

Laporan Tugas Besar 1
IF2211 Strategi Algoritma

Pemanfaatan Algoritma *Greedy* dalam pembuatan bot permainan
Robocode Tank Royale

Disusun oleh:

Rendi Adinata (10123083)
Dzubyan Ilman Ramadhan (10122010)



PROGRAM STUDI TEKNIK INFORMATIKA
SEKOLAH TEKNIK ELEKTRO DAN INFORMATIKA
INSTITUT TEKNOLOGI BANDUNG
2025

Bab 1

Deskripsi Tugas

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

- Buat 4 bot (1 utama dan 3 alternatif) dalam bahasa C# (.net) yang mengimplementasikan algoritma Greedy pada bot permainan Robocode Tank Royale dengan tujuan memenangkan permainan.
- Strategi greedy yang diimplementasikan setiap kelompok harus dikaitkan dengan fungsi objektif dari permainan ini, yaitu memperoleh skor setinggi mungkin pada akhir pertempuran.
- Strategi greedy yang diimplementasikan harus berbeda untuk setiap bot yang diimplementasikan dan setiap strategi greedy harus menggunakan heuristic yang berbeda.
- Bot yang dibuat TIDAK BOLEH sama dengan SAMPEL yang diberikan sebagai CONTOH. Baik dari starter pack maupun dari repository engine asli.
- Program harus mengandung komentar yang jelas, dan untuk setiap strategi greedy yang disebutkan, harus dilengkapi dengan kode sumber yang dibuat. Artinya semua strategi harus diimplementasikan

Bab 2

Landasan Teori

Dasar Teori Algoritma Greedy

Algoritma greedy adalah pendekatan penyelesaian masalah dengan membuat pilihan optimal lokal pada setiap langkah untuk mencapai solusi optimal global. Dalam pengaplikasiannya dalam tugas ini, komponen utamanya meliputi:

- Himpunan Kandidat: Kumpulan semua opsi atau aksi yang mungkin diambil.
- Himpunan Solusi: Subset dari himpunan kandidat yang memenuhi kriteria solusi
- Fungsi Objektif : Fungsi yang ingin dimaksimalkan/diminimalkan pada tiap waktu.
- Fungsi Kelayakan : Menentukan apakah suatu kandidat valid untuk ditambahkan ke solusi.
- Fungsi Seleksi : Fungsi untuk memilih kandidat dengan nilai terbaik berdasarkan fungsi objektif

Algoritma ini bersifat short-sighted (tidak mempertimbangkan konsekuensi jangka panjang), tetapi efisien secara komputasi, cocok untuk aplikasi **real-time** seperti permainan.

Cara Kerja Program

Kerangka Umum Robocode Tank Royale

Bot diimplementasikan dalam bahasa C# sebagai file `.cs` yang di-*compile* dan dijalankan dalam *game engine* yang dimodifikasi. Setiap bot menjalani langkah-langkah yang telah di-*define* pada suatu *event* yang terjadi pada bot, seperti contoh : `IsRunning` adalah event robot yang terjadi selama bot berjalan, `OnScannedBot` adalah event ketika bot menscan bot lain, dan lain-lain.

Cara Menjalankan Bot

Bot yang telah dibuat dan telah dipastikan bebas dari error disimpan pada sebuah folder dengan bot-bot lainnya yang telah dibuat. Pada aplikasi Robocode Tank Royale, pastikan direktori folder yang berisi folder-folder bot telah ditambahkan. Lalu, buka Start Battle. File-file bot yang telah ditambahkan akan muncul di menu Bot Directories. Setelah itu, boot bot yang ingin dimainkan. Apabila bot menghilang dari Booted Bots, perbaiki bot

di editor kode. Jika bot muncul di Joined Bots, artinya bot telah sukses di-boot. Terakhir, tambahkan bot yang ingin dijalankan, lalu tekan Start Battle.

Implementasi Algoritma Greedy

Kita dapat memanfaatkan algoritma greedy untuk memenuhi tujuan utama kita, yaitu untuk memaksimalkan poin dari bot. Menurut dokumentasi bot, poin dapat diperoleh dengan menembak bot lain, menabrak bot lain, atau bertahan dari serangan bot lain. Bot juga mendapatkan poin bonus apabila bot berhasil mematikan bot lain. Maka dari itu, kita dapat mengimplementasikan bot yang secara langsung memaksimalkan poinnya dengan salah satu atau beberapa cara tersebut.

