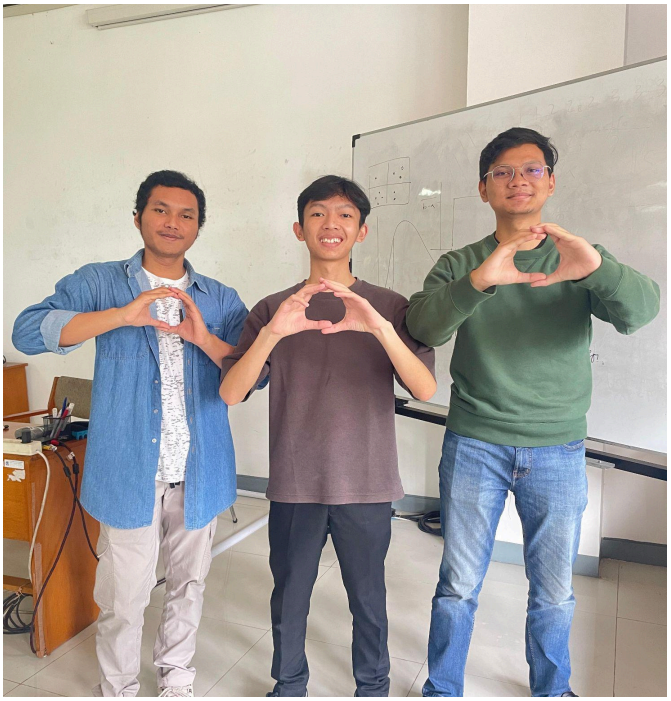
LAPORAN TUGAS BESAR 1

IF2211-STRATEGI ALGORITMA



Disusun oleh :

Kelompok 60 - Omni-Tank

Muhammad Luqman Hakim	13523044
Muhammad Edo Raduputu Aprima	13523096
Ferdin Arsenarendra Purtadi	13523117

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.

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

Komponen-komponen dari permainan ini antara lain:

1. Rounds dan Turns

   Pertempuran dapat terdiri dari beberapa rounds. Secara default, satu pertempuran berisi 10 rounds, di mana setiap rounds akan memiliki pemenang dan yang kalah.

   Setiap round dibagi menjadi beberapa turns, yang merupakan unit waktu terkecil. Satu turn adalah satu ketukan waktu dan satu putaran permainan. Jumlah turn dalam satu round tergantung pada berapa lama waktu yang dibutuhkan hingga hanya tersisa bot terakhir yang bertahan.

   Pada setiap turn, sebuah bot dapat:

   - Menggerakkan bot, memindai musuh, dan menembakkan senjata.
   - Bereaksi terhadap peristiwa seperti saat bot terkena peluru atau bertabrakan dengan bot lain atau dinding.
   - Perintah untuk bergerak, berputar, memindai, menembak, dan sebagainya dikirim ke server untuk setiap turn.

   Perlu diperhatikan bahwa API (Application Programming Interface) bot resmi secara otomatis mengirimkan niat bot ke server di balik layar, sehingga Anda tidak perlu mengkhawatirkannya, kecuali jika Anda membuat API Bot sendiri.

Pada setiap turn, bot akan secara otomatis menerima informasi terbaru tentang posisinya dan orientasinya di medan perang. Bot juga akan mendapatkan informasi tentang bot musuh ketika mereka terdeteksi oleh pemindai.

Perlu diketahui bahwa game engine yang akan digunakan pada tugas besar ini tidak mengikuti aturan default mengenai komponen Round & Turns.

2. Batas Waktu Giliran

   Penting untuk dicatat bahwa setiap bot memiliki batas waktu untuk setiap turn yang disebut turn timeout, biasanya antara 30-50 ms (dapat diatur sebagai aturan pertempuran). Ini berarti bahwa bot tidak bisa mengambil waktu sebanyak yang mereka inginkan untuk bergerak dan menyelesaikan turn saat ini.

   Setiap kali turn baru dimulai, penghitung waktu ulang diatur ulang dan mulai berjalan. Jika batas waktu tercapai dan bot tidak mengirimkan pergerakannya untuk turn tersebut, maka tidak ada perintah yang dikirim ke server. Akibatnya, bot akan melewatkan turn tersebut. Jika bot melewatkan turn, ia tidak akan bisa menyesuaikan gerakannya atau menembakkan senjatanya karena server tidak menerima perintah tepat waktu sebelum turn berikutnya dimulai.

3. Energi

   Semua bot memulai permainan dengan jumlah energi awal sebanyak 100 poin energi.

   - Bot akan kehilangan energi jika ditembak atau ditabrak oleh bot musuh.
   - Bot juga akan kehilangan energi jika menembakkan meriamnya.
   - Bot akan mendapatkan energi jika peluru dari meriamnya mengenai musuh. Energi yang didapat akan lebih banyak 3 kali lipat dari energi yang digunakan untuk menembakkan peluru.
   - Bot dengan energi nol akan dinonaktifkan dan tidak bisa bergerak. Jika bot terkena serangan dalam keadaan ini, bot akan hancur.

4. Peluru

   Semakin banyak energi (daya tembak) yang digunakan untuk menembakkan peluru, semakin berat peluru tersebut dan semakin lambat gerakannya. Namun, peluru yang lebih berat juga menghasilkan lebih banyak kerusakan dan memungkinkan bot mendapatkan lebih banyak energi saat mengenai bot musuh.

   Seperti disebutkan sebelumnya, peluru yang lebih berat akan bergerak lebih lambat. Ini berarti akan membutuhkan waktu lebih lama untuk mencapai target, meningkatkan risiko peluru tidak mengenai sasaran. Sebaliknya, peluru yang lebih ringan bergerak

lebih cepat, sehingga lebih mudah mengenai target, tetapi peluru ringan tidak memberikan banyak poin energi saat mengenai bot musuh.

5. Panas Meriam (Gun Heat)

   Saat menembakkan peluru, meriam akan menjadi panas. Peluru yang lebih berat menghasilkan lebih banyak panas dibandingkan peluru yang lebih ringan. Ketika meriam terlalu panas, bot tidak dapat menembak hingga suhu meriam turun ke nol. Selain itu, meriam juga sudah dalam keadaan panas di awal round dan perlu waktu untuk mendingin sebelum bisa digunakan untuk pertama kalinya.

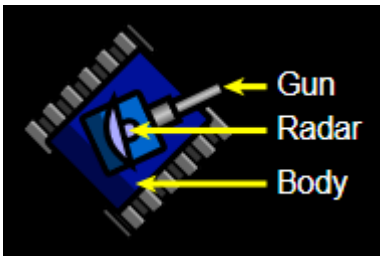
6. Tabrakan

   Perlu diperhatikan bahwa bot akan menerima kerusakan jika menabrak dinding (batas arena), yang disebut wall damage. Hal yang sama juga terjadi jika bot bertabrakan dengan bot lain.

   Jika bot menabrak bot musuh dengan bergerak maju, ini disebut ramming (menabrak dengan sengaja), yang akan memberikan sedikit skor tambahan bagi bot yang menyerang.

