Laporan Tugas Besar 1 IF2211 Strategi Algoritma

Semester II tahun 2024/2025

Pemanfaatan Algoritma Greedy dalam Pembuatan Bot Permainan Robocode Tank Royale

Oleh:

Yosef Rafael Joshua	(13522133)
Asybel B.P. Sianipar	(15223011)
Ignacio Kevin Alberiann	(15223090)

PROGRAM STUDI TEKNIK INFORMATIKA

SEKOLAH TEKNIK ELEKTRO DAN INFORMATIKA

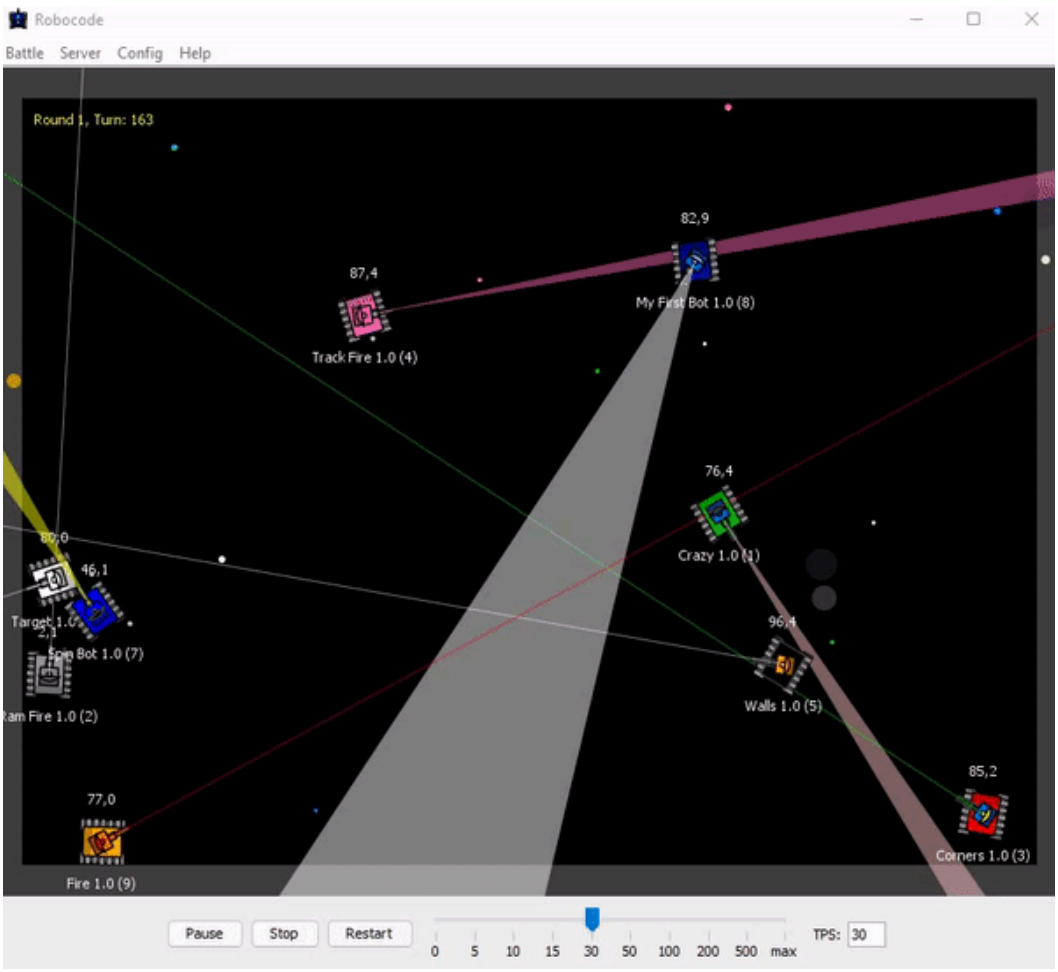
INSTITUT TEKNOLOGI BANDUNG

2025

DAFTAR ISI

BAB I **1**
BAB II **4**
2.1 Algoritma Greedy 4
2.2 Cara Kerja Program 5
2.2.1 Aksi yang Dilakukan Bot 5
2.2.2 Mengimplementasikan Algoritma Greedy ke dalam Bot 6
2.2.3 Menjalankan Bot 6
BAB III **8**
3.1. Mapping Persoalan Robocode Tank Royale 8
1. Himpunan Kandidat 9
2. Himpunan Solusi 10
3. Fungsi Seleksi 10
4. Fungsi Kelayakan 10
5. Fungsi Solusi 11
6. Fungsi Objektif 11
3.2. Eksplorasi Kemungkinan Solusi 12
3.2.1. Non-Greedy 12
3.2.2 Greedy 12
3.3. 4 Alternatif Solusi Greedy 13
BAB IV **26**
4.1. Implementasi 4 Alternatif Solusi 26
4.1.1. Alternatif 1: ProtokolKesehatan 26
4.1.2. Alternatif 2: WallHugger.exe 30
4.1.3. Alternatif 3: SBS 33
4.1.4. Alternatif 4: YangPentingNembak 34
4.2. Penjelasan Struktur Data, Fungsi, dan Prosedur Strategi Terpilih (ProtokolKesehatan) 35
4.3. Pengujian 37
4.4. Analisis Hasil Pengujian 41
BAB V **44**
5.1 Kesimpulan 44
5.2 Saran 44
DAFTAR PUSTAKA **45**
LAMPIRAN **46**
1. Link Repository GitHub 46
2. Link Video 46
3. Tabel Ketercapaian 46

BAB I
DESKRIPSI TUGAS



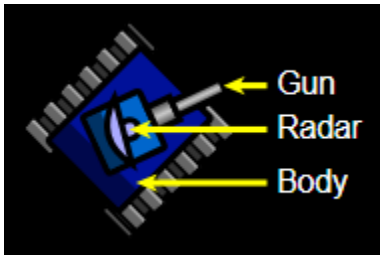
Gambar 1.1 Robocode Tank Royale

Robocode adalah permainan pemrograman yang bertujuan untuk membuat kode bot dalam berlangsung hingga bot-bot bertarung hanya tersisa satu seperti permainan Battle Royale, karena itulah permainan ini dinamakan Tank Royale. 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 utama permainan Robocode Tank Royale:

1. Ronde & Giliran
    - 10 ronde per pertempuran. Setiap ronde terdiri dari giliran (unit waktu terkecil).
    - Bot dapat bergerak, memindai, menembak, atau bereaksi per giliran.
    - Batas waktu giliran (30-50 ms) : Bot yang terlambat mengirim perintah melewatkan giliran.
    - API otomatis mengirim perintah; tidak perlu dikelola kecuali membuat API sendiri.
2. Energi
    - Awal: 100 poin. Berkurang saat tertembak, menembak, atau tabrakan.
    - Bertambah 3x lipat energi tembakan jika peluru mengenai musuh.
    - Energi nol: Bot dinonaktifkan; hancur jika terkena serangan lagi.
3. Peluru
    - Peluru berat: Lambat, damage tinggi, panas meriam lebih besar.
    - Peluru ringan: Cepat, damage rendah, panas rendah.
    - Panas meriam : Menembak saat panas menyebabkan cooldown. Meriam mulai panas di awal ronde.
4. Tabrakan
    - Tabrakan dengan dinding/bot musuh menyebabkan damage.
    - Menabrak musuh (ramming) memberi bonus skor.
5. Bagian Tank



- **Body :** Bergerak maju/mundur, kecepatan maksimal dengan percepatan/perlambatan 1-2 unit/giliran.
- **Gun :** Berputar hingga 20°/giliran, independen dari body.
- **Radar :** Berputar 45°/giliran, deteksi musuh hingga 1200 piksel. Harus bergerak untuk memindai.

6. Pemindaian

   Radar hanya mendeteksi musuh dalam jangkauan sudut (scan arc). Arah radar harus diubah tiap giliran untuk efektivitas.

7. Skor

   Komponen :

   - Bullet Damage (damage ke musuh).
   - Bullet Damage Bonus (20% damage jika musuh mati).
   - Survival Score (50 poin/bot yang bertahan saat musuh mati).
   - Last Survival Bonus (10 poin × jumlah musuh jika jadi yang terakhir).
   - Ram Damage (2× damage tabrakan) dan Ram Bonus (30% damage jika musuh mati).
   - Skor akhir: Total semua komponen. Peringkat berdasarkan skor tertinggi.

Catatan : Game engine tidak mengikuti aturan default Robocode untuk ronde/giliran.

BAB II
LANDASAN TEORI

2.1 Algoritma Greedy

Algoritma Greedy merupakan jenis algoritma yang sering digunakan dalam proses optimasi. Algoritma ini bekerja dengan cara memilih opsi yang dianggap paling optimal pada saat tertentu, tanpa mempertimbangkan dampak atau konsekuensi di masa depan. Berbeda dengan pendekatan konservatif yang lebih berfokus pada efek jangka panjang, algoritma greedy mengutamakan keuntungan langsung. Biasanya, algoritma ini diterapkan melalui pendekatan heuristik, yaitu dengan menetapkan parameter tertentu yang menjadi prioritas atau dasar untuk menentukan solusi terbaik pada setiap langkahnya, tergantung pada masalah yang dihadapi.

Karena sifatnya yang selalu mengambil keputusan berdasarkan kondisi saat itu, algoritma heuristik yang konsisten memilih opsi paling optimal pada momen tertentu dapat digolongkan sebagai algoritma greedy. Meskipun demikian, algoritma ini tidak selalu menjamin solusi yang benar-benar optimal secara keseluruhan. Namun, greedy sering kali mampu menghasilkan solusi yang optimal secara lokal, yang dalam beberapa kasus bisa mendekati solusi optimal global, terutama jika diberikan waktu dan data yang memadai. Dengan kata lain, keefektifan algoritma greedy sangat bergantung pada konteks masalah dan bagaimana parameter optimasinya didefinisikan.

Komponen utama dalam algoritma greedy:

- **Himpunan Kandidat:** berisi kandidat yang akan dipilih pada setiap langkah.
- Himpunan Solusi: berisi kandidat yang sudah dipilih.
- **Fungsi Seleksi:** memilih kandidat berdasarkan strategi greedy tertentu. Strategi greedy ini bersifat heuristik.
- **Fungsi Kelayakan:** memeriksa apakah kandidat yang dipilih dapat dimasukkan ke dalam himpunan solusi (layak atau tidak).
- **Fungsi Solusi:** menentukan apakah himpunan kandidat yang dipilih sudah memberikan solusi.
- **Fungsi Objektif:** memaksimumkan atau meminimumkan suatu fungsi.

2.2 Cara Kerja Program

2.2.1 Aksi yang Dilakukan Bot

Bot pada permainan Robocode Tank Royale dapat menjalankan beberapa aksi seperti:

Aksi	Method API
Mundur	Back(double), SetBack(double)
Maju	Forward(double), SetForward(double)
Tembak	Fire(double)
Pindai Ulang	Rescan()
Melanjutkan Pergerakan	Resume()
Menjalankan Bot	Run()
Putar meriam ke kiri	SetTurnGunLeft(double), TurnGunLeft(double)
Putar meriam ke kanan	SetTurnGunRight(double), TurnGunRight(double)
Putar radar ke kiri	SetTurnRadarLeft(double), TurnRadarLeft(double)
Putar radar ke kanan	SetTurnRadarRight(double), TurnRadarRight(double)
Putar bot ke kanan	SetTurnRight(double), TurnRight(double)
Menunggu kondisi	WaitFor(Condition)
Berhenti	Stop()

Memerintahkan bot untuk menjalankan suatu aksi dapat dilakukan dengan cara memanggil method API yang bersesuaian dengan aksi yang ingin dijalankan. Pada permainan Robocode Tank Royale, bot akan menjalankan aksi setiap giliran. Masing-masing aksi membutuhkan waktu (giliran) yang berbeda sehingga dapat dijalankan hingga selesai, seperti yang telah disebutkan pada bagian Deskripsi Tugas. Terkadang bot akan menghabiskan beberapa giliran hanya untuk melakukan suatu aksi.