Bab 3

Aplikasi Strategi Greedy

Elemen-Elemen Algoritma Greedy

- Himpunan Kandidat: Semua langkah-langkah yang dapat dilakukan bot, menurut dokumentasi bot.
- Himpunan Solusi: Himpunan dari langkah-langkah yang dipilih dan dilaksanakan bot
- Fungsi Objektif : Poin bot dan energi bot.
- Fungsi Kelayakan : Pilih langkah yang membuat poin bertambah dan tidak mengurangi poin.
- Fungsi Seleksi : Pilih langkah yang memaksimalkan penambahan poin.

Solusi-solusi Alternatif

Alternatif 1: Wukong

Greedy by Damage

- Himpunan Kandidat: {Tembak, Hindar, Mendekat, Pindai}.
- Fungsi Objektif: Damage yang dihasilkan jika menembak.
- Fungsi Kelayakan: Jarak cukup dekat, tembak
- Seleksi: Jika energi cukup, pilih "Tembak"; jika tidak, pilih aksi dengan nilai tertinggi berikutnya (misal: hindar).

Efektivitas & Efisiensi

Operasi dalam loop utama dan event handler bersifat konstan ($O(1)$) per iterasi, sehingga secara keseluruhan cukup efisien. Wukong cenderung kurang akurat dalam menembak karena mekanisme penyesuaian meriam yang sederhana dan tidak mempertimbangkan prediksi gerakan musuh secara mendalam.

Alternatif 2: Erlang

Greedy by Energy

- Himpunan Kandidat: {Tembak, Hindar, Pindai}.
- Fungsi Objektif: Memaksimalkan *survival score*.
- Fungsi Kelayakan: Jika terkena serangan/tertabrak, segera ubah arah gerak
- Seleksi: Hitung posisi musuh, lalu gerakkan bot ke arah yang sesuai.

Efisiensi dan efektivitas

Sama seperti Wukong, operasional utama dan penanganan event dilakukan dalam waktu konstan ($O(1)$) per siklus, sehingga performa eksekusi relatif efisien. Erlang memiliki kelemahan dalam menghindari serangan lawan; meski strategi orbit dan evasion diterapkan, manuver penghindarannya masih kurang presisi dan bisa dimanfaatkan oleh musuh untuk mengincar posisi bot.

Alternatif 3: Tembok

Pada bot ini, kita memaksimalkan poin dari survival sekaligus mencegah bot lain memperoleh poin. Apabila bot menabrak bot lain, bot akan pergi berlawanan arah dari lokasi tabrakan untuk pergi sejauh mungkin dari bot lain, lalu melanjutkan pergerakan seperti biasa.

Efektivitas & Efisiensi

Karena algoritma yang dipakai tetap sama dan dieksekusi tepat sekali, dapat disimpulkan kompleksitas waktunya adalah $O(1)$. Namun, karena perilaku bot yang bersifat pasif, akan memerlukan waktu yang lama untuk menyelesaikan sebuah ronde.

Alternatif 4: Wintermute

Bot ini mengedepankan perolehan poin dari menembak bot lain. Bot akan secara agresif mencari bot yang terdekat dari pindaianya saat itu, lalu memilih bot itu sebagai target dan menembaknya. Apabila ada bot lain yang lebih dekat, bot akan menembak bot yang lebih dekat tersebut. Namun, karena keterbatasan jangkauan radar. Maka, bot hanya memilih bot terdekat yang dapat dilihat oleh radar pada suatu waktu.

Efektivitas & Efisiensi

Akibat dari algoritma bot ini yang mengecek jarak suatu bot dan memilih target dengan jarak terpendek, rata-rata bot harus mengecek dengan kompleksitas waktu $120/360 * n = \frac{1}{3} n \in O(n)$. Namun, karena pergerakan bot yang bersifat aktif, bot hanya memerlukan waktu yang singkat untuk menyelesaikan ronde, karena risiko bot ditembak atau ditabrak adalah sangat besar.

Strategi yang Dipilih dan Alasan

Pada pertempuran *battle royale*, kami memilih strategi pada bot Tembok karena bot ini dinilai memiliki pertahanan yang cukup baik dalam pertempuran yang ramai dengan bot-bot lain. Meskipun bot ini tidak cukup efektif dalam 1v1, namun dalam pertempuran yang melibatkan banyak bot, Tembok terbukti cukup baik seperti yang ditunjukkan pada bagian pengujian.