7. Bagian Tubuh Tank

   Tubuh tank terdiri dari 3 bagian:

   Body adalah bagian utama dari tank yang digunakan untuk menggerakkan tank.

   Gun digunakan untuk menembakkan peluru dan dapat berputar bersama *body* atau independen dari *body*.

   Radar digunakan untuk memindai posisi musuh dan dapat berputar bersama *body* atau independen dari *body*.

8. Pergerakan

   Bot dapat bergerak maju dan mundur hingga kecepatan maksimum. Dibutuhkan beberapa giliran untuk mencapai kecepatan maksimum. Bot dapat mengalami

percepatan maksimum sebesar 1 unit per giliran dan pengereman dengan perlambatan maksimum 2 unit per giliran. Percepatan dan perlambatan maksimum tidak bergantung pada kecepatan bot saat itu.

9. Berbelok

   Seperti yang disebutkan sebelumnya, bagian tubuh, turret (meriam), dan radar dapat berputar secara independen satu sama lain. Jika turret atau radar tidak diputar, maka keduanya akan mengarah ke arah yang sama dengan tubuh bot.

   Setiap bagian tubuh memiliki kecepatan putar yang berbeda. Radar adalah bagian tercepat dan dapat berputar hingga 45 derajat per giliran, yang berarti dapat berputar 360 derajat dalam 8 giliran. Turret dan meriam dapat berputar hingga 20 derajat per giliran.

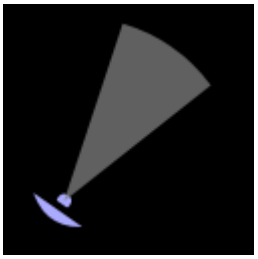
   Bagian paling lambat adalah tubuh tank, yang dalam kondisi terbaik dapat berputar hingga 10 derajat per giliran. Namun, ini bergantung pada kecepatan bot saat ini. Semakin cepat bot bergerak, semakin lambat kemampuannya untuk berbelok.

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

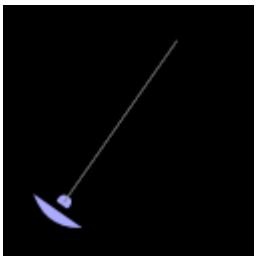
10. Pemindaian

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

    Penting untuk diperhatikan bahwa sebuah bot hanya dapat memindai bot musuh yang berada dalam jangkauan sudut pemindaian (scan arc)-nya. Sudut pemindaian ini merupakan "sapuan radar" dari arah radar sebelumnya ke arah radar saat ini dalam satu giliran.



Jika radar tidak bergerak dalam suatu giliran, artinya radar tetap mengarah ke arah yang sama seperti pada giliran sebelumnya, maka sudut pemindaian akan menjadi nol derajat, dan bot tidak akan dapat mendeteksi musuh.



Oleh karena itu, sangat disarankan untuk selalu mengubah arah radar agar tetap dapat memindai musuh.

11. Skor

Pada akhir pertempuran, setiap bot akan diranking berdasarkan total skor yang diperoleh masing-masing bot selama keseluruhan pertempuran. Tentunya, tujuan utama pada tugas besar ini adalah membuat bot yang memberikan skor setinggi mungkin. Berikut adalah rincian komponen skor pada pertempuran:

- **Bullet Damage:** Bot mendapatkan **poin sebesar *damage*** yang dibuat kepada bot musuh menggunakan peluru.
- **Bullet Damage Bonus:** Apabila peluru berhasil membunuh bot musuh, bot mendapatkan **poin sebesar 20% dari *damage*** yang dibuat kepada musuh yang terbunuh.
- **Survival Score:** Setiap ada bot yang mati, bot lainya yang masih bertahan pada ronde tersebut mendapatkan **50 poin.**
- **Last Survival Bonus:** Bot terakhir yang bertahan pada suatu ronde akan mendapatkan **10 poin** dikali dengan banyaknya musuh.
- **Ram Damage:** Bot mendapatkan **poin sebesar 2 kalinya *damage*** yang dibuat kepada bot musuh dengan cara menabrak.
- **Ram Damage Bonus:** Apabila musuh terbunuh dengan cara ditabrak, bot mendapatkan **poin sebesar 30% dari *damage*** yang dibuat kepada musuh yang terbunuh.

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

Bab 2 Landasan Teori

Algoritma greedy adalah pendekatan penyelesaian masalah dengan mengikuti metode heuristik, yaitu memilih langkah terbaik pada setiap tahap dengan harapan bahwa pilihan lokal yang optimal akan menghasilkan solusi global yang optimal. Algoritma ini memiliki prinsip "pilih yang terbaik untuk saat ini" dengan sedikit pertimbangan konsekuensi jangka panjang.

Dalam kasus Robocode, menentukan solusi optimal yang global, yakni membuat bot yang dapat mengalahkan sebanyak-banyaknya bot yang lain, adalah permasalahan yang sangat sulit. Hal ini disebabkan karena banyaknya ruang kemungkinan keadaan dari sebuah ronde permainan. Berbeda dengan permainan papan seperti catur, dam, atau monopoli yang memiliki jumlah petak dan langkah yang diskrit, Robocode dimainkan pada arena yang lebih luas dan kemungkinan gerak yang kontinu.

Oleh karena itu, membuat bot yang optimal tidak bisa dilakukan dengan exhaustive search atau cara yang semacamnya. Akan tetapi, solusi optimal dapat didekati dengan menggunakan heuristik yang digunakan untuk menentukan langkah terbaik pada saat tertentu berdasarkan keadaan bot tersebut pada saat itu juga.

Bot Robocode bekerja dengan siklus event-driven, yaitu bot akan merespon berbagai event seperti mendeteksi musuh, terkena tembakan, atau menabrak bot lain/tembok. Secara umum, bot dimulai dengan pengaturan dasar seperti warna, nama, dan parameter awal. Kemudian bot digerakkan dengan cara mendeskripsikan respon-respon untuk berbagai event dari lingkungan permainan. Respon tersebut memutuskan gerakan yang dilakukan bot, seperti memindai, bergerak maju/mundur, berbelok, dan menembak.

Untuk mengimplementasikan algoritma greedy dalam bot Robocode, perlu diidentifikasi heuristik yang digunakan untuk memilih keputusan lokal yang perlu diambil pada setiap langkah. Misalnya, jika bot memindai bot lain, bot akan menembak bot tersebut. Jika bot akan menabrak bot lain atau tembok, bot akan menghindar.

Bab 3 Aplikasi Strategi *Greedy*

3.1 Proses Mapping Persoalan Robocode Tank Royale Menjadi Elemen-Elemen Algoritma Greedy

Untuk menerapkan algoritma greedy pada Robocode Tank Royale, elemen-elemen persoalan dipetakan ke dalam komponen-komponen algoritma greedy. Proses mapping ini membantu merancang bot yang dapat mengambil keputusan optimal pada setiap langkah.