2.2.2 Mengimplementasikan Algoritma Greedy ke dalam Bot

Implementasi algoritma Greedy pada bot permainan Robocode Tank Royale dapat dilakukan dengan membuat folder baru yang berisi file .cs tempat logic bot diimplementasikan. Pada file ini harus dibuat sebuah kelas baru yang melakukan inheritance dari kelas "Bot" pada API Robocode Tank Royale. Pada kelas baru ini harus dibuat instance yang menjalankan "Start()" method. Implementasi logika bot dan sekaligus algoritma Greedy bot dilakukan dengan melakukan method overriding pada method "Run()". Di dalam method inilah tempat menuliskan kode algoritma Greedy yang ingin diimplementasikan.

Mengimplementasikan algoritma Greedy pada bot membutuhkan informasi mengenai bot (IsRunning, GunTurnRate, Energy), arena pertandingan (ArenaHeight, ArenaWidth), dan musuh (Energy, GunTurnRate, TurnRate). Informasi mengenai bot sendiri ataupun tentang arena pertandingan dapat langsung diakses dari kelas yang dibuat karena semua informasi tersebut ada di kelas parent. Namun untuk informasi mengenai musuh hanya bisa didapatkan ketika bot memindai musuh menggunakan radar. "OnScannedBot()" adalah method event handler yang menangani kejadian ketika bot memindai musuh. Method ini dapat di-override untuk mendapatkan informasi musuh yang dibutuhkan. Selain itu pada method ini juga dapat diimplementasikan logika tambahan yang akan dijalankan bot ketika memindai musuh.

Selain melakukan overriding terhadap method "Run()" dan "OnScannedBot()", method event handler lain seperti OnBulletHit(), OnHitWall(), OnHitBot() juga dapat dimodifikasi dengan cara method overriding untuk menambah aksi yang dapat dilakukan oleh bot dan sebagai tempat mengimplementasi algoritma Greedy. Algoritma Greedy bot dapat diimplementasikan sesuai kebutuhan, dan tujuan bot.

2.2.3 Menjalankan Bot

Bot Robocode Tank Royale dijalankan pada game engine tersendiri. Game engine dapat diunduh pada tautan https://github.com/robocode-dev/tank-royale. Untuk tugas besar ini digunakan versi yang sudah dimodifikasi

https://github.com/Ariel-HS/tubes1-if2211-starter-pack/releases/tag/v1.0. Namun cara menjalankan bot tetap sama. Game engine yang sudah diunduh perlu dijalankan terlebih dahulu. Caranya adalah dengan masuk ke direktori "tank-royale-0.30.0" dengan perintah

```
cd tank-royale-0.30.0
```

Kemudian executable .jar perlu di-build dan dijalankan. Untuk itu dapat menggunakan perintah

```
./gradlew :gui-app:clean
```

Executable .jar dapat dijalankan dengan perintah

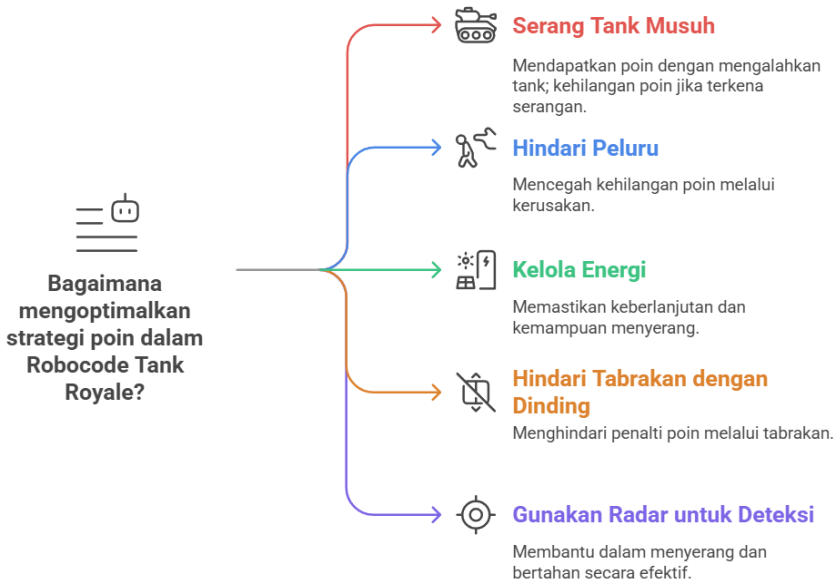
```
java -jar
./gui-app/build/libs/robocode-tankroyale-gui-0.30.0.jar
```

Setelah game engine sudah berjalan, direktori folder berisi bot yang telah dibuat perlu ditambahkan sebelum dapat dijalankan. Direktori dapat ditambahkan ke game engine dengan cara memilih menu "Config" pada aplikasi kemudian memilih "Bot Root Directories". Langkah selanjutnya adalah membuat pertandingan dan menambahkan bot ke pertandingan tersebut. Cara membuat pertandingan adalah dengan memilih menu "Battle" kemudian "Start Battle". Pada layar ini bot yang sudah ditambahkan direktorinya dapat dimasukkan ke dalam pertandingan. Bot harus di-boot terlebih dahulu. Cara melakukan boot up bot adalah dengan memilih direktori bot yang ada di kotak kiri atas kemudian memilih "Boot". Semua bot yang di-boot akan terlihat pada kotak kanan atas. Bot yang berhasil di-boot dan siap dimasukkan ke pertandingan akan muncul pada kotak kiri bawah. Pada kotak ini dapat dipilih bot mana saja yang akan dimasukkan ke pertandingan. Caranya adalah dengan memilih bot pada kotak kiri bawah dan memilih "Add". Semua bot yang berhasil dimasukkan ke pertandingan akan muncul di kotak kanan bawah. Setelah semua bot yang diinginkan masuk ke pertandingan, pertandingan dapat dimulai dengan memilih "Start Battle" di bagian bawah.

BAB III
APLIKASI STRATEGI GREEDY

3.1. *Mapping* Persoalan *Robocode Tank Royale*

Dalam permainan Robocode Tank Royale, strategi greedy diterapkan untuk mengoptimalkan keputusan bot dalam setiap langkah demi memaksimalkan poin. Strategi ini bekerja dengan memilih aksi terbaik secara lokal berdasarkan kondisi saat itu.



Gambar 3.1 Diagram Strategi Permainan Robocode
(Sumber: kelompok lima_empat)

Strategi dibuat dengan mempertimbangkan seluruh komponen utama yang ada di dalam permainan. Rangkuman komponen utama permainan:

No	Komponen Teknis Permainan	Kegunaan	Pengaruh Terhadap Poin dan Energi
1	Energy	Sumber daya untuk menembak dan bergerak. - Energi awal = 100 - Energi = 0 dan terkena damage → tank mati	Manajemen energi diperlukan agar bot dapat bertahan di arena. Poin: + 50 (setiap ada bot lain yang mati).

2	Enemy Tanks	Objek utama yang menjadi target untuk diserang atau dihindari	Mengalahkan tank musuh memberikan poin; dihantam musuh mengurangi poin. Poin: + 2 x *damage* (*ramming*, penyerang) + 20% x *damage* (musuh mati dari *ramming*)
3	Gun & Bullets	*Bullets* 'meriam' adalah ancaman dari tank musuh yang dapat mengurangi energi bot.	Menghindari peluru mencegah kehilangan poin akibat kerusakan Poin: + *firepower* (mengenai musuh) + 20% x *firepower* (musuh mati dari tembakan) Energi: + *3* x *firepower* (mengenai musuh) - *firepower* (menembakkan meriam atau terkena tembakan)
4	Walls	Batas arena yang harus dihindari	Menghindari tabrakan mencegah pengurangan energi. Energi: - *wall damage* **(**menabrak dinding)
5	Radar	Mendeteksi posisi tank musuh di arena	Deteksi musuh membantu bot menyerang dan bertahan.

Mapping komponen greedy dalam permainan Robocode Tank Royale:

1. Himpunan Kandidat

- Bot memiliki beberapa opsi tindakan yang dapat diambil oleh bot pada setiap putaran permainan, seperti:
    - Gerakan: Maju atau mundur sejauh jarak tertentu (misalnya, maju 100 unit, mundur 50 unit).

    - Rotasi: Memutar badan, senjata, atau radar pada sudut tertentu (misalnya, putar kiri 45°, putar kanan 30°).
    - Tembakan: Menembak dengan kekuatan tertentu (misalnya, kekuatan 1, 2, atau 3).
    - Kombinasi: Melakukan beberapa tindakan sekaligus, seperti bergerak sambil memutar senjata.
- Contoh: Pada suatu putaran, himpunan kandidat bisa berupa {maju 100 unit, putar kiri 45°, tembak dengan kekuatan 2, putar senjata kanan 10°}.

2. Himpunan Solusi

- Karena permainan berlangsung secara kontinu, himpunan solusi biasanya adalah tindakan yang dipilih untuk putaran tersebut, yang kemudian diperbarui pada putaran berikutnya.
- Contoh: Jika bot memilih untuk "maju 100 unit" dan "tembak dengan kekuatan 2", maka himpunan solusi untuk putaran ini adalah {maju 100 unit, tembak dengan kekuatan 2}.

3. Fungsi Seleksi

- Fungsi ini bisa berbeda tergantung pada prioritas bot (menyerang, bertahan, atau mengelola energi). Beberapa contoh heuristik:
    - Untuk Gerakan: Memilih arah yang memaksimalkan jarak dari musuh terdekat atau meminimalkan risiko tertembak.
    - Untuk Penargetan: Memilih musuh yang paling dekat, memiliki energi terendah, atau paling mengancam.
    - Untuk Tembakan: Menembak jika senjata sejajar dengan target.
- Contoh: Jika bot ingin menghindari musuh, fungsi seleksi memilih arah gerakan yang menjauhkan bot dari musuh terdekat. Jika bot fokus menyerang, fungsi seleksi memilih musuh dengan energi terendah sebagai target.

4. Fungsi Kelayakan

- Dalam Konteks Robocode Tank Royale: Fungsi ini memeriksa:
    - Gerakan: Bot tidak menabrak dinding atau keluar dari arena.

    - Tembakan: Bot memiliki energi yang cukup untuk menembak (misalnya, tembakan kekuatan 3 membutuhkan energi minimal 3).
    - Rotasi: Rotasi tidak melebihi batas mekanis bot.
- Contoh: Jika bot memilih maju 100 unit tetapi hanya berjarak 20 unit dari dinding, tindakan ini tidak layak. Jika energi bot kurang dari 2, tembakan dengan kekuatan 2 juga tidak layak.

5. Fungsi Solusi

- Dikarenakan permainan bersifat kontinu, fungsi solusi biasanya terpenuhi ketika bot telah memilih tindakan untuk putaran tersebut. Tidak ada "solusi akhir" kecuali permainan selesai.
- Contoh: Setelah memilih "memutar senjata ke musuh terdekat" dan "tembak jika sejajar", bot menganggap keputusan untuk putaran ini selesai dan melanjutkan ke putaran berikutnya.

6. Fungsi Objektif

- Definisi: Fungsi objektif adalah tujuan utama yang ingin dicapai oleh bot melalui serangkaian keputusan.
- Dalam Konteks Robocode Tank Royale: Tujuan umum bot adalah memaksimalkan poin, yang biasanya didapat dari:
    - Memberikan damage kepada musuh.
    - Bertahan hidup selama mungkin.
    - Mengeliminasi musuh.
- Contoh: Bot yang agresif mungkin memiliki fungsi objektif untuk memaksimalkan damage yang diberikan, sementara bot defensif mungkin fokus meminimalkan damage yang diterima.