Bab 4

Implementasi dan Pengujian

Alternatif 1: Wukong

```
// Inisialisasi variabel utama
moveDirection ← 1
consecutiveHits ← 0
lastEnemyDistance ← 0

// Main loop selama bot berjalan
while (bot is running):
    TurnRight(45)                // Gerakan memutar untuk mengubah
    arah
    Forward(100)                 // Maju sejauh 100 unit
    Execute movement command    // Menjalankan perintah
    pergerakan

// Ketika bot mendeteksi musuh
on ScannedBotEvent(e):
    TurnToFaceTarget(e.X, e.Y)   // Menyesuaikan
    arah agar menghadap musuh
    lastEnemyDistance ← DistanceTo(e.X, e.Y) // Simpan jarak
    ke musuh terakhir terdeteksi
    bearing ← BearingTo(e.X, e.Y)
    absoluteBearing ← current Direction + bearing
    gunAdjust ← NormalizeBearing(absoluteBearing -
    GunDirection)
    TurnGunRight(gunAdjust)      // Menyesuaikan posisi meriam
    untuk menghadap musuh
    Fire(3)                      // Menembak dengan kekuatan 3
    if (lastEnemyDistance > 200):
```



```

        TurnRight(bearing) // Mengubah arah jika musuh terlalu
jauh

        Forward(100)        // Maju untuk mendekat

        Rescan()            // Memperbarui pemindaian musuh

// Ketika terkena peluru
on HitByBulletEvent(e):
    if (consecutiveHits > 2):
        Back(150)            // Mundur jauh untuk
menghindari serangan beruntun
        TurnRight(180 - Direction) // Mengubah arah secara
drastis
        consecutiveHits ← 0    // Reset hit counter
    else:
        bulletBearing ← NormalizeBearing(e.Bullet.Direction -
Direction)
        TurnLeft(90 - bulletBearing) // Menyesuaikan sudut
untuk menghindar
        Forward(100)          // Maju untuk mengubah
posisi
        Rescan()              // Memperbarui pemindaian

// Ketika mengenai dinding
on HitWallEvent(evnt):
    SetForward(200 * moveDirection) // Mengubah arah
berdasarkan variabel moveDirection
    Execute movement command        // Menjalankan perintah
pergerakan

// Fungsi untuk menormalkan sudut (agar berada dalam rentang
0-360 derajat)
function NormalizeBearing(bearing):
    return (bearing + 360) mod 360

```

STRATEGI GREEDY:

- Prioritaskan tembakan cepat (power 3)
- Optimalkan damage per detik
- Menghindar hanya jika menerima damage

Alternatif 2: Erlang

```
// Inisialisasi variabel utama
orbitDirection ← 1

// Konfigurasi warna bot (tema putih-abu dan grey)
SetColor(body, turret, radar, bullet, scan)

// Main loop selama bot berjalan
while (bot is running):
    Forward(1000)           // Maju jauh untuk eksplorasi area
    TurnLeft(90)            // Memutar ke kiri untuk mengubah
    jalur
    SetTurnRadarRight(Infinite) // Memutar radar tanpa henti
    untuk pemindaian terus menerus

// Ketika bot mendeteksi musuh
on ScannedBotEvent(e):
    bearing ← BearingTo(e.X, e.Y)
    distance ← DistanceTo(e.X, e.Y)

    // Jika musuh terlalu dekat, lakukan manuver menghindar
    if (distance < 250):
        orbitDirection ← orbitDirection * -1 // Balik arah
        orbit
        TurnLeft(90)                          // Memutar
        untuk mengubah arah
```

```

        Back(100)                                // Mundur
sedikit

    // Penyesuaian meriam untuk mengincar musuh
    absoluteBearing ← current Direction + bearing
        gunAdjust ← NormalizeBearing(absoluteBearing -
GunDirection)
    TurnGunRight(gunAdjust)

    // Menembak dengan daya kecil jika musuh dekat
    if (distance < 200):
        Fire(1)

    Rescan() // Memperbarui pemindaian

    // Menjaga jarak aman dengan berputar berdasarkan
orbitDirection
    TurnRight(NormalizeBearing(bearing + 90 * orbitDirection))

// Ketika bertabrakan dengan bot lain
on HitBotEvent(e):
    Back(200)                // Mundur untuk menghindari
    TurnLeft(45)             // Berubah arah
    orbitDirection ← orbitDirection * -1 // Balik arah orbit
    Rescan()

// Ketika terkena peluru
on HitByBulletEvent(e):
    bulletBearing ← NormalizeBearing(e.Bullet.Direction -
Direction)
    TurnLeft(90 - bulletBearing) // Mengubah arah untuk
menghindar peluru