3.1.1 Himpunan Kandidat

Himpunan kandidat dalam Robocode Tank Royale terdiri dari semua kemungkinan aksi yang dapat dilakukan bot pada setiap langkah. Himpunan kandidat sebuah bot adalah bergerak maju/mundur dengan berbagai kecepatan, berputar dengan berbagai sudut, menembak dengan berbagai kekuatan, memutar radar, dan kombinasi dari aksi-aksi tersebut.

3.1.2 Himpunan Solusi

Himpunan Solusi merupakan sekumpulan aksi yang telah dipilih oleh bot sepanjang pertandingan. Dalam konteks Robocode, solusi ini berbentuk sequence dari aksi-aksi yang membentuk strategi keseluruhan bot dalam menghadapi pertandingan. Sehingga, himpunan solusi adalah semua strategi perilaku bot yang mungkin.

3.1.3 Fungsi Solusi

Fungsi Solusi menentukan apakah satu set aksi yang dipilih telah membentuk solusi lengkap. Dalam Robocode, semua himpunan kandidat sudah merupakan solusi yang valid, karena setiap robot bebas melakukan pergerakan apapun selama pertandingan berlangsung. Sehingga fungsi solusi bernilai benar untuk semua himpunan kandidat.

3.1.4 Fungsi Seleksi

Fungsi Seleksi adalah kriteria yang digunakan untuk memilih kandidat terbaik pada setiap langkah. Dalam Robocode, fungsi ini berupa heuristik yang digunakan oleh robot untuk memutuskan langkah terbaik. Fungsi ini dapat berupa pemilihan target berdasarkan jarak terdekat, pemilihan kekuatan tembakan berdasarkan jarak dan energi musuh, atau pemilihan gerakan yang menjauhkan dari musuh berbahaya. Fungsi ini memilih pilihan yang tampak optimal pada momen tersebut tanpa pertimbangan jangka panjang.

3.1.5 Fungsi Kelayakan

Fungsi Kelayakan mengevaluasi apakah suatu kandidat aksi dapat diterima untuk ditambahkan ke dalam himpunan solusi. Dalam Robocode, fungsi ini memeriksa apakah aksi yang akan dipilih dapat dilaksanakan pada kondisi saat ini, seperti memastikan energi cukup untuk menembak, posisi tidak akan menabrak dinding, atau gerakan tidak akan menyebabkan tabrakan dengan robot lain.

3.1.6 Fungsi Objektif

Fungsi objektif adalah ukuran keseluruhan yang mengevaluasi seberapa baik solusi yang dihasilkan. Dalam Robocode, fungsi yang ingin dimaksimalkan adalah skor yang didapatkan oleh bot. Fungsi ini menjadi tujuan utama yang ingin dimaksimalkan melalui serangkaian keputusan greedy.

3.2 Eksplorasi Alternatif Solusi

3.2.1 Conquest

Bot ini memiliki heuristik yang memanfaatkan pemindaian yang dilakukan secara terus menerus dengan kondisi pergerakan robot yang bergerak spiral dengan tujuan menghindari peluru-peluru dari lawan. Ketika bot memindai bot musuh, maka bot akan memprediksi posisi musuh berdasarkan kecepatan dan arah gerakan musuh saat dideteksi oleh radar, informasi ini akan digunakan untuk mengarahkan senjata ke posisi prediksi musuh dan menembak dengan kekuatan yang bergantung pada jarak bot dan bot musuh yang terdeteksi. Jika musuh berada dalam jarak dekat (kurang dari 500 unit), bot akan menembak dengan kekuatan sebesar 3, jika musuh berada pada jarak yang lebih jauh, bot akan menembak dengan kekuatan 2.

Ketika bot bertabrakan dengan bot musuh, bot akan mundur sedikit sejauh 50 unit dan berputar 90 derajat ke kanan. Ini membuat bot dapat menghindari tabrakan berulang atau menghindari bot musuh yang memiliki spesialisasi jarak dekat. Bot dibuat agar dapat menghadapi bot musuh jenis apapun.

3.2.2 Omniman

Bot ini memiliki heuristik yang membuatnya menjadi robot yang ofensif baik untuk jarak jauh maupun jarak dekat. Bot akan maju dan berputar ke kanan secara bergantian untuk bisa aktif di arena pertandingan sembari memutar radar 360 derajat secara terus-menerus. Ketika berhasil memindai bot musuh, bot akan terus menembak dengan kekuatan yang juga disesuaikan berdasarkan jarak. Jika jarak <100, maka bot akan masuk ke dalam mode pertarungan jarak dekat dimana bot akan fokus menghadap ke satu musuh dan menembaknya terus-menerus dan maju disaat yang bersamaan agar mendapatkan poin *ram*. Jika musuh berada pada jarak >100 namun <500 maka bot akan menembak dengan kekuatan 3, jika musuh berada pada jarak yang lebih jauh, maka bot akan menembak dengan kekuatan yang lebih rendah agar peluru yang dikeluarkan lebih cepat dan lebih menghemat energi.

Ketika bot bertabrakan dengan bot musuh, bot akan menghadap ke arah musuh yang menabrak, lalu menembak dengan kekuatan maksimal dan bergerak maju untuk memastikan bot dapat memanfaatkan tabrakan dan menyerang musuh dengan agresif.

3.2.3 Invincible

Bot ini dirancang agar benar-benar "*Invincible*". Bot ini memiliki heuristik yang memanfaatkan pergerakan secara spiral, kemudian jika dalam proses gerak memutar tersebut terpindai bot lain, maka bot akan secara berkala bergerak ke arah bot target tersebut untuk memungkinkan terjadinya *ram* (tumbukan). Bot akan mendekat tergantung jarak bot lain, jika sangat jauh (di atas 800) bot akan mendekat sebesar 0.2, jika di bawah 800 bot akan mendekat sebesar 0.3, jika di bawah 100 bot akan mendekat sebesar 0.5, dan jika di bawah 50 bot akan melakukan ram dengan menembakkan peluru untuk memaksimalkan poin.

Penembakan bot juga dilakukan secara *greedy* untuk memberikan serangan yang optimal, yakni menggabungkan kondisi energi yang tersisa dengan jarak bot target maka diharapkan tembakan yang dikeluarkan dapat memberikan poin maksimal dan tidak merugikan bot sendiri. Apabila energi bot semakin menipis atau jarak bot dengan bot target semakin jauh, maka peluru yang dikeluarkan juga semakin kecil, begitu pula sebaliknya sehingga energi yang ada dimanfaatkan dengan baik.

Selanjutnya ditambah pula dengan kemampuan untuk menghindari tembok, dalam hal ini, asumsi lebar tembok adalah 20 sehingga menjadikan bot ini tidak akan mengenai tembok dan tidak membuang-buang energi.

