 65 | 8 | 4 | 0 | 0 |
| 2 | Crazy 1.0 | 528 | 300 | 0 | 204 | 8 | 16 | 0 | 0 | 4 | 0 |
| 3 | Fire 1.0 | 266 | 150 | 0 | 112 | 4 | 0 | 0 | 0 | 0 | 4 |
| 4 | Corners 1.0 | 138 | 0 | 0 | 138 | 0 | 0 | 0 | 0 | 0 | 0 |

**Analisis:**

- Dalam pertandingan melawan tiga bot asisten, ScoobyGodBot menghadapi pengambilan keputusan target dan manajemen energi yang lebih berat dari sebelumnya. Pada pertandingan pertama, ScoobyGodBot kalah dari SpinBot karena strategi rotasi konstan SpinBot yang membuat target sulit ditembak, Namun, pada pertandingan berikutnya, ScoobyGodBot berhasil mengatasi kelemahan ini dengan menerapkan strategi adaptif yang lebih efektif. ScoobyGodBot berhasil meraih peringkat pertama 2 kali berturut-turut dari 3 turnamen. Poin terbanyak didapat dari survival, yang menunjukkan keberhasilan implementasi strategi adaptif yang menyesuaikan perilaku berdasarkan kondisi energi. Dari 10 ronde di setiap pertandingan, ScoobyGodBot meraih posisi pertama "1st" sebanyak 2 kali pada pertandingan 1, 3 kali pada pertandingan 2, dan 4 kali pada pertandingan 3. Kemenangan ini menunjukkan bahwa strategi greedy ScoobyGodBot berhasil mendapatkan nilai optimal.

**Pengujian 4 - Skenario Battle Royale**

**Hasil:**

Results for 10 rounds

| Rank | Name | Total Score | Survival | Surv. Bonus | Bullet D... | Bullet Bo... | Ram Dmg. | Ram Bonus | 1sts | 2nds | 3rds |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | ScoobyGodBot 1.0 | 1503 | 750 | 140 | 464 | 17 | 132 | 0 | 3 | 0 | 0 |
| 2 | ScoobyRoma3000 1.0 | 752 | 450 | 0 | 208 | 19 | 74 | 0 | 0 | 0 | 1 |
| 3 | ScoobyShadowBot 1.0 | 723 | 350 | 0 | 240 | 15 | 97 | 20 | 0 | 2 | 0 |
| 4 | Corners 1.0 | 687 | 400 | 0 | 266 | 20 | 0 | 0 | 0 | 0 | 1 |
| 5 | Crazy 1.0 | 672 | 450 | 0 | 192 | 6 | 24 | 0 | 0 | 1 | 0 |
| 6 | Target 1.0 | 451 | 450 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 |
| 7 | ScoobyLuicy 1.0 | 372 | 200 | 0 | 144 | 0 | 28 | 0 | 0 | 0 | 1 |
| 8 | Fire 1.0 | 208 | 100 | 0 | 108 | 0 | 0 | 0 | 0 | 0 | 0 |

Results for 10 rounds

| Rank | Name | Total Score | Survival | Surv. Bonus | Bullet D... | Bullet Bo... | Ram Dmg. | Ram Bonus | 1sts | 2nds | 3rds |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | Spin Bot 1.0 | 2082 | 1000 | 140 | 816 | 72 | 53 | 0 | 3 | 0 | 0 |
| 2 | ScoobyGodBot 1.0 | 1393 | 750 | 0 | 492 | 39 | 100 | 12 | 0 | 2 | 0 |
| 3 | Crazy 1.0 | 910 | 700 | 0 | 180 | 2 | 28 | 0 | 0 | 1 | 0 |
| 4 | ScoobyLuicy 1.0 | 746 | 550 | 0 | 128 | 20 | 48 | 0 | 0 | 0 | 2 |
| 5 | ScoobyShadowBot 1.0 | 599 | 400 | 0 | 146 | 5 | 48 | 0 | 0 | 0 | 0 |
| 6 | Fire 1.0 | 494 | 300 | 0 | 184 | 9 | 0 | 0 | 0 | 0 | 1 |
| 7 | ScoobyRoma3000 1.0 | 322 | 250 | 0 | 64 | 0 | 8 | 0 | 0 | 0 | 0 |
| 8 | Target 1.0 | 204 | 200 | 0 | 0 | 0 | 4 | 0 | 0 | 0 | 0 |