```

```

        MaxSpeed ← 8                                // Menaikkan kecepatan
sementara
        Forward(150)                                // Maju untuk menghindari
        MaxSpeed ← 5                                // Mengembalikan kecepatan
semula

// Ketika mengenai dinding
on HitWallEvent(evnt):
        SetTurnLeft(45)                             // Berputar ke kiri untuk
menghindari dinding
        orbitDirection ← orbitDirection * -1 // Balik arah orbit
        SetForward(2000 * orbitDirection)          // Maju dengan
jarak jauh sesuai arah orbit

// Fungsi untuk menormalkan sudut (agar berada dalam rentang
0-360 derajat)
function NormalizeBearing(bearing):
        return (bearing + 360) mod 360

```

STRATEGI GREEDY:

- Prioritaskan bertahan hidup
- Tembak hanya dalam kondisi optimal
- Kelola energi untuk pertahanan

Alternatif 3: Tembok

```

//Inisialisasi Variabel Penting
WALL MARGIN ← 30

//Putar tank ke arah 0 derajat
IF Direction <= 180 THEN
        PUTAR KANAN sebesar Direction
ELSE
        PUTAR KIRI sebesar (360-Direction)

```

```

END IF

// Maju sampai ke jarak margin dari tembok
MAJU sebanyak (Lebar Arena - X - WALL_MARGIN) unit

// Putar pistol agar menghadapi arena
PUTAR PISTOL ke KIRI sebesar 90 derajat

// Putar bodi tank lalu mulai maju ke pojok kanan atas
PUTAR ke KIRI sebesar 90 derajat
MAJU sebesar (Tinggi Arena - Y - WALL_MARGIN) unit

// Loop pergerakan tank
WHILE IsRunning DO
    // Repeat the patrol around the arena
    peek ← FALSE
    PUTAR ke KIRI sebesar 90 derajat
    peek ← TRUE
    MAJU sebesar (Lebar Arena - 2 * WALL_MARGIN) unit

    peek ← FALSE
    PUTAR ke KIRI sebesar 90 derajat
    peek ← TRUE
    MAJU sebesar (Tinggi Arena - 2 * WALL_MARGIN) unit

    peek ← FALSE
    PUTAR ke KIRI sebesar 90 derajat
    peek ← TRUE
    MAJU sebesar (Lebar Arena - 2 * WALL_MARGIN) unit

```

```

    peek ← FALSE
    PUTAR ke KIRI sebesar 90 derajat
    peek ← TRUE
    MAJU sebesar (Tinggi Arena - 2 * WALL_MARGIN) unit
END WHILE

EVENT PEMINDAIAN MUSUH:
    JIKA terdeteksi musuh:
        TEMBAK dengan kekuatan 2
    JIKA peek adalah TRUE:
        PINDAI ULANG
    ENDIF

EVENT TABRAKAN:
    bearing ← Arah ke Koordinat X & Y lawan
    IF bearing > -90 AND bearing < 90 THEN
        MUNDUR 100 unit
    ELSE // If the object is behind, move forward
        MAJU 100 unit
    END IF

```

Strategi greedy :

- Memiliki pergerakan yang sangat aktif walaupun konstan.
- Menembak bot lain hanya jika suatu bot terlihat pada radar.
- Menghindari bot dengan cara menjauhi bot apabila tertabrak, sehingga mengurangi kemungkinan bot lain memperoleh poin dari tabrakan.