3.2.4 Allen

Bot ini memiliki heuristik yang sederhana. Bot akan terus bergerak memutar dan memindai selama bot tidak menemukan bot lain dan tidak menabrak bot lain maupun dinding. Saat bot menemukan lawan, bot akan berusaha untuk menembak lawan dengan peluru berkekuatan 2. Saat bot menabrak lawan, bot akan berbelok ke kanan, dan saat bot menabrak dinding, bot berputar balik.

3.3 Strategi yang Dipilih

Berdasarkan strategi greedy yang dikumpulkan berikut analisis efektivitas dan efisiensi :

- Conquest

Bot ini berfokus pada strategi bertahan dengan gerakan spiral yang efektif menghindari tembakan lawan sekaligus melakukan pemindaian radar secara terus-menerus untuk mendeteksi posisi musuh. Dengan pendekatan ini, bot secara konsisten memprediksi posisi musuh berdasarkan arah dan kecepatan geraknya ketika terdeteksi oleh radar sehingga dapat meningkatkan akurasi tembakan. Efektivitas serangan juga meningkat dengan adanya penyesuaian kekuatan tembakan berdasarkan jarak di mana apabila musuh berada dalam jarak kurang dari 500 unit, bot menembak dengan kekuatan tinggi (3), sedangkan untuk musuh yang lebih jauh, bot menggunakan kekuatan lebih rendah (2) untuk menghemat energi sekaligus menjaga efektivitas serangan. Selain itu, bot juga dilengkapi dengan mekanisme defensif sederhana yang akan mundur sejauh 50 unit dan berputar 90 derajat ke kanan saat terjadi tabrakan dengan lawan, sehingga mengurangi risiko serangan bertubi-tubi maupun serangan lanjutan dari bot musuh bertipe jarak dekat. Namun, strategi ini memiliki kelemahan, seperti potensi prediksi posisi yang kurang akurat jika musuh bergerak secara acak atau berubah arah secara tiba-tiba, serta gerakan spiral yang mungkin bisa ditebak oleh lawan yang lebih canggih. Pemindaian radar secara kontinu juga efektif dalam mendeteksi musuh tetapi berpotensi menghabiskan sumber daya komputasi.

- Omniman

  Bot ini memiliki strategi ofensif yang efektif dalam pertarungan baik jarak dekat maupun jarak jauh. Strategi pergerakannya yang maju sambil berputar secara bergantian, dikombinasikan dengan pemindaian radar 360 derajat secara terus-menerus memberikan keunggulan berupa deteksi musuh yang cepat serta kemampuan mendekati lawan secara agresif. Efektivitas serangannya meningkat melalui adaptasi kekuatan tembakan yang disesuaikan berdasarkan jarak musuh: untuk jarak dekat (<100 unit), bot melakukan serangan agresif sembari maju untuk mendapatkan poin tambahan melalui ramming; untuk jarak menengah (antara 100–500 unit), bot menembak dengan kekuatan tinggi agar lebih efektif mengenai musuh; dan untuk jarak jauh (>500 unit), bot menggunakan kekuatan tembakan yang lebih rendah demi meningkatkan kecepatan proyektil sekaligus menghemat energi. Dari sisi efisiensi, strategi bot ini terbilang hemat energi karena menyesuaikan kekuatan serangan secara adaptif, terutama untuk target yang jauh. Namun, pendekatan ofensif yang dominan ini juga memiliki kelemahan berupa risiko tinggi

terhadap serangan balik, terutama jika menghadapi lawan yang menggunakan strategi bertahan atau memiliki mobilitas tinggi. Pemindaian radar terus-menerus memang sangat efektif dalam pendeteksian musuh, tetapi bisa membebani sumber daya komputasi jika tidak dikelola dengan baik.

- Invincible

  Bot ini dibuat memiliki sifat agresif yang menggabungkan strategi pergerakan spiral untuk mendeteksi keberadaan musuh sekaligus mendekatinya secara adaptif. Setelah lawan terdeteksi, bot akan mendekati target dengan menyesuaikan jarak musuh hingga akhirnya memungkinkan terjadinya ramming (tumbukan) di jarak sangat dekat (<50 unit). Selain bergerak agresif, bot ini juga secara optimal melakukan penembakan menggunakan pendekatan greedy yang mempertimbangkan kondisi energi bot sendiri serta jarak dengan target sehingga peluru yang ditembakkan efektif menghasilkan poin maksimal tanpa membuang energi secara sia-sia. Efisiensi tambahan diperoleh melalui sistem penghindaran tembok dengan asumsi lebar tembok sebesar 20 unit, yang membuat bot terhindar dari tabrakan sia-sia yang menguras energi. Namun, pendekatan agresif yang dominan ini memiliki kelemahan karena dapat meningkatkan risiko terkena serangan balik, terutama jika lawan memiliki strategi bertahan yang baik atau mobilitas tinggi.

- Allen

  Bot ini dirancang sederhana, tetapi cukup efektif dalam kondisi pertarungan umum. Strategi utamanya adalah bergerak secara kontinu dengan gerakan memutar sambil secara aktif melakukan pemindaian radar. Ketika bot berhasil menemukan musuh, bot langsung menembak lawan dengan peluru berkekuatan tetap sebesar 2. Jika terjadi tabrakan dengan bot musuh, bot secara otomatis berbelok ke kanan agar dapat segera keluar dari situasi tersebut dan melanjutkan pola geraknya. Begitu pula ketika bot menabrak dinding arena, bot akan secara otomatis berputar balik untuk menghindari hambatan fisik tersebut. Meski sederhana dan efisien dalam pemanfaatan sumber daya komputasi, strategi ini memiliki keterbatasan, seperti kurangnya adaptasi kekuatan tembakan terhadap jarak lawan serta minimnya gerakan menghindar yang membuatnya rentan terhadap serangan balik, baik dari musuh yang memiliki strategi ram ataupun musuh yang melakukan serangan jauh secara bertubi-tubi.

Berdasarkan analisis efektivitas dan efisiensi di atas diputuskan bahwa strategi bot Conquest yang akan dipilih sebagai bot utama dalam permainan ini. Hal ini karena bot tersebut memiliki kemampuan yang baik dalam mengatasi bot yang memiliki serangan jarak jauh dan mampu menghindar dengan baik jika berhadapan dengan bot yang memiliki strategi ram. Meskipun bot ini memiliki kelemahan ketika terjadi banyak tumbukan, tapi dalam pengujian dan arena sebenarnya bot tidak akan mengalami banyak tumbukan sehingga kelemahan ini dapat diabaikan, menjadikan Conquest dapat bertahan dengan sangat baik ke depannya.

Bab 4 Implementasi dan pengujian.