3.2. Eksplorasi Kemungkinan Solusi

Secara umum, permainan Robocode Tank Royale dapat diimplementasikan dengan strategi yang beragam baik non-*greedy* maupun *greedy*.

3.2.1. Non-Greedy

- **Reinforcement Learning (RL)**

  Pendekatan ini memungkinkan bot belajar dari pengalaman melalui trial-and-error. Bot mencoba berbagai aksi dan menerima reward berdasarkan hasilnya, seperti energi yang diperoleh atau kerusakan yang dihindari. Seiring waktu, bot mengembangkan strategi optimal.

  Contoh: Menggunakan Q-Learning, bot membangun tabel Q yang memetakan state (posisi musuh, energi bot, dll.) ke aksi terbaik (menembak, bergerak, atau menghindar).

- **Dynamic Clustering**

  Dynamic clustering di Robocode adalah teknik berbasis K-nearest neighbor yang memanfaatkan situasi masa lalu untuk membantu robot membuat keputusan cerdas di tengah pertempuran. Robot menyimpan informasi tentang situasi yang pernah dihadapi, seperti posisi musuh, arah gerakan, atau pola tembakan. Ketika menghadapi situasi baru, robot membandingkan kondisi saat ini dengan data masa lalu untuk menemukan kemiripan.

- **Guess Factor**

  Algoritma ini digunakan untuk menghasilkan sudut tembakan yang baik secara statistik. Bot dapat merekam data seperti pergerakan musuh terlebih dahulu dan mengolahnya menjadi histogram untuk analisis statistik.

3.2.2 Greedy

- **Greedy by Position or Movement**

Bot dapat memilih posisi yang meminimalkan risiko terkena tembakan dengan menghitung posisi musuh dan peluru, lalu bergerak ke arah paling aman. Sebaliknya, bot juga dapat bergerak mendekati musuh untuk meningkatkan akurasi (jika diperlukan).

Contoh: MinimumRiskMovement

- **Greedy by Energy Management (Gain or Saving)**

  Mengoptimalkan penggunaan energi untuk aksi yang memberikan keuntungan maksimal (misal: menembak) atau menghemat energi untuk situasi kritis. Bot menghitung biaya energi untuk setiap aksi (bergerak dan menembak).

  Contoh: Menembak musuh dengan kekuatan peluru yang disesuaikan untuk memaksimalkan damage per energi.

- **Greedy by Gun (Bullet Hit or Firing Frequency)**

  Meningkatkan peluang serangan langsung dengan memaksimalkan *hit rate* (persentase tembakan yang mengenai musuh) atau *damage* untuk memaksimalkan poin.

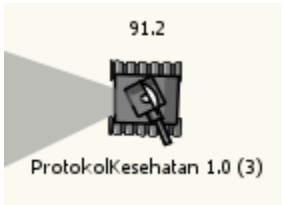
  Contoh: Menembak agresif, bahkan dengan peluru berenergi tinggi.

- **Greedy by Targeting**

  Memilih musuh mana yang akan ditargetkan, seperti selalu menargetkan musuh terlemah atau terdekat. Namun, ini tampaknya sudah tercakup dalam "Greedy by Bullet Hit," karena memilih target memengaruhi hit rate dan damage.

3.3. 4 Alternatif Solusi Greedy

3.3.1. ProtokolKesehatan



a. Deskripsi umum

ProtokolKesehatan adalah bot yang dirancang untuk Robocode Tank Royale dengan strategi berbasis algoritma greedy yang fokus pada penjagaan jarak dari musuh. Bot ini selalu bergerak menuju posisi di arena yang memiliki risiko minimum. Risiko dikalkulasi berdasarkan berdasarkan energi, sudut, dan jarak dari musuh yang masih hidup.

Strategi ini mencerminkan prinsip menjaga jarak dari keramaian—seperti protokol kesehatan di dunia, seperti *social distancing*. Pada setiap langkah, ProtokolKesehatan mengevaluasi 200 posisi di sekitar lokasinya saat ini. Bot ini menghitung risiko setiap posisi, lalu memilih untuk bergerak ke posisi dengan risiko paling rendah.

Mapping komponen *greedy*:

a. Himpunan kandidat (C): 200 kandidat titik posisi di sekitar bot.
b. Himpunan solusi (S): 1 titik posisi dengan risiko terendah
c. Fungsi solusi: telah dipilih 1 titik posisi dengan risiko minimum yang bukan posisi saat ini
d. Fungsi seleksi: pilih titik posisi dengan risiko terendah dari kandidat titik yang tersedia
e. Fungsi kelayakan: kandidat titik berada di dalam arena
f. Fungsi objektif: memilih posisi dengan risiko minimum berdasarkan akumulasi data energi, sudut, dan jarak musuh yang masih hidup pada saat itu.

ProtokolKesehatan memenuhi prinsip algoritma greedy karena bot hanya memilih titik terbaik pada iterasi saat itu (berdasarkan informasi saat itu [posisi musuh, energi, dll.]), bukan secara global. Secara spesifik, ProtokolKesehatan menerapkan “Greedy by Position or Movement”.

Beberapa fungsi utama ProtokolKesehatan: DoGun() untuk strategi penembakan, DoMovement() untuk strategi pergerakan, dan EvaluatePosition() untuk kalkulasi risiko kandidat-kandidat posisi yang dapat bot tempati.

b. Analisis Efisiensi Solusi

- Fungsi DoMovement()

  Generasi dan Evaluasi Posisi: Bot mengevaluasi 200 posisi kandidat. Untuk setiap posisi, EvaluatePosition() dipanggil, yang memiliki kompleksitas **O(n)** (dengan *n* sebagai jumlah musuh) karena iterasi melalui semua musuh yang hidup. Total kompleksitas untuk mencari destinasi baru adalah **O(200 * n)** atau secara asimptotik **O(n)**.

  Frekuensi: Pencarian destinasi baru hanya terjadi saat bot mendekati destinasi saat ini, sehingga kompleksitas rata-rata per putaran lebih rendah, mendekati **O(1)**, dengan lonjakan **O(n)** sesekali.

  Pergerakan: Perhitungan sudut dan arah adalah **O(1)** karena tidak bergantung pada jumlah musuh.

- Fungsi EvaluatePosition()

  Fungsi ini mengiterasi semua musuh dalam enemies, sehingga kompleksitasnya adalah **O(n)**. Operasi seperti perhitungan sudut dan jarak adalah O(1).

- Fungsi DoGun()

  Semua operasi adalah **O(1)** karena tidak ada iterasi yang bergantung pada jumlah musuh.

Jadi, strategi ProtokolKeshatan untuk bertahan hidup dengan kompleksitas yang efisien (**O(n)** saat mencari destinasi, O(1) rata-rata).

c. Analisis Efektivitas Solusi

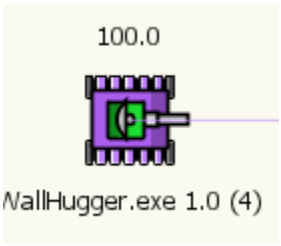
Strategi ProtokolKesehatan efektif:

- Menghindari tembakan musuh dengan menjaga jarak dan dapat bergerak osilasi, bot mengurangi risiko terkena tembakan, terutama jika musuh menggunakan strategi tembakan sederhana seperti HeadOnTargeting.
- Bertahan lebih lama. Strategi ini memungkinkan bot untuk tetap hidup lebih lama dalam pertempuran yang dapat memaksimalkan poin *survival*.
- Mengenai musuh pada jarak dekat. Tembakan HeadOnTargeting lebih mungkin mengenai musuh, terutama jika musuh tidak bergerak terlalu cepat.

Strategi kurang efektif:

- Jika musuh memiliki kecepatan lebih tinggi atau menarget ProtokolKesehatan, bot mungkin kesulitan menjaga jarak dan bisa terpojok dalam posisi yang tidak menguntungkan.
- Melawan musuh yang bergerak dengan *linear movement* atau *circular movement* karena HeadOnTargeting seringkali tidak mengenai target. Contohnya adalah sampel bot *Walls* dan juga *Crazy*.

3.3.2. WallHugger.exe



a. Deskripsi Umum

WallHugger.exe adalah bot permainan Robocode Tank Royale yang menerapkan strategi berupa gabungan dari dua strategi Greedy yang berbeda. Strategi Greedy yang pertama adalah nearest wall. Jadi pada awal ronde, bot akan mencari tembok yang paling dekat dengan posisinya dan bergerak ke tembok tersebut. Strategi ini bertujuan untuk menghindari daerah tengah yang cenderung ramai dan dengan bergerak ke salah satu tembok, bot tidak perlu memperhatikan musuh di belakang

(karena belakangnya tembok). Strategi Greedy kedua adalah menghemat energi. Bot menghemat energi dengan cara menghitung jarak musuh yang dipindai dengan posisi saat ini. Jika jarak dengan musuh cukup jauh bot tidak akan menembak. Jika jarak dengan musuh sedang, bot akan menembak peluru kecil. Jika jarak dengan musuh dekat, bot akan menembak peluru yang lebih besar.

Mapping komponen Greedy:

a. Himpunan kandidat (C): 4 tembok batas pertandingan, semua bot musuh
b. Himpunan solusi (S): 1 tembok terdekat dengan bot, bot musuh dengan jarak sedang atau dekat
c. Fungsi solusi: Terpilih satu tembok terdekat dengan bot, terpilih bot musuh yang berjarak sedang atau dekat untuk ditembak
d. Fungsi seleksi: Memilih 1 tembok (terdekat) dari himpunan kandidat, memilih bot musuh yang berjarak sedang atau dekat
e. Fungsi kelayakan: Memeriksa apakah tembok yang dipilih memiliki jarak paling kecil dengan bot, memeriksa apakah bot musuh yang dipindai berjarak jauh, sedang, atau dekat dengan posisi bot saat ini.
f. Fungsi objektif: Memilih tembok terdekat sebagai area bermain untuk menghindari area tengah, menghemat energi yang digunakan untuk menembakkan peluru.

b. Analisis Efisiensi Strategi

Kompleksitas waktu untuk bot WallHugger.exe adalah:

- Fungsi Run() memiliki kompleksitas waktu mendekati **O(1)** karena tidak ada pengulangan / iterasi yang memproses data apapun.
- Fungsi OnScannedBot() memiliki kompleksitas waktu **O(1)** karena pada method ini tidak ada perulangan. Method ini hanya memeriksa suatu conditional
- Fungsi OnHitBot() memiliki kompleksitas waktu **O(1)** karena tidak melakukan perulangan dan hanya memanggil satu method API.

- Fungsi OnHitWall() memiliki kompleksitas waktu **O(1)** karena tidak melakukan perulangan dan hanya memanggil satu method API.

c. Analisis Efektivitas Strategi

WallHugger efektif dalam kondisi:

- Ketika jumlah bot banyak dan terpusat di tengah area pertandingan. Ini berarti bot-bot di tengah lebih cenderung untuk melawan satu sama lain sehingga WallHugger dapat pergi ke tembok dengan aman.
- Ketika ukuran arena besar. Ini menguntungkan WallHugger karena meminimalkan area yang harus diperhatikan (menjaga arah belakang). Dengan kondisi ini WallHugger juga cenderung dapat bertahan hidup lebih lama dan lebih konsisten

WallHugger tidak efektif dalam kondisi:

- Jika posisi awal berdekatan dengan musuh dengan strategi yang agresif. WallHugger mementingkan untuk pergi ke tembok terdekat dan tidak memiliki strategi menyerang yang agresif di awal. Jadi musuh dengan strategi ramming, ataupun mendekat itu dapat dengan mudah mengalahkan WallHugger. Contohnya adalah sampel bot *RamFire.*
- Melawan musuh dengan strategi tembak yang dapat memprediksi posisi WallHugger selanjutnya. Karena bot ini memiliki pergerakan yang tidak terlalu rumit, maka akan sangat mudah bagi musuh yang dapat memprediksi gerakan untuk mengalahkan WallHugger.