Alternatif 4: Wintermute

```

BEGIN
    count ← 0
    gunTurnAmt ← 0

```

```

        //Putar tank ke pusat arena

        directionToCenter ← Arah ke koordinat (ArenaWidth / 2,
        ArenaHeight / 2)

        rotAtm ← SELISIH SUDUT antara directionToCenter dan
        Direction

        PUTAR ke KIRI sebesar rotATM derajat

    WHILE IsRunning DO

        // Besar derajat penyapuan radar tiap waktu
        gunTurnAmt ← 60

        // Putar bot apabila tidak terdeteksi bot apapun
        IF count > 1 THEN
            PUTAR ke KIRI sebesar 120 derajat
            count ← 0
        END IF

        // Scan area with the radar
        PUTAR RADAR ke KIRI sebesar gunTurnAmt
        PUTAR RADAR ke KANAN sebesar gunTurnAmt * 2
        PUTAR RADAR ke KIRI sebesar gunTurnAmt

        count ← count + 1
    END WHILE
END

EVENT PEMINDAIAN MUSUH evt :

    trackedDir ← 0

    IF Jarak ke koordinat (evt.X, evt.Y) > 100 THEN
        trackedDir ← ARAH ke koordinat (evt.X, evt.Y)
        rotAtm ← SELISIH SUDUT antara trackedDir & Direction
    
```

```

    PUTAR ke KIRI sebesar rotAtm derajat
    MAJU sebesar 100 unit
    TEMBAK dengan damage 2
ELSE
    MUNDUR sebesar 50 unit
    trackedDir ← ARAH ke koordinat (evt.X, evt.Y)
    rotAtm ← SELISIH SUDUT antara trackedDir & Direction
    PUTAR ke KIRI sebesar rotAtm derajat

    timesFire ← Floor(evt.Energy / 3)
    WHILE timesFire > 0 DO
        TEMBAK dengan damage 3
        timesFire ← timesFire - 1
    END WHILE
END IF
END EVENT

EVENT TERKENA PELURU evt
    BELOK KIRI sebesar 90 derajat
    MUNDUR 100 unit
END EVENT

EVENT TABRAKAN e
    DECLARE collisionDir, turnAmt

    collisionDir ← ARAH ke koordinat (e.X, e.Y)
    turnAmt ← SELISIH SUDUT antara collisionDir & Direction

    PUTAR ke KIRI sebesar turnAmt derajat
    TEMBAK dengan damage 2
END FUNCTION

```


EVENT TABRAK TEMBOK evt

PUTAR ke KANAN sebesar 180 derajat

MAJU 300 unit

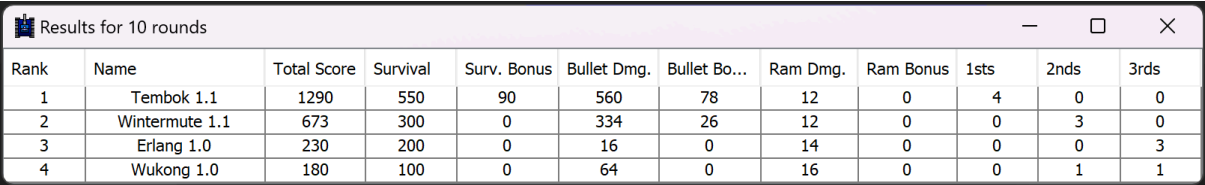
END EVENT

Strategi greedy :

- Memutar bot sesering mungkin untuk mencari target
- Mencari target bot pada sudut 120 derajat, yang memberikan waktu memindai dengan cepat sekaligus jangkauan yang cukup
- Menembak dengan damage medium apabila jarak target cukup jauh untuk menghemat energi
- Menembak dengan damage maksimum apabila target cukup dekat untuk memprioritaskan perolehan poin.
- Memilih target dengan jarak terdekat pada tiap waktu untuk memperbesar kemungkinan sukses ditembak

Analisis Hasil Pengujian

Berikut adalah hasil pengujian dari pertarungan 4 bot tersebut.



Rank	Name	Total Score	Survival	Surv. Bonus	Bullet Dmg.	Bullet Bo...	Ram Dmg.	Ram Bonus	1sts	2nds	3rds
1	Tembok 1.1	1290	550	90	560	78	12	0	4	0	0
2	Wintermute 1.1	673	300	0	334	26	12	0	0	3	0
3	Erlang 1.0	230	200	0	16	0	14	0	0	0	3
4	Wukong 1.0	180	100	0	64	0	16	0	0	1	1

Gambar 1 : Peroleh skor pertarungan 4 bot

Terlihat bahwa ranking bot adalah Tembok, Wintermute, Wukong, dan Erlang. Dapat disimpulkan strategi greedy bot Tembok berhasil mendapatkan solusi optimal. Ada alasan-alasan mengapa bot-bot lainnya gagal memperoleh solusi optimal. Pertama, tentunya karena adanya cacat logika akibat kurangnya pemahaman pengguna terhadap API program. Kedua, karena program yang terlalu kompleks menyebabkan bot tidak cukup cepat untuk merespons situasi yang simultan dan instan. Ketiga, pentingnya kesinkronan antara strategi gerakan, strategi radar, dan strategi menembak agar bot dapat memiliki performa lebih optimal dalam pertempuran. Keempat, desain algoritma yang diimplementasikan pada bot harus sesuai dengan kondisi pertarungan (dalam hal ini Battle Royale), karena kondisi pertarungan yang berbeda akan mengungguli sifat algoritma yang berbeda. Misalnya, mode pertempuran 1v1 tentunya akan berbeda dengan tim dan berbeda juga dengan Battle Royale.