Results for 10 rounds

| Rank | Name | Total Score | Survival | Surv. Bonus | Bullet D... | Bullet Bo... | Ram Dmg. | Ram Bonus | 1sts | 2nds | 3rds |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | Spin Bot 1.0 | 3001 | 1400 | 210 | 1168 | 143 | 79 | 0 | 4 | 1 | 0 |
| 2 | ScoobyGodBot 1.0 | 1489 | 550 | 70 | 536 | 43 | 289 | 0 | 1 | 1 | 0 |
| 3 | Crazy 1.0 | 1168 | 850 | 0 | 276 | 2 | 40 | 0 | 0 | 1 | 1 |
| 4 | ScoobyShadowBot 1.0 | 1145 | 850 | 0 | 252 | 11 | 28 | 3 | 0 | 1 | 1 |
| 5 | Fire 1.0 | 1138 | 800 | 0 | 312 | 23 | 2 | 0 | 0 | 1 | 1 |
| 6 | ScoobyLuicy 1.0 | 761 | 550 | 0 | 144 | 0 | 67 | 0 | 0 | 0 | 0 |
| 7 | ScoobyRoma3000 1.0 | 750 | 500 | 0 | 192 | 0 | 58 | 0 | 0 | 0 | 1 |
| 8 | Corners 1.0 | 706 | 450 | 0 | 252 | 0 | 4 | 0 | 0 | 0 | 1 |

Results for 10 rounds

| Rank | Name | Total Score | Survival | Surv. Bonus | Bullet D... | Bullet Bo... | Ram Dmg. | Ram Bonus | 1sts | 2nds | 3rds |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | ScoobyGodBot 1.0 | 1572 | 900 | 70 | 522 | 30 | 49 | 0 | 1 | 2 | 0 |
| 2 | Walls 1.0 | 1178 | 700 | 0 | 440 | 28 | 10 | 0 | 1 | 0 | 1 |
| 3 | Velocity Bot 1.0 | 1104 | 650 | 70 | 292 | 7 | 85 | 0 | 1 | 1 | 0 |
| 4 | ScoobyRoma3000 1.0 | 822 | 500 | 0 | 192 | 0 | 85 | 44 | 0 | 0 | 1 |
| 5 | ScoobyShadowBot 1.0 | 730 | 500 | 0 | 200 | 0 | 30 | 0 | 0 | 0 | 1 |
| 6 | Corners 1.0 | 527 | 350 | 0 | 174 | 0 | 2 | 0 | 0 | 0 | 0 |
| 7 | ScoobyLuicy 1.0 | 471 | 350 | 0 | 80 | 0 | 41 | 0 | 0 | 0 | 0 |
| 8 | Target 1.0 | 106 | 100 | 0 | 0 | 0 | 6 | 0 | 0 | 0 | 0 |

**Analisis:**

- ScoobyGodBot berhasil meraih peringkat pertama 2 kali dari 4 turnamen, dan peringkat kedua pada 2 turnamen lainnya. Poin terbanyak didapat dari survival, yang menunjukkan keberhasilan implementasi strategi adaptif yang menyesuaikan perilaku berdasarkan kondisi energi, ScoobyGodBot berhasil meraih posisi pertama sebanyak 3 kali pada pertandingan 1 dan 1 kali pada pertandingan 4 untuk mendapatkan kemenangan. Kemenangan ini menunjukkan bahwa strategi greedy ScoobyGodBot berhasil mendapatkan nilai optimal.