4.1 Implementasi Alternatif Solusi

4.1.1 Conquest

```
Run()
{
    BodyColor = Color.White;
    TurretColor = Color.WhiteSmoke;
    RadarColor = Color.DarkGray;
    BulletColor = Color.Gold;
    ScanColor = Color.Red;

    AdjustGunForBodyTurn = true;
    AdjustRadarForBodyTurn = true;
    AdjustRadarForGunTurn = true;

    double moveAmount = Math.Max(ArenaWidth, ArenaHeight);
    TurnRight(Direction % 90);
    Forward(moveAmount);
    TurnRight(90);

    while (IsRunning)
    {
        SetTurnRadarRight(360);
        SetForward(40);
        SetTurnRight(40);
        SetForward(40);
        SetTurnLeft(40);

        Go();
    }
}
OnScannedBot(ScannedBotEvent e)
{
    double enemyX = e.X;
    double enemyY = e.Y;
    double enemyVelocity = e.Speed;
```

```
    double enemyHeading = e.Direction;

    double predictedX = enemyX + enemyVelocity * Math.Sin(enemyHeading *
Math.PI / 180);
    double predictedY = enemyY + enemyVelocity * Math.Cos(enemyHeading *
Math.PI / 180);

    double bearing = GunBearingTo(predictedX, predictedY);
    TurnGunLeft(bearing);

    double distance = DistanceTo(predictedX, predictedY);
    if (distance < 500){
        Fire(3);
    }else{
        Fire(2);
    }

    lastEnemyX = enemyX;
    lastEnemyY = enemyY;
    lastEnemyVelocity = enemyVelocity;
    lastEnemyDistance = distance;
}
OnHitBot(HitBotEvent e)
{
    SetBack(50);
    TurnRight(90);
    Go();

}
```

4.1.2 Omniman

```
Run()
{
    BodyColor = Color.White;
    TurretColor = Color.FromArgb(255, 209, 0, 0);
    RadarColor = Color.FromArgb(255, 209, 0, 0);
    BulletColor = Color.FromArgb(255, 209, 0, 0);
    ScanColor = Color.Red;

    AdjustGunForBodyTurn = true;
    AdjustRadarForBodyTurn = true;
    AdjustRadarForGunTurn  = true;

    moveAmount = Math.Max(ArenaWidth, ArenaHeight);
    TurnRight(Direction % 90);
    Forward(moveAmount);
    TurnRight(90);
    while (IsRunning)
```

```
    {
        SetTurnRadarRight(360);
        SetForward(40);
        SetTurnRight(40);
        SetForward(20);
        Go();
}
OnScannedBot(ScannedBotEvent e)
{
    TurnGunLeft(GunBearingTo(e.X, e.Y));
    var distance = DistanceTo(e.X, e.Y);
    if (distance < 500){
        Fire(3);
    }else{
        Fire(2);
    }
}
OnHitBot(HitBotEvent e)
{
    TurnToFaceTarget(e.X, e.Y);
    Fire(3);
    Forward(30);
}
TurnToFaceTarget(double x, double y)
{
    var bearing = BearingTo(x, y);
    if (bearing >= 0)
        turnDirection = 1;
    else
        turnDirection = -1;

    TurnLeft(bearing);
}
```

4.1.3 Invincible

```
Run()
{
       BodyColor = Color.Black;
       TurretColor = Color.DarkBlue;
       RadarColor = Color.Blue;
       BulletColor = Color.Blue;
       ScanColor = Color.LightBlue;

       movingForward = true;

       while (IsRunning){
```

```
            if (IsNearWall(20)){

                AvoidWall();

            }

            else {

                SetTurnRight(5000);

                MaxSpeed = 5;

                Forward(10000);

            }

        }

}

OnScannedBot(ScannedBotEvent e)
{
   double distance = DistanceTo(e.X, e.Y);
   double firePower = DetermineFirePower(distance);

   if (distance <= 50){
       Fire(firePower);
       TurnToFaceTarget(e.X, e.Y);
       Fire(firePower);
       Forward(distance + 5);
    }
    else if (distance <= 100) {
       Fire(firePower);
       TurnToFaceTarget(e.X, e.Y);
       Forward(distance * 0.5);
}
    else if (distance <= 800){
        Fire(firePower);
        TurnToFaceTarget(e.X, e.Y);
        Forward(distance * 0.3);
    }
    else{
        Fire(firePower);
        TurnToFaceTarget(e.X, e.Y);
        Fire(firePower);
        Forward(distance * 0.2);
    }
}
OnHitBot(HitBotEvent e)
{
    var bearing = BearingTo(e.X, e.Y);
    if (bearing > -10 && bearing < 10)
```

```
    {
        Fire(3);
    }
    if (e.IsRammed)
    {
        TurnLeft(10);
        ReverseDirection();
    }
}

OnHitWall(HitWallEvent e)
{
      ReverseDirection();
}

ReverseDirection()
{
    if (movingForward)
    {
        SetBack(200);
        movingForward = false;
    }
    else
    {
        SetForward(200);
        movingForward = true;
    }
}

TurnToFaceTarget(double x, double y)
{
    var bearing = BearingTo(x, y);
    if (bearing >= 0)
        turnDirection = 1;
    else
        turnDirection = -1;

    TurnLeft(bearing);
}

DetermineFirePower(double distance)
{
    if (Energy > 60)
    {
        if (distance <= 500) return 3;
        return 2;
    }

    if (Energy > 20)
    {
        if (distance <= 50) return 3;
        if (distance <= 100) return 2;
```

```
        return 1;
    }

    if (Energy > 10)
    {
        if (distance <= 50) return 2;
        return 1;
    }

    return 0.5;
}

IsNearWall(double margin)
{
    double arenaWidth = ArenaWidth;
    double arenaHeight = ArenaHeight;

    if (X <= margin || X >= (arenaWidth - margin)) return true;
    if (Y <= margin || Y >= (arenaHeight - margin)) return true;

    return false;
}

AvoidWall()
{
    TurnRight(90);
    Back(100);
}
```

4.1.4 Allen

```
Run()
{
    // atur atribut bot
    BodyColor = Color.FromArgb(255, 80, 200, 80);
    TurretColor = Color.White;
    RadarColor = Color.Orange;
    BulletColor = Color.Orange;
    ScanColor = Color.FromArgb(255, 200, 230, 200);

    AdjustGunForBodyTurn = true;
    AdjustRadarForBodyTurn = true;
    AdjustRadarForGunTurn = true;
    // gerakan default
    while (IsRunning)
    {
        SetForward(10);
        SetTurnRadarRight(360);
```

```
        SetTurnRight(10);
        Go(); // jalan berputar sambil menggerakkan radar
    }
}
OnScannedBot(ScannedBotEvent e)
{
    TurnGunLeft(GunBearingTo(e.X, e.Y)); //arahkan meriam ke lawan
    Fire(2);
}
OnHitBot(HitBotEvent e)
{
    TurnRight(90);
}
OnHitWall(HitWallEvent e)
{
    TurnRight(180);
}
```

4.2 Penjelasan Implementasi

Bot Conquest memiliki beberapa variabel yang digunakan untuk menyimpan informasi terkait dengan keadaan bot musuh, diantaranya:

- lastEnemyX, lastEnemyY = untuk menyimpan koordinat terakhir musuh yang terpindai
- lastEnemyVelocity = untuk menyimpan kecepatan terakhir musuh yang terpindai
- lastEnemyDistance = untuk menyimpan jarak terakhir ke musuh yang terpindai

Variabel tersebut digunakan untuk memprediksi pergerakan bot musuh sehingga bot dapat menembak musuh dengan lebih baik dan lebih efektif.