Selanjutnya, akan dijabarkan kemungkinan kondisi-kondisi bot lainnya tidak berhasil mendapatkan nilai optimal Karena sifat bot itu sendiri, bot Wintermute akan kalah dengan bot

yang aktif bergerak seperti bot Tembok. Ini karena bot Wintermute akan memerlukan pencarian lokasi dari radar, lalu menembak lokasi tersebut. Jelas, bot yang sering bergerak dapat kabur dari tembakan bot Wintermute yang hanya akan menembak pada lokasi awal saat bot Wintermute mendeteksi lokasi bot lainnya. Selain itu, sifat bot Wintermute yang selalu menembak bot yang terdeteksi akan menghabiskan energi dengan boros.

Adapun bot-bot seperti Wukong dan Erlang tidak memiliki performa yang cukup baik karena kurangnya keharmonisan antara strategi gerakan dan strategi menyerang ataupun bertahan yang telah didesain pada masing-masing bot. Contohnya, pada bot Wukong, meskipun telah didesain algoritma greedy untuk menyerang, serangan yang dihasilkan belum akurat karena pengaturan “Gun” belum cukup baik seiring dengan gerakan tank, lalu pada bot Erlang, belum cukup baik dalam menghindari serangan karena manuvernya terbatas terhadap jarak dan respons terhadap serangan dan tabrakan saja.

Bab 5

Kesimpulan dan Saran

Kesimpulan

Berdasarkan pengujian dan analisis yang dilakukan, dapat disimpulkan bot Tembok adalah bot yang memiliki algoritma greedy yang paling optimal dalam memaksimalkan fungsi objektifnya dalam mode pertempuran Battle Royale karena memiliki metode bertahan yang cukup baik dan menyerang yang efisien.

Saran

Beberapa saran yang dapat dipertimbangkan untuk pengembangan lebih lanjut, yaitu:

1. Dokumentasi

Disarankan bagi yang ingin melanjutkan membuat bot yang optimal untuk mengeksplor API dari bot ini. Implementasi bot sebelumnya memang terkesan sederhana dan hanya menggunakan sedikit dari fungsi-fungsi yang tersedia. Maka, diperlukan pemahaman yang kuat dari mekanisme dan cara kerja program Robocode untuk membuat sebuah bot yang optimal.

Sebagai contoh, dalam web API Robocode Tank Royale, sebenarnya sudah diberikan strategi-strategi yang cukup menarik, misalnya seperti virtual bullet, yakni mensimulasikan serangan dari lawan berdasarkan energi serta posisi yang dimilikinya untuk memprediksi serangannya sehingga meningkatkan akurasi dalam bot yang greedy dalam pertahanan seperti Erlang. Namun, karena keterbatasan kemampuan dan waktu untuk mengeksplorasi, ide tersebut tidak cukup maksimal untuk diaplikasikan dalam proyek kali ini.

2. Pengujian

Dengan melakukan pengujian terhadap bot-bot yang memiliki algoritma yang lebih canggih, dapat diperoleh inspirasi serta inovasi terhadap bot-bot yang sudah ada.

Bab 6 : Lampiran

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

Pranala

Kode dapat diakses di : https://github.com/countz-zero/Tubes1_deepsick

Video demonstrasi dapat diakses di : <https://youtu.be/QFeFC1UhdPQ>

Daftar Pustaka

- Robocode Tank Royale* (2025) *Robocode Tank Royale* | *Robocode Tank Royale Docs*.
<https://robocode-dev.github.io/tank-royale/>
- Munir, R. (2025). *Algoritma Greedy part 1* [slide PowerPoint]
[https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2024-2025/04-Algoritma-Greedy-\(2025\)-Bag1.pdf](https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2024-2025/04-Algoritma-Greedy-(2025)-Bag1.pdf)
- Dokumentasi Robocode Tank Royale