3.3.3. SBS

a. Deskripsi Umum

SBS adalah bot yang diprogram untuk permainan Robocode Tank Royale dengan menggunakan strategi algoritma *greedy* yang fokus pada penyerangan efisien dan penyerangan balik ketika terserang bot lain. Pada awal ronde permainan, bot akan menghindari wilayah tengah dan segera menuju tembok/dinding batas permainan yang ditetapkan. Terdapat strategi untuk tiga kondisi utama, yaitu yang pertama adalah ketika melakukan penyerangan umum setelah memindai musuh. Strategi ini berfungsi untuk menghemat energi peluru berbasis jarak dari bot SBS ke bot musuh yang dipindai. Kondisi kedua adalah ketika terserang peluru di mana bot SBS akan menyerang balik musuh sesuai arah peluru tersebut. Kondisi ketiga adalah ketika menghantam bot musuh di mana bot SBS akan langsung menyerang maksimal pada arah bot musuh lalu menjauh.

Mapping komponen Greedy:

a. Himpunan kandidat (C): 4 dinding batas permainan dan semua musuh dalam permainan.
b. Himpunan solusi (S): Dinding batas permainan yang terpilih sebagai arah gerak awal dan musuh yang terdeteksi berdasarkan jarak (dekat, sedang, jauh).
c. Fungsi solusi: Memilih dinding untuk dijadikan arah gerak awal, menetapkan satu musuh yang terdeteksi.
d. Fungsi seleksi: Memilih satu dinding untuk dijadikan arah gerak awal, memilih musuh berdasarkan jarak (dekat, sedang, jauh). Memeriksa arah

peluru yang menyerang. Memeriksa apakah musuh ketika menabrak berada di belakang atau depan.

e. Fungsi kelayakan: Musuh masih hidup. Bot terkena *damage* dari peluru. Bot menghantam bot lain.

f. Fungsi objektif: Menembak musuh berdasarkan jarak untuk menghemat energi. Energi peluru dimaksimalkan untuk musuh yang berada dalam jangkauan yang dekat.

Bot SBS menggunakan strategi algoritma *greedy* dalam pengambilan keputusan karena setiap tindakan yang diambil selalu berfokus pada pilihan yang ada pada saat itu untuk memaksimalkan efektivitas serangan, efisiensi energi, dan bertahan. Setiap kali musuh terdeteksi pada tiga kondisi utama, bot langsung menyerang. Strategi yang diimplementasikan oleh bot SBS adalah strategi algoritma "Greedy by Position." Akan tetapi, strategi ini belum tentu menjadi strategi yang paling optimal untuk melawan musuh.

b. Analisis Efisiensi Solusi

Kompleksitas waktu dari operasi-operasi pemrograman yang dilakukan adalah O(n) karena tidak adanya pengulangan atau perhitungan kompleks sehingga kompleksitasnya adalah linear. Setiap keputusan bot dilakukan berdasarkan kondisi saat itu juga sehingga menjadi lebih cepat untuk bertindak.

c. Analisis Efektivitas Solusi

Strategi SBS efektif dalam:

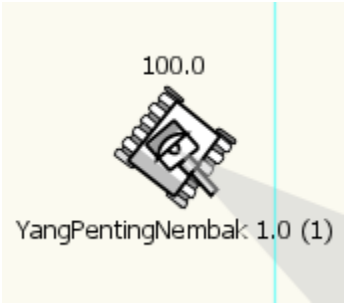
- Menemukan posisi awal strategis. Navigasi bot yang langsung menuju dinding mengurangi kemungkinan diserang bot lain dari banyak arah.

- Menyesuaikan kekuatan energi peluru berdasarkan jarak. Strategi ini efektif dalam memilah untuk penghematan konsumsi energi sehingga tidak boros akibat musuh yang terlalu jauh dan terus bergerak.
- Menyerang balik ketika ditembak. Hal ini dapat menjadi efektif untuk melawan musuh yang tidak dipindai.
- Menyerang langsung saat bertabrakan. Hal ini bisa efektif untuk segera memberikan serangan balasan.

Strategi SBS tidak efektif dalam:

- Memprediksi gerakan musuh. Program yang ada belum menghitung pola gerakan lawan dalam kecepatan dan arah gerak musuh sehingga memungkinkan adanya serangan yang tidak mengenai lawan atau adanya serangan yang boros. Bot tidak cocok untuk menyerang lawan yang terlalu lincah.
- Memilih posisi. Bot hanya bergerak di awal tanpa ada strategi navigasi lanjutan sehingga dapat memungkinkan bot terserang ketika strategi awal sudah selesai diimplementasi.

3.3.4. YangPentingNembak



a. Deskripsi Umum

YangPentingNembak adalah bot yang dirancang untuk Robocode Tank Royale dengan strategi greedy algorithm yang berfokus pada penyerangan agresif dan pemindaian musuh secara maksimal. Bot ini mengutamakan tindakan instan untuk

memaksimalkan damage per detik (memaksimalkan poin), seperti menembak dengan firepower maksimal (3.0) dan memutar radar 360° terus-menerus untuk mendeteksi musuh. Strategi ini mencerminkan prinsip "serang dulu, pertahankan diri kemudian" dengan mengabaikan perencanaan jangka panjang seperti manajemen energi atau posisi aman.

Mapping Komponen Greedy:

a. Himpunan Kandidat (C) : Semua musuh di dalam permainan.
b. Himpunan Solusi (S) : Musuh yang terdeteksi.
c. Fungsi Solusi: Solusi tercapai ketika bot telah menembak musuh yang terdeteksi dengan *firepower* 3.0 pada putaran saat ini.
d. Fungsi Seleksi : Musuh yang terdeteksi OnScannedBot().
e. Fungsi Kelayakan : Musuh masih hidup dan masih dapat terdeteksi oleh radar.
f. Fungsi Objektif : Memaksimalkan jumlah musuh yang terdeteksi oleh radar dan juga memaksimalkan damage per tembakan dengan firepower 3.0 untuk memaksimalkan *energy gain* dan poin permainan.

Prinsip Greedy yang diterapkan adalah setiap kali musuh terdeteksi, bot langsung menembak dan bergerak maju. Bot menghabiskan energi sebanyak mungkin untuk peluru berat, mengabaikan kebutuhan energi di masa depan demi memaksimalkan poin sejak awal permainan.

b. Analisis Efisiensi Solusi

Kompleksitas dari semua operasi di bot ini adalah O(1) karena tidak ada iterasi atau perhitungan kompleks. Efisiensi tinggi, tetapi tidak ada optimisasi untuk akurasi atau survival.

c. Analisis Efektivitas Solusi

Strategi YangPentingNembak efektif dalam:

- Firepower 3.0 memastikan musuh kehilangan energi cepat jika terkena tembakan. Dalam skenario awal pertempuran atau melawan bot dengan sistem penargetan sederhana (misalnya, Head-On Targeting), strategi ini bisa sangat mematikan.
- Mempertahankan jarak dekat dengan targetnya dengan selalu maju 100 unit setelah mendeteksi musuh meningkatkan kemungkinan mengenai musuh dengan tembakannya yang kuat.

Strategi kurang efektif dalam:

- Pertarungan jangka panjang (energi cepat habis). Firepower 3.0 menguras energi cepat (3 poin per tembakan), berisiko mati di fase akhir pertarungan.

3.4. Solusi Greedy Terpilih

Setelah mengevaluasi 4 opsi strategi greedy yang telah dirancang, kelompok kami memutuskan untuk menerapkan algoritma *Greedy by Position or Movement* sebagai strategi utama bot *ProtokolKesehatan*. Algoritma ini mampu memberikan hasil optimal dalam beberapa situasi tertentu, khususnya ketika fokus utama adalah bertahan hidup sambil tetap memberikan damage kepada musuh dalam pertempuran multi-bot yang kompleks. Dalam simulasi, algoritma ini terbukti unggul dalam skenario di mana bot harus menghindari risiko tinggi, seperti posisi yang dikelilingi musuh atau terjebak di sudut arena, dan sering kali menghasilkan solusi yang mendekati optimal pada kasus lainnya, misalnya ketika jumlah musuh mulai menurun namun ancaman masih ada. Salah satu keunggulan algoritma ini adalah kemampuannya untuk melakukan perhitungan dengan cepat, terutama ketika jumlah target musuh tidak terlalu banyak. Dengan kompleksitas waktu O(n) per evaluasi posisi—di mana n adalah jumlah musuh—dan evaluasi dilakukan terhadap 200 posisi tetap, total kompleksitasnya adalah O(200*n), yang tetap efisien dalam lingkungan dengan jumlah musuh terbatas seperti di Robocode Tank Royale.

Kami memutuskan untuk menerapkan algoritma Greedy by Position or Movement sebagai strategi utama bot ProtokolKesehatan karena hasil simulasi menunjukkan performa terbaiknya dibandingkan opsi lain. Dalam pengujian, ProtokolKesehatan secara konsisten mencetak poin tinggi melalui kombinasi survival point dan *bullet damage*, menunjukkan kemampuan untuk bertahan lebih lama sambil tetap efektif menyerang. Kami juga menolak strategi naive shortest distance to the wall untuk bot WallHugger.exe karena strategi ini tidak menjamin keputusan optimal dalam situasi apa pun, meskipun kerap memberikan hasil yang memadai dalam kondisi tertentu, seperti pada arena kecil dengan sedikit musuh. Strategi tersebut hanya berfokus pada jarak terdekat ke dinding tanpa mempertimbangkan faktor kritis seperti energi musuh, posisi relatif terhadap ancaman, atau potensi pergerakan musuh. Akibatnya, bot berisiko terjebak di sudut arena atau menjadi sasaran empuk bagi musuh yang lebih adaptif, sehingga meningkatkan risiko langkah yang kurang efektif dalam jangka panjang.

Selain itu, kami tidak memilih strategi agresif menembak dengan firepower maksimal untuk bot YangPentingNembak karena kompleksitas waktunya yang tinggi dalam konteks tertentu, dipengaruhi oleh luasnya arena, serta sifatnya yang kurang adaptif, yang dapat menghasilkan solusi suboptimal tanpa memerhatikan jarak bot YangPentingNembak ke kelompok musuh. Meskipun strategi ini memiliki kompleksitas O(1) untuk setiap keputusan menembak, pendekatan agresifnya yang tidak memperhitungkan manajemen energi menyebabkan bot cepat kehabisan tenaga, terutama pada arena besar tempat di mana musuh tersebar luas. Ketidakmampuan untuk beradaptasi dengan situasi—seperti menyesuaikan intensitas serangan berdasarkan jarak atau jumlah musuh—membuatnya rentan kalah dalam pertempuran jangka panjang, terutama ketika menghadapi kelompok musuh yang terorganisir atau bergerak secara dinamis.

Kami juga tidak memilih algoritma reaktif berbasis jarak sebagai strategi utama bot SBS karena algoritma ini memiliki kompleksitas waktu yang lebih tinggi dalam situasi tertentu, terutama dipengaruhi oleh ukuran arena pertempuran. Selain itu, sifatnya yang kurang fleksibel—hanya menyerang saat musuh mendekat atau saat diserang—membuatnya rentan menghasilkan solusi yang tidak optimal. Algoritma ini juga tidak memperhitungkan jarak antara bot SBS dan kelompok musuh terpadat, yang dapat menjadi kelemahan signifikan. Misalnya, jika musuh bergerak dalam pola kompleks seperti circular movement atau berkumpul di area tertentu, SBS

tidak mampu mengantisipasi atau menyesuaikan posisinya secara proaktif sehingga sering kali terpojok atau kalah dalam pertarungan melawan bot yang lebih cerdas.