Bot ini memiliki beberapa fungsi lain selain fungsi Main, diantaranya fungsi Run() yang menjadi titik utama saat bot mulai beroperasi, dimana pada fungsi ini menginisialisasi warna, pergerakan awal dan menjalankan *looping* utama untuk memindai dan bergerak.

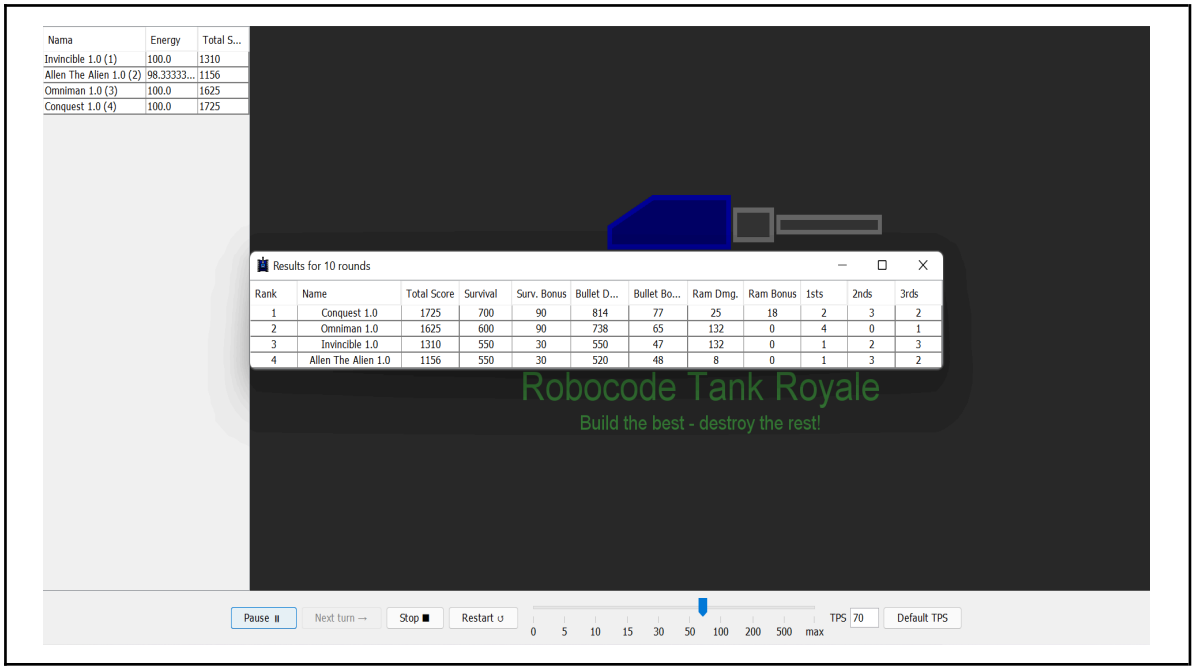
Selanjutnya ada fungsi OnScannedBot yang akan dipanggil ketika bot memindai bot musuh. Pada fungsi ini bot menggunakan informasi keadaan musuh untuk memprediksi posisi musuh lalu menembak musuh berdasarkan kalkulasi prediksi, kekuatan tembakan juga diatur dan disesuaikan berdasarkan jarak antara bot dengan bot musuh.

Selain itu juga ada fungsi OnHitBot yang akan dipanggil ketika bot bertabrakan dengan musuh. Ketika bertabrakan dengan bot musuh, maka bot akan mundur sejauh

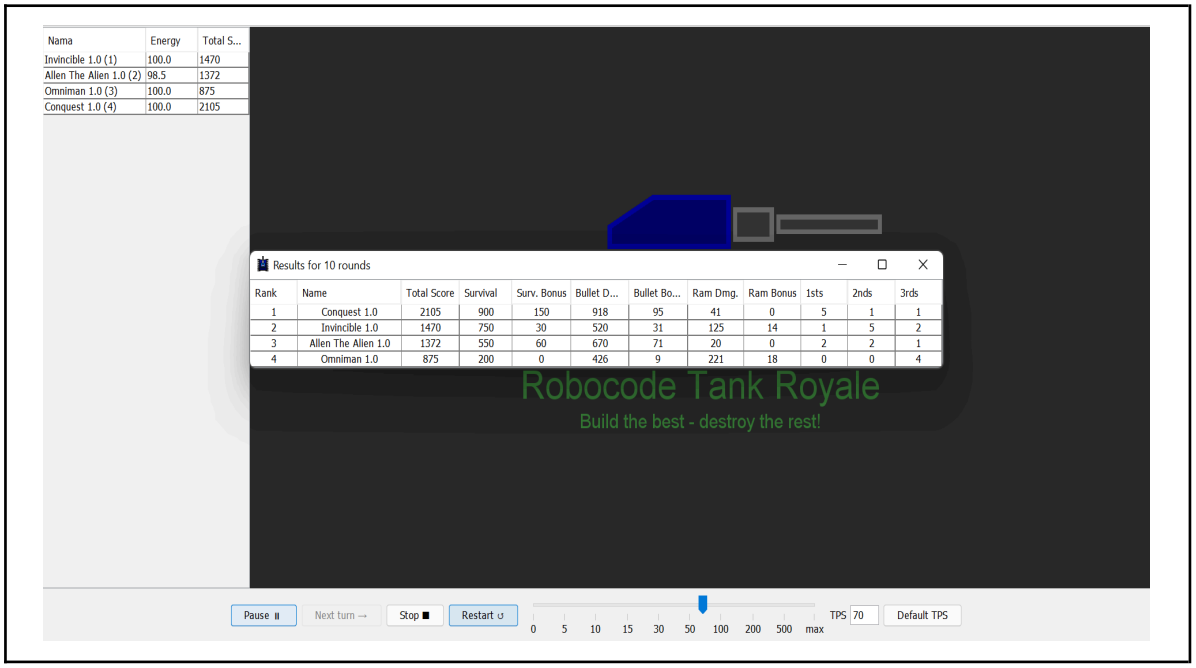
50 unit lalu berputar 90 derajat ke kanan untuk menghindari tabrakan lebih lanjut dan agar bot dapat segera mencari posisi yang lebih baik untuk bisa menembak.