  Pada pertandingan kedua dan ketiga, ScoobyGodBot kalah dari SpinBot karena strategi rotasi konstan Spin Bot yang membuat target sulit ditembak, sama seperti di Pengujian 3 - Skenario Multi-Target. Walaupun begitu, ScoobyGodBot tetap menjadi bot terbaik kedua, mengalahkan semua bot lainnya termasuk bot-bot alternatif kami.

  Di antara bot alternatif kami buat, ScoobyRoma3000 menunjukkan performa terbaik pada pertandingan 1 dan 4, sementara ScoobyLuicy lebih baik pada pertandingan 2, dan ScoobyShadowBot unggul pada pertandingan 3. Hasil performa ini menunjukkan bahwa masing-masing bot alternatif memiliki kekuatan pada skenario berbeda, sesuai dengan heuristik greedy yang diimplementas

# BAB V
# KESIMPULAN DAN SARAN

## 5.1 Kesimpulan

Berdasarkan hasil pengembangan dan pengujian bot tank untuk permainan Robocode Tank Royale, kami dapat menarik beberapa kesimpulan penting terkait dengan tujuan yang telah ditetapkan di awal. Algoritma Greedy berhasil diterapkan pada bot yang telah kami buat. Algoritma Greedy yang paling optimal dalam permainan Robocode Tank Royale adalah algoritma yang mempertimbangkan energi bot yang bersangkutan dan jarak bot tersebut dengan bot lawan.

## 5.2 Saran

Berdasarkan pengalaman dalam pengembangan dan pengujian bot untuk Robocode Tank Royale, kami menyarankan beberapa hal untuk pengembangan bot lebih lanjut dan perbaikan implementasi algoritma, diantaranya:

1. Mengembangkan sistem prediksi trayektori peluru untuk meningkatkan akurasi penembakan, terutama terhadap target bergerak.
2. Mengembangkan strategi targeting yang lebih canggih untuk skenario multi-target dengan mempertimbangkan tingkat ancaman dan potensi payoff.
3. Mengeksplorasi pendekatan hybrid yang menggabungkan algoritma greedy untuk keputusan taktis dengan algoritma lain untuk strategi jangka panjang.
4. Mengembangkan mekanisme adaptasi terhadap strategi spesifik lawan untuk meningkatkan performa dalam berbagai skenario pertempuran.
5. Melakukan pengujian sistematis untuk optimasi parameter seperti threshold energi dan jarak optimal untuk variasi daya tembak.

## 5.3 Refleksi

Dari tugas besar 1 IF2211 Strategi Algoritma 2025 ini, kami, para anggota kelompok ScoobyDude, menyatakan bahwa wawasan kami terhadap Algoritma Greedy meluas setelah menerapkannya pada permainan Robocode Tank Royale.

# LAMPIRAN

Link Repository: https://github.com/danenftyessir/Tubes1_ScoobyDude.git

Link Video: https://youtu.be/oZV8lH66iSo

Link Bot Starter Pack: https://github.com/Ariel-HS/tubes1-if2211-starter-pack/releases/tag/v1.0

Tabel Laporan Akhir:

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

# DAFTAR PUSTAKA

Rinaldi Munir. Algoritma Greedy (Bagian 1). Diakses pada 12 Maret 2025 dari https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2020-2021/Algoritma-Greedy-(2021)-Bag1.pdf

Rinaldi Munir. Algoritma Greedy (Bagian 2). Diakses pada 12 Maret 2025 dari https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2020-2021/Algoritma-Greedy-(2021)-Bag2.pdf

Rinaldi Munir. Algoritma Greedy (Bagian 3). Diakses pada 15 Maret 2025 dari https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2021-2022/Algoritma-Greedy-(2022)-Bag3.pdf

Geeksforgeeks. Greedy Algorithms. Diakses pada 8 Maret 2024 dari https://www.geeksforgeeks.org/greedy-algorithms/

Robocode Tank Royale Documentation. Diakses pada 8 Maret 2025 dari https://robocode-dev.github.io/tank-royale/

Robocode Tank Royale Repository. Diakses pada 8 Maret 2025 dari https://github.com/robocode-dev/tank-royale