Dengan mempertimbangkan semua faktor di atas, kami menyimpulkan bahwa algoritma Greedy by Position or Movement pada bot ProtokolKesehatan adalah pilihan terbaik untuk kompetisi ini. Strategi ini tidak hanya efisien secara komputasi, tetapi juga adaptif terhadap berbagai situasi pertempuran, memungkinkan bot untuk bertahan lebih lama sambil tetap memberikan damage yang signifikan kepada musuh. Penolakan terhadap strategi alternatif seperti naive shortest distance pada WallHugger.exe, pendekatan agresif pada YangPentingNembak, dan reaksi berbasis jarak pada SBS didasarkan pada keterbatasan masing-masing dalam hal adaptabilitas, efisiensi, dan kemampuan untuk menghasilkan solusi optimal dalam skenario yang kompleks. Oleh karena itu, ProtokolKesehatan dengan strategi Greedy by Position or Movement menjadi representasi terbaik dari keseimbangan antara kelangsungan hidup dan efektivitas serangan dalam lingkungan kompetitif Robocode Tank Royale.

BAB IV

IMPLEMENTASI DAN PENGUJIAN

4.1. Implementasi 4 Alternatif Solusi

4.1.1. Alternatif 1: ProtokolKesehatan

Pseudocode ProtokolKesehatan

KELAS

```
Kelas ProtokolKesehatan : Bot
    KAMUS GLOBAL
        enemies : Hashtable
        target : EnemyData
        nextDestination, lastPosition, currentPosition : Vector2D
        myEnergy : double
        random : Random
        delta : double

Kelas EnemyData
    position: Vector2D
    energy: double
    live: boolean
```

STRUKTUR DATA

```
structure Vector2D:
    KAMUS LOKAL
    X: double
    Y: double

    procedure BuatVector2D(x: double, y: double):
        X <- x
        Y <- y

    function DistanceSquared(other: Vector2D):
        KAMUS LOKAL
        deltaX: double
        deltaY: double

        ALGORITMA
        deltaX <- this.X - other.X
        deltaY <-
        return deltaX * deltaX + deltaY * deltaY

    function DistanceSquared(other: Vector2D):
        return sqrt(DistanceSquared(other))
```

```
structure Rectangle2D:
    KAMUS LOKAL
        X: double
        Y: double
        Width: double
        Height: double

    procedure BuatRectangle2D(x: double, y: double, width: double, height: double):
        X <- x
        Y <- y
        Width <- width
        Height <- height

    function ContainsRectangle(x: double, y: double) -> boolean:
        return x >= X and x <= X + Width and y >= Y and y <= Y + Height
```

FUNGSI

```
{ Konstruktor }
 procedure ProtokolKesehatan():
     panggil konstruktor induk(BotInfo.FromFile("ProtokolKesehatan.json"))
     inisialisasi variabel:
         enemies <- kosong
         target <- null
         nextDestination <- lastPosition <- currentPosition <- posisi awal(X, Y)
         myEnergy <- Energy
         random <- Random()
         delta <- 0

 { Loop utama }
 procedure Run()
     { Set radar untuk memindai arena secara terus-menerus }
     RadarTurnRate <- MaxRadarTurnRate
     nextDestination <- lastPosition <- currentPosition <- new Vector2D(X, Y)
     target <- new EnemyData()

     { Loop utama aktivitas bot }
     while IsRunning do:
         perbarui currentPosition <- new Vector2D(X, Y)
         perbarui myEnergy <- Energy

         // Fokus mendaftarkan data musuh di awal
         if target.live AND TurnNumber > 10 then:
              DoMovement()
              DoGun()
         Go()

{ Event kematian bot untuk meng-update status bot}
 procedure OnBotDeath(BotDeathEvent e):
     ((EnemyData)enemies[e.VictimId]).live <- false

 { Event pemindaian bot }
```

```
 procedure OnScannedBot(ScannedBotEvent e):
     en <- (EnemyData)enemies[e.ScannedBotId]
     if en == null then:
         en <- new EnemyData()
         enemies[e.ScannedBotId] <- en

     en.energy <- e.Energy
     en.live <- true

     e_distance <- jarak(currentPosition, new Vector2D(e.X, e.Y))
     en.pos <- new Vector2D(e.X, e.Y)

 { Perbarui target jika salah satu kondisi benar }
     if target == null OR !target.live OR e_distance <jarak(currentPosition, target.pos) then:
         target <- en
```

```
procedure DoMovement():
    KAMUS LOKAL
        distanceToNextDestination : double
        aliveEnemies : integer
        oneLeftWeight : double
        safeArea : Rectangle2D
        testPoint : Vector2D

    ALGORITMA
        distanceToDestination <- jarak(currentPosition, nextDestination)

        if distanceToDestination < 15 then:
            hitung aliveEnemies

            { Faktor penyesuaian untuk situasi 1v1. Bot akan cenderung lebih dekat dengan
            posisi terakhir (lastPosition) di 1v1 }
            oneLeftWeight <- 1 - (random.NextDouble() ^ aliveEnemies)

            Definisi safeArea = area(30,30, ArenaWidth-60, ArenaHeight-60)

            { Evaluasi 200 titik acak }
            for i <- 0 to 199:
                testPoint <- titikAcak di sekitar bot
                if testPoint ada di dalam safeArea AND EvaluatePosition(testPoint) <
                EvaluatePosition(nextDestination) then:
                    nextDestination <- testPoint
                endif
            endfor

            lastPosition <- currentPosition
        else:
            sudut <- arah(nextDestination) - arahBotSekarang
            arahGerak <- 1

            { Jika perlu belok tajam, gerak mundur }
            if cos(sudut) < 0 then:
```

```
                sudut <- sudut + 180
                arahGerak <- -1
            endif

            sudut <- NormalisasiSudutRelatif(sudut)
            SetForward(jarakKeTujuan * arahGerak)
            SetTurnLeft(sudut)

            { Atur kecepatan berdasarkan sudut belok }
            MaxSpeed <- 0 jika |sudut| > 57, else 8
        endif
```

```
function EvaluatePosition(p: Vector2D, oneLeftWeight: double) -> double:
    KAMUS LOKAL
        eval : double
        en : EnemyData
        angleDiff : double

    ALGORITMA
        { Basis risiko: hindari posisi dekat lastPosition }
        eval <- oneLeftWeight * 0.08 / jarak²(p, lastPosition)

        { Tambahkan risiko dari setiap musuh hidup }
        for enemy in enemies do:
            if musuh.hidup then:
                sudutSelisih <- arah(currentPosition ke p) - arah(musuh ke p)
                faktorEnergi <- min(enemy.energi/myEnergy, 2)
                eval <- eval + (faktorEnergi * (1 + |cos(sudutSelisih)|)) / jarak²(p, enemy)
            endif
        endfor

        return eval
```

```
procedure DoGun():
    KAMUS LOKAL
        distanceToTarget : double
        delta : double
    ALGORITMA
        delta <- selisihSudut(arahSenjata, arahTarget)

        { Kondisi tembak:
        - Sudut selisih < 5 derajat
        - Senjata tidak panas (GunHeat=0)
        - Energi cukup (>1) }

        if |delta| < 5 AND GunHeat == 0 AND myEnergy > 1 then:
            SetFire(min(energy/6, 1300/distanceToTarget))
        endif

    PutarSenjata(delta)  { Sesuaikan arah senjata ke target }
```

4.1.2. Alternatif 2: WallHugger.exe

Pseudocode WallHugger.exe

KELAS

Kelas WallHugger : Bot

FUNGSI

```
procedure Run():
    KAMUS LOKAL
        toLeft, toRight, toTop, toBottom : boolean
        distanceToRight, distanceToLeft, distanceToTop,distanceToBottom : double

    ALGORITMA
        toLeft <- false
        toRight <- false
        toTop <- false
        toBottom <- false

        distanceToRight <- (ArenaWidht - this.X)
        distanceToLeft <- this.X
        distanceToTop <- (ArenaHeight - this.Y)
        distanceToBottom <- this.Y


        if distanceToBottom <= distanceToRight AND distanceToBottom <= distanceToTop AND
distanceToBottom <= distanceToLeft then:
            toBottom <- true
        endif

        if distanceToTop <= distanceToBottom AND distanceToTop <= distanceToLeft AND
distanceToTop <= distanceToRight then:
            toTop <- true
        endif

        if distanceToRight <= distanceToLeft AND distanceToRight <= distanceToTop AND
distanceToRight <= distanceToBottom then:
            toRight <- true
        endif

        if distanceToLeft <= distanceToRight AND distanceToLeft <= distanceToTop AND
distanceToLeft <= distanceToBottom then:
            toLeft <- true
        endif


        if toBottom = true AND IsRunning = true then:
            TurnLeft(CalcBearing(-90))
            while this.Y > 50 AND IsRunning = true do:
                Forward(15)
                TurnGunLeft(CalcBearing(90))
                TurnGunLeft(CalcBearing(-90))
            endwhile
```

```
        while IsRunning = true do:

            while this.X < (ArenaWidth - 50) AND IsRunning = true do:
                TurnLeft(CalcBearing(0))
                TurnGunLeft(CalcGunBearing(90))
                TurnGunLeft(CalcGunBearing(180))
                TurnGunLeft(CalcGunBearing(90))
                TurnGunLeft(CalcGunBearing(0))
                Forward(25)
            endwhile

            while this.X > 50 AND IsRunning = true do:
                TurnLeft(CalcBearing(180))
                TurnGunLeft(CalcGunBearing(90))
                TurnGunLeft(CalcGunBearing(0))
                TurnGunLeft(CalcGunBearing(90))
                TurnGunLeft(CalcGunBearing(180))
                Forward(25)
            endwhile


        endwhile
    endif

    if toTop = true AND IsRunning = true then:
        TurnLeft(CalcBearing(90))
        while this.Y < (ArenaHeight - 50) AND IsRunning = true do:
            Forward(15)
            TurnGunLeft(CalcBearing(-90))
            TurnGunLeft(CalcBearing(90))
        endwhile

        while IsRunning = true do:

            while this.X < (ArenaWidth - 50) AND IsRunning = true do:
                TurnLeft(CalcBearing(0))
                TurnGunLeft(CalcGunBearing(-90))
                TurnGunLeft(CalcGunBearing(-180))
                TurnGunLeft(CalcGunBearing(-90))
                TurnGunLeft(CalcGunBearing(0))
                Forward(25)
            endwhile

            while this.X > 50 AND IsRunning = true do:
                TurnLeft(CalcBearing(180))
                TurnGunLeft(CalcGunBearing(-90))
                TurnGunLeft(CalcGunBearing(0))
                TurnGunLeft(CalcGunBearing(-90))
                TurnGunLeft(CalcGunBearing(-180))
                Forward(25)
            endwhile
        endwhile
    endif

    if toLeft = true AND IsRunning = true then:
        TurnLeft(CalcBearing(180))
        while this.X > 50 AND IsRunning = true do:
            Forward(15)
            TurnGunLeft(CalcBearing(0))
```

```
            TurnGunLeft(CalcBearing(180))
        endwhile

        while IsRunning = true do:

            while this.Y < (ArenaHeight - 50) AND IsRunning = true do:
                TurnLeft(CalcBearing(90))
                TurnGunLeft(CalcGunBearing(0))
                TurnGunLeft(CalcGunBearing(-90))
                TurnGunLeft(CalcGunBearing(0))
                TurnGunLeft(CalcGunBearing(90))
                Forward(25)
            endwhile

            while this.Y > 50 AND IsRunning = true do:
                TurnLeft(CalcBearing(-90))
                TurnGunLeft(CalcGunBearing(0))
                TurnGunLeft(CalcGunBearing(90))
                TurnGunLeft(CalcGunBearing(0))
                TurnGunLeft(CalcGunBearing(-90))
                Forward(25)
            endwhile
        endwhile
    endif

    if toRight = true AND IsRunning = true then:
        TurnLeft(CalcBearing(0))
        while this.X < (ArenaWidth - 50) AND IsRunning = true do:
            Forward(15)
            TurnGunLeft(CalcBearing(180))
            TurnGunLeft(CalcBearing(0))
        endwhile

        while IsRunning = true do:

            while this.Y < (ArenaHeight - 50) AND IsRunning = true do:
                TurnLeft(CalcBearing(90))
                TurnGunLeft(CalcGunBearing(180))
                TurnGunLeft(CalcGunBearing(-90))
                TurnGunLeft(CalcGunBearing(180))
                TurnGunLeft(CalcGunBearing(90))
                Forward(25)
            endwhile

            while this.Y > 50 AND IsRunning = true do:
                TurnLeft(CalcBearing(-90))
                TurnGunLeft(CalcGunBearing(180))
                TurnGunLeft(CalcGunBearing(90))
                TurnGunLeft(CalcGunBearing(180))
                TurnGunLeft(CalcGunBearing(-90))
                Forward(25)
            endwhile
        endwhile
    endif
```

```
procedure OnScannedBot(e : ScannedBotEvent):

    KAMUS LOKAL
```

```
    ALGORITMA
        if EnemyCount = 1 then:
            Fire(1)
        else:
            if DistanceTo(e.X, e.Y) <= 600 AND DistanceTo(e.X, e.Y) >= 200 then:
                Fire(1)
            else if DistanceTo(e.X, e.Y) < 200 then:
                Fire(2)
            endif
        endif
```

```
procedure OnHitBot(e : HitBotEvent):
    KAMUS LOKAL

    ALGORITMA
        Back(15)
```

```
procedure OnHitWall(e : HitWalltEvent):
    KAMUS LOKAL

    ALGORITMA
        Back(15)
```