4.3 Pengujian

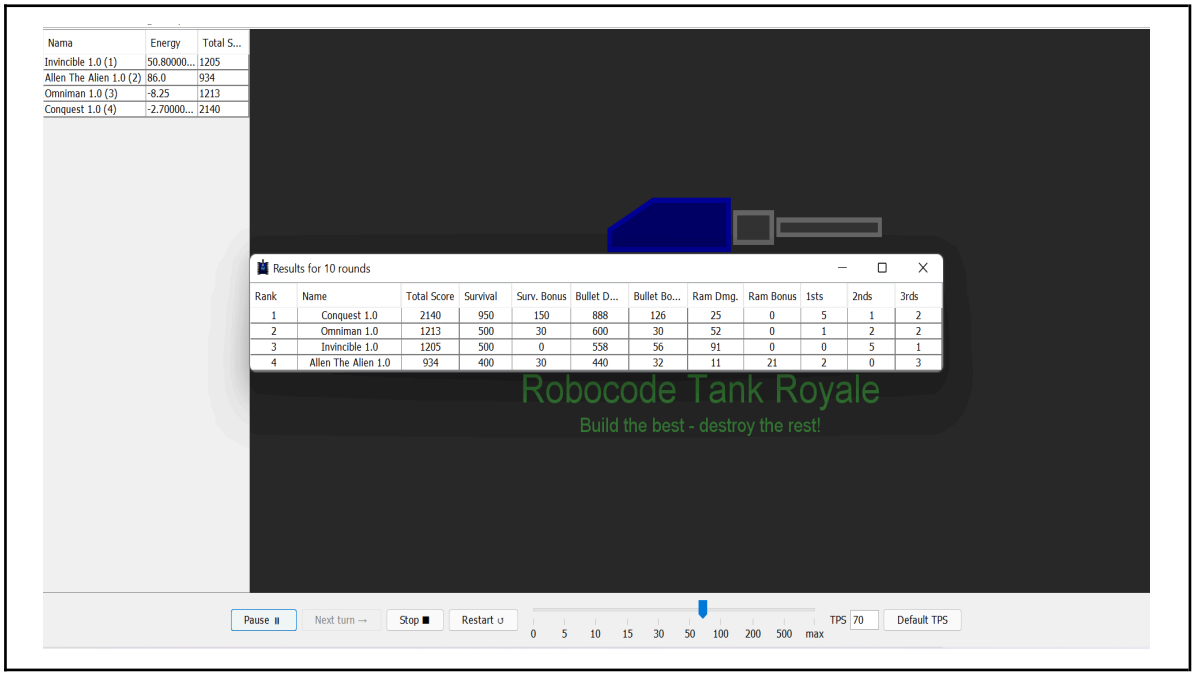
Hasil Pengujian 1



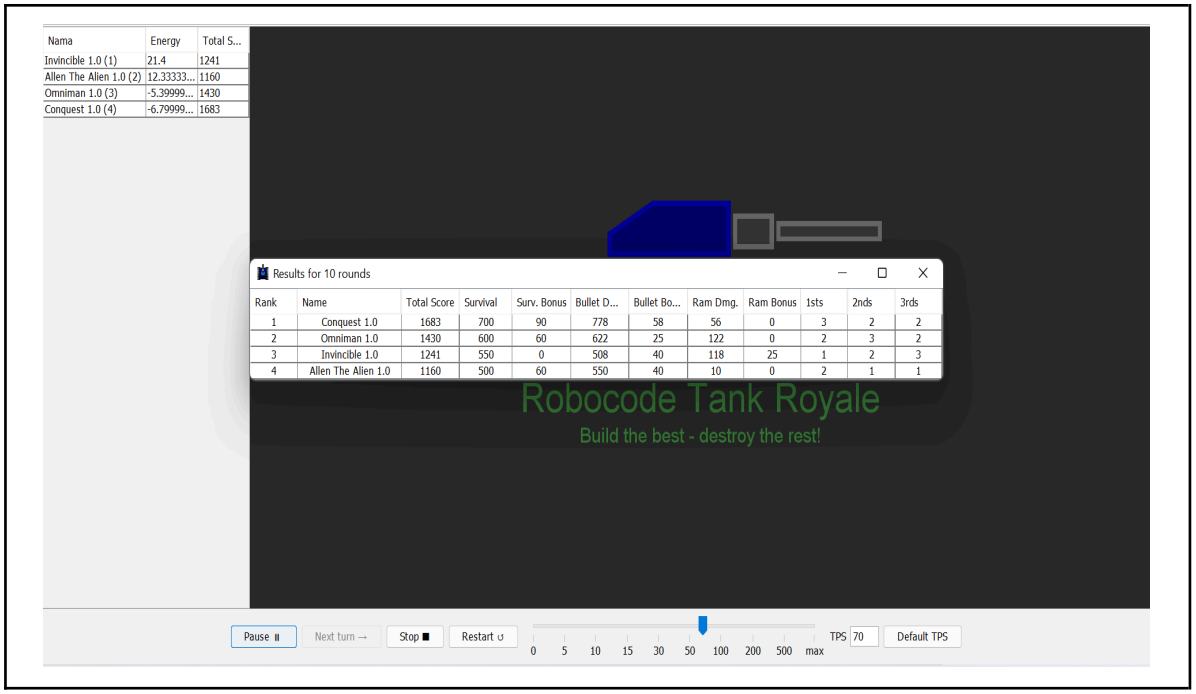
Hasil Pengujian 2



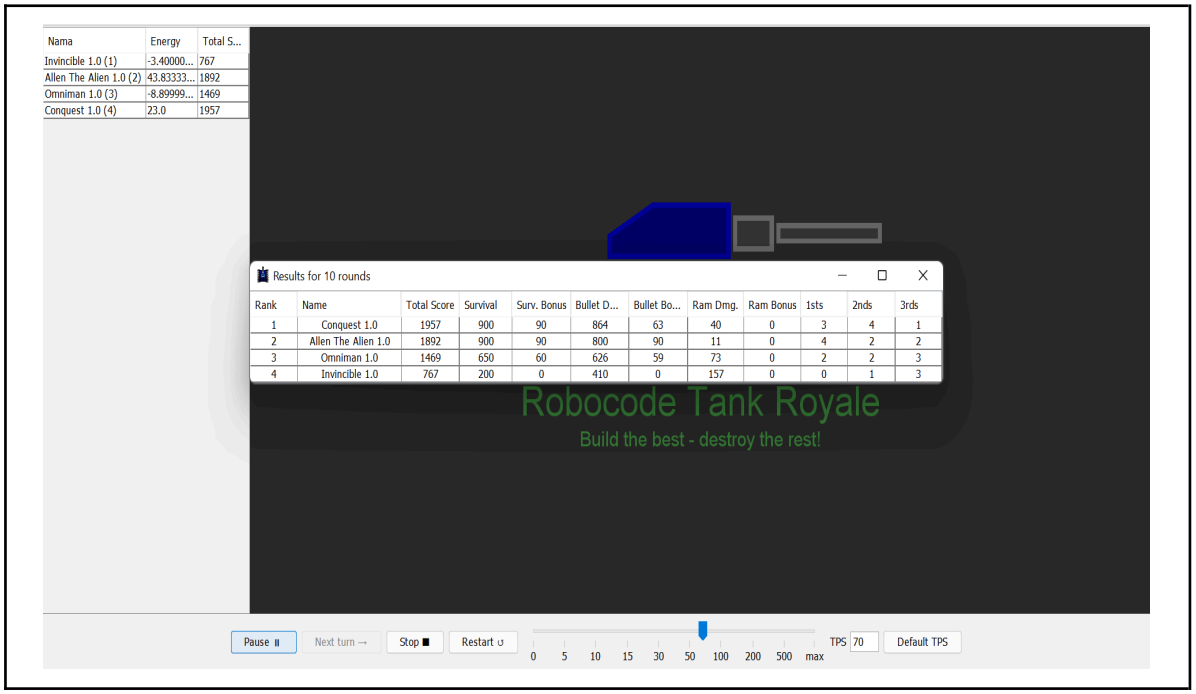
Hasil Pengujian 3



Hasil Pengujian 4



Hasil Pengujian 5



4.4 Analisis

4.4.1 Conquest

Bot ini mampu mengatasi serangan bot jarak jauh karena di samping pergerakannya yang memutar menjadikannya sulit terkena peluru dari bot musuh, bot ini juga mampu melancarkan serangan pelurunya dengan akurasi tinggi sehingga memaksimalkan perolehan poin dan pemanfaatan energi. Bot ini mampu menghindari serangan ram saat pengujian yang terbukti sukses bertahan dan kembali menyerang dengan agresif menjadikannya bertahan hingga akhir ronde.

Pada pengujian, bot sudah mendapatkan nilai optimalnya, hal ini dapat dilihat dari poin *survival* dan poin kerusakan pelurunya yang sangat tinggi selama pengujian berlangsung, apalagi jika tumbukan yang terjadi tidak terlalu sering sehingga bot memanfaatkan waktunya sebaik mungkin untuk melakukan serangan dan memperoleh poin sebanyak-banyaknya.

4.4.2 Omniman

Bot yang memiliki kemampuan untuk melakukan serangan jarak jauh untuk musuh yang berada jauh dari posisinya dan serangan ram untuk musuh jarak dekat ini terbukti sudah melakukan tugasnya dengan optimal pada saat pengujian. Hal ini ditunjukkan dengan tingginya kerusakan yang ditimbulkan dari peluru maupun

kerusakan ram. Bot ini sudah memaksimalkan kedua kemampuannya dengan sangat baik meskipun terkadang bot menjadi cepat hancur ketika dikerumuni banyak bot atau ketika berada di tengah arena.

Dengan sedikitnya jumlah bot yang ada di dalam pengujian (4 bot), memungkinkan bot ini bertahan lebih lama karena minimnya serangan saat bot sedang dalam kondisi rentan. Kelemahan bot dalam pengujian dapat diabaikan selama pengujian berlangsung. Bot sangat optimal jika hanya terdapat sedikit musuh di arena dan apabila mendapatkan posisi strategis untuk menembak bot-bot lain di arena.

4.4.3 Invincible

Bot memiliki kemampuan untuk melakukan ram kepada bot lain dengan cara mendekat secara berkala. Hal ini merupakan kelebihan sekaligus kekurangan dari bot ini, jika ram dan peluru berhasil mengenai bot lawan maka akan memberikan keuntungan yang besar, tetapi jika tidak maka akan menghabiskan banyak energi dan rentan terhadap serangan jarak jauh.

Kemampuan bot dalam menentukan kekuatan tembak hanya akan optimal jika mengenai lawan, jika tidak maka energi bot akan semakin menipis. Selain itu, kondisi optimal bot ini adalah ketika berada dekat dengan musuh dan sedang tidak berada ditengah-tengah arena sehingga meminimumkan terkena serangan dari bot lain sembari memberikan kerusakan ekstra untuk bot terdekat.