4.1.3. Alternatif 3: SBS

Pseudocode SBS

```
KELAS
```

```
Kelas SBS : Bot
    peek : boolean
    moveAmount : double
```

```
FUNGSI
```

```
procedure Run():
    { Bot Bergerak ke Tembok Terlebih Dahulu)
    moveAmount = Math.Max(ArenaHeight, ArenaWidth);
    Peek = false
    TurnRight(Direction %90)
    peek = true
    TurnGunRight(90)
    TurnRight(90)

    while isRunning == true do:
        TurnGunRight(360)
    endwhile
```

```
procedure OnScannedBot(e: ScannedBotEvent):
    SetTurnRight(CalcBearing(e.Direction))
    Range = Math.sqrt(e.X * e.X + e.Y * e.Y)
    if Range >= 500 then:
```

```
        SetFire(1)
        SetForward(100)
        Rescan()
    elseif Range >= 400 And Range < 500 then:
        SetFire(2)
        SetForward(100)
        Rescan()
    elseif Range < 400 then:
        SetFire(3)
        SetForward(100)
        Rescan()
    endif
```

```
procedure OnHitByBullet(e : HitByBulletEvent) :
    {Hit oleh peluru, balas dengan serangan mini ke arah musuh}
    bearing = CalcBearing(e.Bullet.Direction)

    TurnLeft(90 - bearing)
    Back(100)
    TurnGunLeft(bearing)
    SetFire(1)
    Forward(100)
```

```
procedure OnHitBot(e : HitBotEvent)
    {Bot bertabrakan dengan bot lain, langsung menyerang}
    bearing = BearingTo(e.X, e.Y)

    if bearing > -90 And bearing < 90 then:
        TurnGunRight(bearing)
        SetFire(3)
        Back(100)
    else
        TurnGunRight(bearing)
        SetFire(3)
        Forward(100)
    endif
```

4.1.4. Alternatif 4: YangPentingNembak

Pseudocode YangPentingNembak

```
KELAS
```

```
Kelas YangPentingNembak : Bot
    gunDirection: integer
```

```
FUNGSI
```

```
procedure Run():
    while isRunning == true do:
        TurnGunRight(360)
```

```
procedure OnScannedBot(e: ScannedBotEvent):
    { Arahkan bot ke musuh }
    SetTurnRight(CalcBearing(e.Direction)
    { HeadOnTargeting }
    SetFire(3)
    { bergerak maju }
    SetForward(100)
    { Balikkan senjata di setiap turn }
    GunTurnRate = -gunDirection
    { Berbalik 360 derajat }
    SetTurnGunRight(360 * gunDirection)
```

4.2. Penjelasan Struktur Data, Fungsi, dan Prosedur Strategi Terpilih (ProtokolKesehatan)

Struktur Data

1. Vector2D

- Tujuan : Menyimpan koordinat (X, Y) dan operasi geometris.
- Properti :
    - X, Y: Koordinat dalam arena.
- Metode :
    - DistanceTo(Vector2D other): Hitung jarak Euclidean.
    - DistanceSquared(Vector2D other): Hitung jarak kuadrat (efisiensi).

2. Rectangle2D

- Tujuan : Membatasi area aman di arena.
- Properti :
    - X, Y, Width, Height: Batas area.
- Metode :
    - Contains(Vector2D point): Cek apakah titik berada dalam area.

3. EnemyData

- Tujuan : Menyimpan dan meng-*update* informasi musuh pada saat musuh berhasil di-*scan*.
- Properti :
    - Position: Posisi musuh.
    - Energy: Energi musuh.
    - Live: Status hidup/mati.

4. Dictionary<Guid, EnemyData>

- Tujuan : Melacak semua musuh yang terdeteksi.

Fungsi dan Prosedur Utama

1. Run()

- Alur :
    - Inisialisasi radar dan posisi awal.
    - Loop aktivitas: perbarui posisi, panggil DoMovement() dan DoGun().

2. DoMovement()

- Alur :
    - Jika dekat tujuan :
        - Evaluasi 200 titik acak.
        - Pilih titik dengan risiko terendah.
    - Jika jauh dari tujuan :
        - Sesuaikan kecepatan berdasarkan sudut belok.

3. EvaluatePosition(Vector2D p, double oneLeftWeight)

- Rumus Risiko :

  Risiko = (Penalti Osilasi) + Σ(Ancaman Musuh)

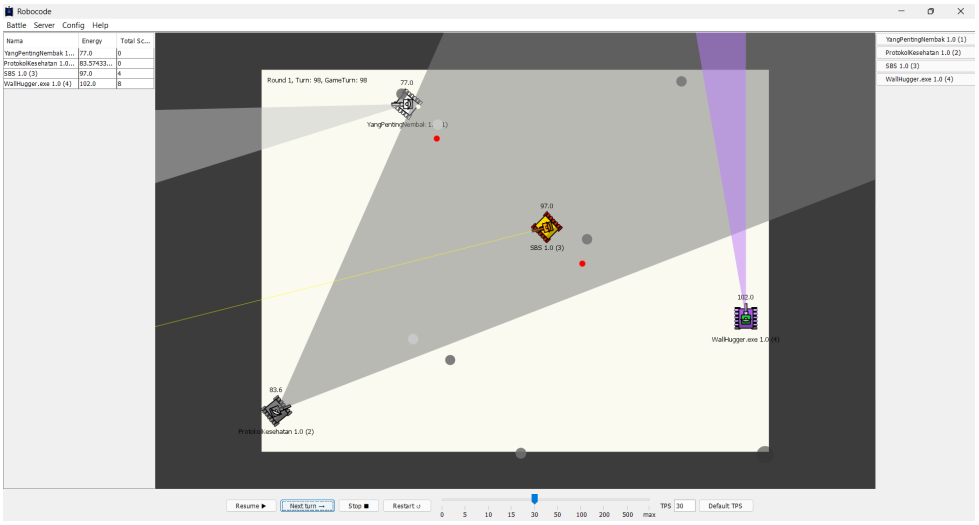
- Penalti Osilasi : Hindari kembali ke posisi sebelumnya.
- Ancaman Musuh : Faktor energi, sudut, dan jarak.

4. DoGun()

- Kondisi Tembak :
  - Sudut selisih <5°.
  - Senjata siap (GunHeat = 0).
  - Energi cukup (myEnergy > 1).
- Daya Tembak : min(myEnergy/6, 1300/distanceToTarget).

4.3. Pengujian

Setelah menerapkan algoritma strategi yang dirancang, dilakukan pengujian 1v1v1v1 dengan mengoperasikan bot dalam pertarungan melawan bot lain. Selama pertarungan, perilaku bot dipantau untuk memastikan tindakan yang diambil (seperti pergerakan, penembakan, atau respons terhadap musuh) selaras dengan strategi yang telah direncanakan. Evaluasi ini bertujuan memastikan bot beroperasi sesuai desain awal dan mengidentifikasi potensi penyimpangan atau area perbaikan.



Gambar 4.3.1. *Screenshot* Pengujian

4.3.1. Pengujian 1v1v1v1

Results for 10 rounds

Rank	Name	Total Score	Survival	Surv. Bonus	Bullet Dmg.	Bullet Bonus	Ram Dmg.	Ram Bonus	1sts	2nds	3rds
1	ProtokolKesehatan 1.0	1864	900	180	698	86	0	0	6	0	1
2	YangPentingNembak 1.0	1154	450	30	640	32	2	0	1	4	0
3	SBS 1.0	677	350	0	300	15	12	0	0	1	3
4	WallHugger.exe 1.0	674	400	0	208	4	61	0	0	2	3

Gambar 4.3.1.1. Pengujian 1

Results for 10 rounds

Rank	Name	Total Score	Survival	Surv. Bonus	Bullet Dmg.	Bullet Bonus	Ram Dmg.	Ram Bonus	1sts	2nds	3rds
1	ProtokolKesehatan 1.0	2508	1000	180	1141	186	0	0	7	0	0
2	YangPentingNembak 1.0	721	250	0	448	9	13	0	0	2	2
3	WallHugger.exe 1.0	600	400	0	198	1	0	0	0	2	4
4	SBS 1.0	594	400	0	180	6	7	0	0	3	2

Gambar 4.3.1.2. Pengujian 2

Results for 10 rounds

Rank	Name	Total Score	Survival	Surv. Bonus	Bullet Dmg.	Bullet Bonus	Ram Dmg.	Ram Bonus	1sts	2nds	3rds
1	SBS 1.0	1213	650	90	412	35	25	0	3	1	2
2	ProtokolKesehatan 1.0	1159	500	60	563	36	0	0	2	2	1
3	YangPentingNembak 1.0	1109	400	0	608	89	11	0	1	3	2
4	WallHugger.exe 1.0	602	400	30	140	0	32	0	1	1	2

Gambar 4.3.1.4. Pengujian 3

Results for 10 rounds

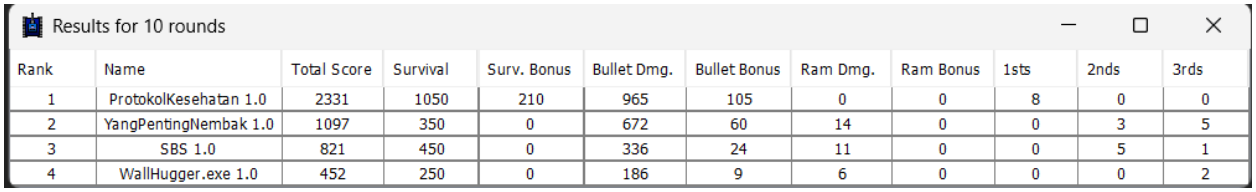
Rank	Name	Total Score	Survival	Surv. Bonus	Bullet Dmg.	Bullet Bonus	Ram Dmg.	Ram Bonus	1sts	2nds	3rds
1	ProtokolKesehatan 1.0	2346	950	150	1074	170	1	0	5	2	0
2	SBS 1.0	825	500	60	240	18	7	0	3	1	2
3	YangPentingNembak 1.0	823	350	0	432	25	16	0	0	3	3
4	WallHugger.exe 1.0	499	300	0	184	8	7	0	0	2	3