Pada pengujian, bot belum sepenuhnya mendapatkan nilai optimal. Hal ini karena energi bot seringkali habis saat berusaha melakukan ram di mana bot lain sedang melakukan serangan bertubi-tubi sehingga bot belum dapat melakukan kerusakan dan lebih dahulu kehabisan energi. Selain itu, pergerakan bot lain yang bisa menghindar peluru dari bot ini terkadang menjadikan peluru bot ini sia-sia tanpa memberikan keuntungan yang berarti.

4.4.5 Allen

Bot ini tidak memiliki mekanisme untuk menghindar dari tembakan lawan, sehingga meskipun lawan mengincar bot ini dan bot ini tertembak beberapa kali, bot tidak tahu cara menjauhi lawan. Bot ini juga tidak menghindar dari potensi tabrakan dengan bot lain dan hanya berganti arah setelah tertabrak, sehingga jika bot ini berpotensi sering ditabrak lawan dan berkurang banyak energi. Selain itu, bot ini juga selalu menembak dengan peluru kekuatan 2 tanpa menyesuaikan jarak dari lawan, sehingga lawan yang

jauh dapat menghindar dan lawan yang dekat seharusnya dapat ditembak dengan peluru yang lebih besar.

Bab 5 Kesimpulan dan saran.

Pendekatan algoritma greedy untuk Robocode merupakan pendekatan yang efektif untuk mengendalikan bot dengan membuat keputusan terbaik di setiap momen berdasarkan kondisi saat itu. Dengan menggunakan heuristik yang melibatkan posisi lawan yang terpindai, jarak dinding, situasi tabrakan, peluru mengenai bot lain, dan lain lain, bot dapat secara langsung memilih aksi yang memberikan keuntungan optimal tanpa mempertimbangkan seluruh kemungkinan di masa depan.

Pendekatan ini cocok digunakan dalam Robocode karena kompleksitas ruang kemungkinan yang tinggi, sehingga pencarian solusi secara menyeluruh (exhaustive search) tidak mungkin. dengan perancangan heuristik yang tepat, bot dapat menunjukkan performa yang baik dalam menghadapi berbagai situasi di pertandingan, walaupun pada konteks ini, algoritma greedy tidak menjamin solusi yang optimal.

Implementasi yang berfokus pada pengumpulan data, evaluasi opsi, dan eksekusi aksi secara efisien menjadikan algoritma greedy sebagai pilihan yang sederhana, cepat, dan efektif untuk membangun bot kompetitif dalam Robocode.

Lampiran

Tautan *repository* GitHub

https://github.com/poetoeee/Tubes1_Omni-Tank.git

Video Simulasi

https://youtu.be/XV5QiBsaEjM?si=l4a3gwinF2HL1vfd

Daftar Pustaka

RoboCode Documentation-RoboWiki. (n.d.). https://robowiki.net/wiki/Robocode_Documentation

Munir, R. (2025). Algoritma Greedy (Bagian 1). Institut Teknologi Bandung. https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2024-2025/04-Algoritma-Greedy-(2025)-Bag1.pdf. Diakses pada 21 Maret 2025.

Munir, R. (2025). Algoritma Greedy (Bagian 2). Institut Teknologi Bandung. https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2024-2025/05-Algoritma-Greedy-(2025)-Bag2.pdf.Diakses pada 21 Maret 2025.

Munir, R. (2025). Algoritma Greedy (Bagian 3). Institut Teknologi Bandung. http://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2024-2025/06-Algoritma-Greedy-(2025)-Bag3.pdf .Diakses pada 21 Maret 2025.