Gambar 4.3.1.5. Pengujian 4

Results for 10 rounds

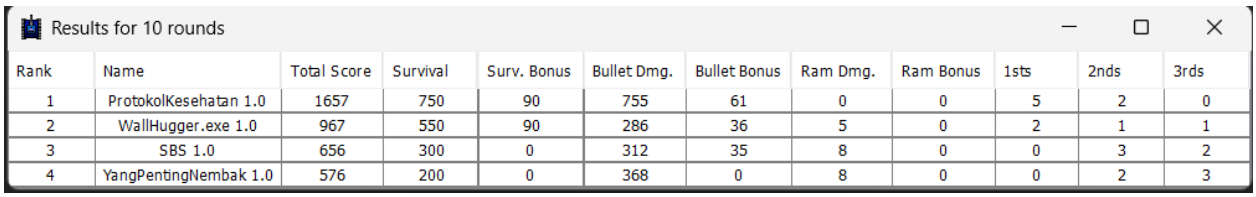
Rank	Name	Total Score	Survival	Surv. Bonus	Bullet Dmg.	Bullet Bonus	Ram Dmg.	Ram Bonus	1sts	2nds	3rds
1	ProtokolKesehatan 1.0	2403	1000	180	1058	165	0	0	7	0	0
2	SBS 1.0	796	500	30	240	11	14	0	0	3	3
3	YangPentingNembak 1.0	687	250	0	416	19	1	0	0	1	4
4	WallHugger.exe 1.0	549	350	0	190	6	2	0	0	3	0

Gambar 4.3.1.6. Pengujian 5



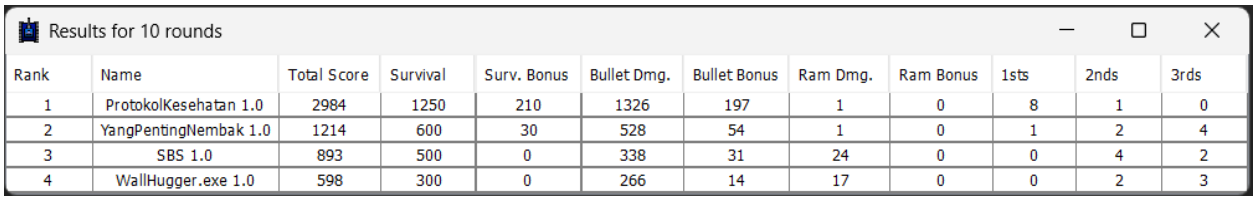
Rank	Name	Total Score	Survival	Surv. Bonus	Bullet Dmg.	Bullet Bonus	Ram Dmg.	Ram Bonus	1sts	2nds	3rds
1	ProtokolKesehatan 1.0	2331	1050	210	965	105	0	0	8	0	0
2	YangPentingNembak 1.0	1097	350	0	672	60	14	0	0	3	5
3	SBS 1.0	821	450	0	336	24	11	0	0	5	1
4	WallHugger.exe 1.0	452	250	0	186	9	6	0	0	0	2

Gambar 4.3.1.7. Pengujian 6



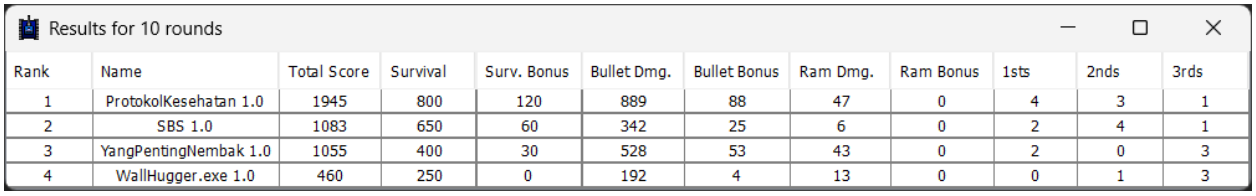
Rank	Name	Total Score	Survival	Surv. Bonus	Bullet Dmg.	Bullet Bonus	Ram Dmg.	Ram Bonus	1sts	2nds	3rds
1	ProtokolKesehatan 1.0	1657	750	90	755	61	0	0	5	2	0
2	WallHugger.exe 1.0	967	550	90	286	36	5	0	2	1	1
3	SBS 1.0	656	300	0	312	35	8	0	0	3	2
4	YangPentingNembak 1.0	576	200	0	368	0	8	0	0	2	3

Gambar 4.3.1.3. Pengujian 7



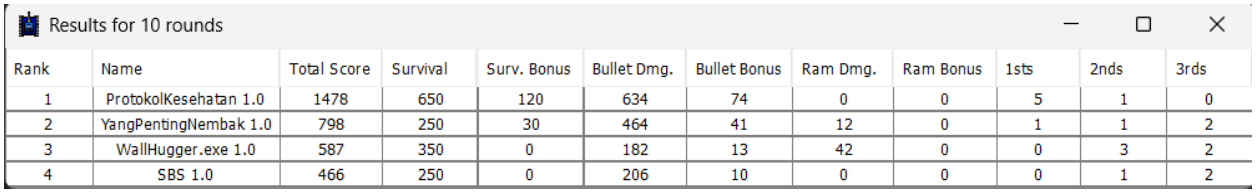
Rank	Name	Total Score	Survival	Surv. Bonus	Bullet Dmg.	Bullet Bonus	Ram Dmg.	Ram Bonus	1sts	2nds	3rds
1	ProtokolKesehatan 1.0	2984	1250	210	1326	197	1	0	8	1	0
2	YangPentingNembak 1.0	1214	600	30	528	54	1	0	1	2	4
3	SBS 1.0	893	500	0	338	31	24	0	0	4	2
4	WallHugger.exe 1.0	598	300	0	266	14	17	0	0	2	3

Gambar 4.3.1.8. Pengujian 8



Rank	Name	Total Score	Survival	Surv. Bonus	Bullet Dmg.	Bullet Bonus	Ram Dmg.	Ram Bonus	1sts	2nds	3rds
1	ProtokolKesehatan 1.0	1945	800	120	889	88	47	0	4	3	1
2	SBS 1.0	1083	650	60	342	25	6	0	2	4	1
3	YangPentingNembak 1.0	1055	400	30	528	53	43	0	2	0	3
4	WallHugger.exe 1.0	460	250	0	192	4	13	0	0	1	3

Gambar 4.3.1.9. Pengujian 9



Rank	Name	Total Score	Survival	Surv. Bonus	Bullet Dmg.	Bullet Bonus	Ram Dmg.	Ram Bonus	1sts	2nds	3rds
1	ProtokolKesehatan 1.0	1478	650	120	634	74	0	0	5	1	0
2	YangPentingNembak 1.0	798	250	30	464	41	12	0	1	1	2
3	WallHugger.exe 1.0	587	350	0	182	13	42	0	0	3	2
4	SBS 1.0	466	250	0	206	10	0	0	0	1	2

Gambar 4.3.1.10. Pengujian 10

4.3.2. Pengujian ProtokolKesehatan vs Sample Bots

Results for 10 rounds

Rank	Name	Total Score	Survival	Surv. Bonus	Bullet Dmg.	Bullet Bonus	Ram Dmg.	Ram Bonus	1sts	2nds	3rds
1	ProtokolKesehatan 1.0	3683	1900	200	1412	101	50	20	2	3	0
2	Spin Bot 1.0	3373	1950	200	1040	136	47	0	2	1	2
3	Walls 1.0	2351	1600	0	690	34	26	0	0	1	1
4	Ram Fire 1.0	1656	800	0	444	62	318	32	1	0	0
5	Crazy 1.0	1544	1300	0	216	0	28	0	0	0	1
6	Track Fire 1.0	1384	750	0	628	5	0	0	0	0	1
7	Fire 1.0	1089	900	0	184	0	5	0	0	0	0
8	Corners 1.0	1082	800	0	274	4	4	0	0	0	0
9	My First Bot 1.0	890	650	0	236	0	4	0	0	0	0
10	My First Droid 1.0	800	800	0	0	0	0	0	0	0	0
11	Target 1.0	502	500	0	0	0	2	0	0	0	0

Gambar 4.3.2.1. Pengujian 1

Results for 10 rounds

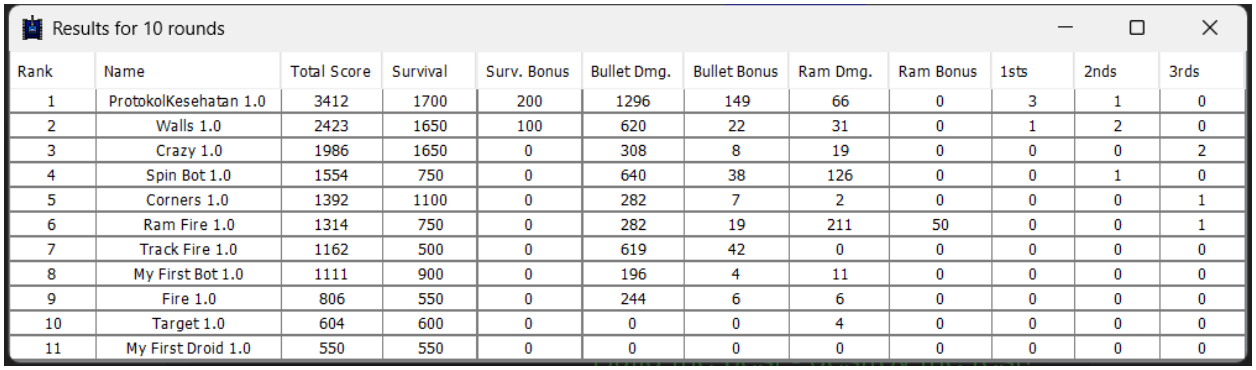
Rank	Name	Total Score	Survival	Surv. Bonus	Bullet Dmg.	Bullet Bonus	Ram Dmg.	Ram Bonus	1sts	2nds	3rds
1	ProtokolKesehatan 1.0	4085	2250	300	1374	138	23	0	4	1	0
2	Walls 1.0	2785	1950	100	690	20	25	0	0	1	3
3	Spin Bot 1.0	2642	1600	0	912	56	73	0	1	1	1
4	Track Fire 1.0	2501	1350	0	1103	48	0	0	0	2	0
5	Corners 1.0	1754	1300	0	422	26	5	0	0	0	1
6	Crazy 1.0	1650	1400	0	224	2	24	0	0	0	0
7	My First Bot 1.0	1515	1200	0	288	12	14	0	0	0	0
8	Fire 1.0	1438	1100	0	316	18	4	0	0	0	0
9	Ram Fire 1.0	871	300	0	320	0	218	32	0	0	0
10	Target 1.0	501	500	0	0	0	1	0	0	0	0
11	My First Droid 1.0	300	300	0	0	0	0	0	0	0	0

Gambar 4.3.2.2. Pengujian 2

Results for 10 rounds

Rank	Name	Total Score	Survival	Surv. Bonus	Bullet Dmg.	Bullet Bonus	Ram Dmg.	Ram Bonus	1sts	2nds	3rds
1	ProtokolKesehatan 1.0	3735	1950	300	1292	188	5	0	4	0	0
2	Spin Bot 1.0	2611	1600	0	880	44	86	0	0	3	0
3	Track Fire 1.0	2038	1100	0	884	53	0	0	0	1	1
4	Walls 1.0	1984	1350	0	550	31	53	0	0	0	2
5	My First Bot 1.0	1289	1050	0	224	6	8	0	0	0	1
6	Corners 1.0	1280	950	0	314	12	4	0	0	0	0
7	Crazy 1.0	1214	950	0	228	1	35	0	0	0	0
8	Target 1.0	807	800	0	0	0	7	0	0	0	0
9	Ram Fire 1.0	786	500	0	164	0	115	7	0	0	0
10	Fire 1.0	488	350	0	136	0	2	0	0	0	0
11	My First Droid 1.0	350	350	0	0	0	0	0	0	0	0

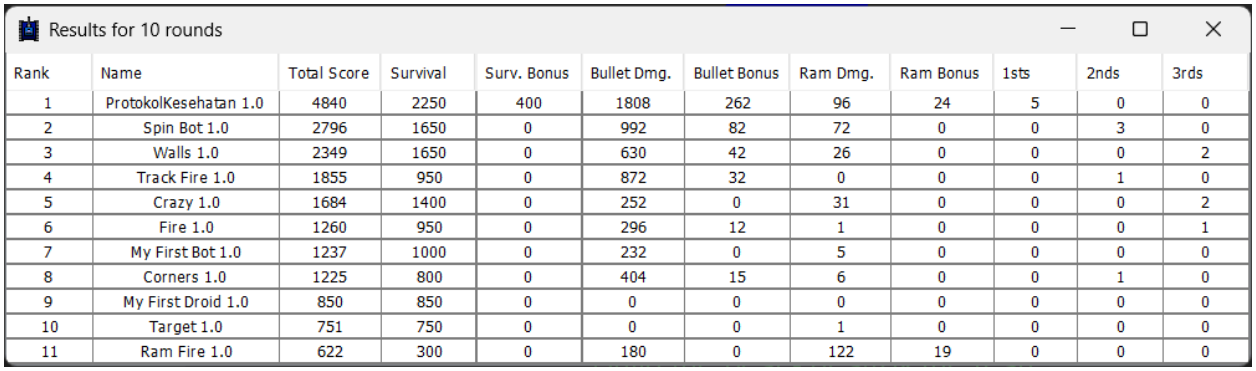
Gambar 4.3.2.3. Pengujian 3



Results for 10 rounds

Rank	Name	Total Score	Survival	Surv. Bonus	Bullet Dmg.	Bullet Bonus	Ram Dmg.	Ram Bonus	1sts	2nds	3rds
1	ProtokolKesehatan 1.0	3412	1700	200	1296	149	66	0	3	1	0
2	Walls 1.0	2423	1650	100	620	22	31	0	1	2	0
3	Crazy 1.0	1986	1650	0	308	8	19	0	0	0	2
4	Spin Bot 1.0	1554	750	0	640	38	126	0	0	1	0
5	Corners 1.0	1392	1100	0	282	7	2	0	0	0	1
6	Ram Fire 1.0	1314	750	0	282	19	211	50	0	0	1
7	Track Fire 1.0	1162	500	0	619	42	0	0	0	0	0
8	My First Bot 1.0	1111	900	0	196	4	11	0	0	0	0
9	Fire 1.0	806	550	0	244	6	6	0	0	0	0
10	Target 1.0	604	600	0	0	0	4	0	0	0	0
11	My First Droid 1.0	550	550	0	0	0	0	0	0	0	0

Gambar 4.3.2.4. Pengujian 4



Results for 10 rounds

Rank	Name	Total Score	Survival	Surv. Bonus	Bullet Dmg.	Bullet Bonus	Ram Dmg.	Ram Bonus	1sts	2nds	3rds
1	ProtokolKesehatan 1.0	4840	2250	400	1808	262	96	24	5	0	0
2	Spin Bot 1.0	2796	1650	0	992	82	72	0	0	3	0
3	Walls 1.0	2349	1650	0	630	42	26	0	0	0	2
4	Track Fire 1.0	1855	950	0	872	32	0	0	0	1	0
5	Crazy 1.0	1684	1400	0	252	0	31	0	0	0	2
6	Fire 1.0	1260	950	0	296	12	1	0	0	0	1
7	My First Bot 1.0	1237	1000	0	232	0	5	0	0	0	0
8	Corners 1.0	1225	800	0	404	15	6	0	0	1	0
9	My First Droid 1.0	850	850	0	0	0	0	0	0	0	0
10	Target 1.0	751	750	0	0	0	1	0	0	0	0
11	Ram Fire 1.0	622	300	0	180	0	122	19	0	0	0

Gambar 4.3.2.5. Pengujian 5

4.4. Analisis Hasil Pengujian

Berdasarkan hasil pengujian, bot ProtokolKesehatan secara konsisten menempati peringkat pertama dengan skor total tertinggi di setiap pengujian. Keunggulan ProtokolKesehatan terlihat dari performa yang seimbang antara survival point dan bullet damage, yang mencerminkan efektivitas strategi Greedy by Position or Movement yang diterapkan.

ProtokolKesehatan unggul karena strategi utamanya yang berfokus pada penjagaan jarak dari musuh, sehingga memaksimalkan survival point dan survival bonus. Selain itu, bot ini mampu memberikan bullet damage yang signifikan dengan bullet bonus yang konsisten, menunjukkan kemampuan ofensif yang efektif meskipun prioritas utamanya adalah bertahan hidup. Minimnya ram damage menegaskan bahwa bot ini berhasil menghindari kontak fisik dengan musuh, sesuai dengan pendekatan "social distancing" yang diterapkan.

Dibandingkan dengan bot lain, ProtokolKesehatan mengungguli WallHugger.exe, SBS, dan YangPentingNembak dalam hal skor total dan konsistensi peringkat. WallHugger.exe, meskipun memiliki survival point yang baik, kurang efektif dalam memberikan bullet damage , sehingga skornya lebih rendah. SBS juga menunjukkan performa yang tidak seimbang, dengan bullet damage yang cukup baik tetapi survival point yang rendah, menunjukkan strategi yang kurang adaptif. YangPentingNembak, dengan pendekatan agresifnya, menghasilkan bullet damage yang tinggi tetapi gagal bertahan lama, sehingga skor totalnya jauh di bawah ProtokolKesehatan.

Berdasarkan hasil pengujian selama 5 ronde dengan *sample bots*, bot ProtokolKesehatan menempati peringkat pertama dengan total skor 3683, mengungguli sample bots seperti Spin Bot Walls, Ram Fire, Crazy, Track Fire, Fire, Corners, My First Bot, My First Droid, dan Target (502). Keunggulan ProtokolKesehatan terletak pada strategi Greedy by Position or Movement, yang memungkinkan bot untuk bertahan lama sambil memberikan bullet damage tinggi). Ram damage dan ram bonus yang kecil menunjukkan efektivitas bot dalam menjaga jarak dari musuh, sesuai dengan pendekatan *"social distancing"*.

Namun, ProtokolKesehatan tidak jarang ingin dilangkahi oleh Walls (skor 2351) dan Spin Bot (skor 3373), yang menunjukkan skor kompetitif. Spin Bot, misalnya, memiliki bullet damage 1040 dan bullet bonus 136, serta survival point 1950, yang sangat mendekati ProtokolKesehatan. Hal ini disebabkan oleh kelemahan utama ProtokolKesehatan, yaitu penggunaan strategi HeadOnTargeting untuk penyerangan, yang kurang efektif melawan bot dengan pergerakan linear (seperti Walls) atau circular (seperti Spin Bot). HeadOnTargeting hanya menembak lurus ke arah musuh tanpa memprediksi pergerakan sehingga tembakan sering meleset terhadap bot yang bergerak dinamis, memungkinkan Walls dan Spin Bot untuk bertahan lebih lama atau memberikan damage balik yang lebih konsisten.

Meskipun demikian, ProtokolKesehatan tetap unggul secara keseluruhan karena kemampuan strateginya dalam mengevaluasi 200 posisi di sekitar bot dan memilih posisi dengan risiko terendah berdasarkan energi, sudut, dan jarak musuh. Ini memungkinkan bot untuk bertahan lebih lama dan tetap memberikan damage yang signifikan, terbukti dari posisi 1st sebanyak 2 kali dan 2nd sebanyak 3 kali. Dibandingkan sample bots lainnya seperti Ram Fire (1656) atau Crazy (1544), ProtokolKesehatan jauh lebih kompetitif, meskipun perlu peningkatan pada

strategi penargetan untuk mengatasi bot dengan pergerakan kompleks seperti Walls dan Spin Bot.

BAB V

KESIMPULAN DAN SARAN

5.1 Kesimpulan

Dari keseluruhan tugas besar ini, penulis dapat menyimpulkan bahwa algoritma Greedy dapat digunakan dalam pembuatan bot permainan Robocode Tank Royale. Algoritma Greedy dapat diimplementasikan dengan berbagai cara dan masing-masing implementasi memiliki keunggulan dan kekurangan. Meskipun demikian algoritma Greedy tidak bisa memberikan hasil paling baik untuk suatu persoalan, termasuk pembuatan bot Robocode Tank Royale, karena algoritma ini hanyalah aproksimasi nilai terbaik untuk persoalan tersebut. Namun walau tidak menjamin hasil terbaik, algoritma Greedy dapat secara efektif memberikan hasil yang memuaskan untuk pembuatan bot Robocode Tank Royale.

Dari hasil pengujian, penulis dapat menyimpulkan bahwa bot paling baik untuk kondisi pertarungan 1v1v1v1 adalah bot ProtokolKesehatan karena strategi Greedy by Movement yang dapat menentukan posisi aman membuat bot ini sangat fleksibel dan adaptif terhadap berbagai situasi, termasuk *bullet bonus* dan *survival bonus*.

5.2 Saran

Robocode Tank Royale adalah permainan yang terus berkembang. Untuk membuat suatu bot permainan ini, diperlukan strategi dan pertimbangan mengenai seluruh aspek pertandingan dan bot. Mempelajari API dengan menyeluruh dapat membantu pembuat bot memahami semua aspek dari Robocode Tank Royale. Kemudian, Greedy bukanlah satu-satunya algoritma yang bisa dipakai untuk membuat bot permainan ini. Oleh karena itu jika ingin membuat bot, penulis menyarankan untuk mempertimbangkan algoritma lain yang lebih efektif dan mangkus dibandingkan dengan Greedy. Mungkin algoritma Greedy dapat digabungkan dengan algoritma lain sehingga menghasilkan logika bot yang lebih baik. Namun jika ingin membuat bot dengan batasan hanya menggunakan algoritma Greedy, penulis menyarankan untuk mempertimbangkan sebanyak-banyaknya kemungkinan yang dapat terjadi ketika pertandingan berlangsung serta mengimplementasikan solusi Greedy untuk kemungkinan tersebut hingga pada kasus ekstrim (edge case).

DAFTAR PUSTAKA

Institut Teknologi Bandung. (2024). *Spesifikasi Tugas Besar 1 Stima 2024/2025* [Google Document]. Diambil dari https://docs.google.com/document/d/14MCaRiFGiA6Ez5W8-OLxZ9enXyENcep7AzSH6sUHKM8/edit?tab=t.0

Munir, R. (2025). *04-Algoritma-Greedy-(2025)-Bag1* [PDF]. Diambil dari https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2024-2025/04-Algoritma-Greedy-(2025)-Bag1.pdf

Munir, R. (2025). *05-Algoritma-Greedy-(2025)-Bag2* [PDF]. Diambil dari https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2024-2025/05-Algoritma-Greedy-(2025)-Bag2.pdf

Munir, R. (2025). *06-Algoritma-Greedy-(2025)-Bag3* [PDF]. Diambil dari https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2024-2025/06-Algoritma-Greedy-(2025)-Bag3.pdf

RoboWiki Community. (n.d.). *Main Page* . Diambil dari https://robowiki.net/wiki/Main_Page

Robocode Development Team. (n.d.). *Robocode Tank Royale* . Diambil dari https://robocode-dev.github.io/tank-royale/

LAMPIRAN

1. **Link Repository GitHub**

   https://github.com/KalengBalsem/Tubes1_lima_empat

2. **Link Video**

   https://youtu.be/OQQW_99x8lA?si=jsd_dhXBcKkKuGUm

3. **Tabel Ketercapaian**

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.	√	