Laporan Tugas Besar 1 - IF2211 Strategi Algoritma
Pemanfaatan Algoritma *Greedy* dalam Pembuatan Bot permainan Robocode Tank Royale
Semester II Tahun 2024/2025

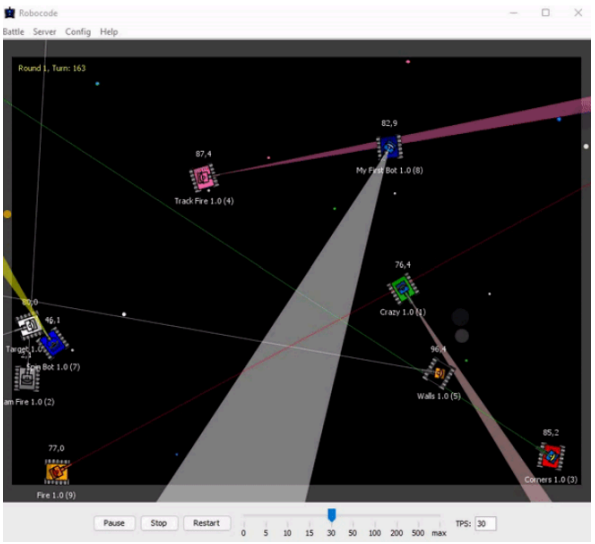
disusun oleh : biskitop (Kelompok 18)
Shannon Aurellius Anastasya Lie (13523019)
Jessica Allen (13523059)

Program Studi Teknik Informatika
Sekolah Teknik Elektro Dan Informatika
Institut Teknologi Bandung

2025

DAFTAR ISI

BAB I
DESKRIPSI TUGAS



Gambar 1 Robocode Tank Royale

Robocode adalah permainan pemrograman yang membuat kode bot dalam bentuk tank virtual untuk berkompetisi melawan bot lain di arena. Pertempuran Robocode berlangsung hingga bot-bot yang 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 tergantung 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.

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.

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 II
LANDASAN TEORI

2.1.Algoritma *Greedy*

Algoritma *greedy* merupakan metode yang paling populer dan sederhana untuk memecahkan persoalan optimasi. Algoritma ini biasanya digunakan untuk menyelesaikan persoalan optimasi (*optimization problems*) yaitu persoalan mencari solusi optimal. Terdapat dua macam persoalan optimasi yaitu maksimasi (*maximization*) dan minimasi (*minimization*). Algoritma *greedy* merupakan suatu algoritma yang memecahkan persoalan secara langkah per langkah (*step by step*) yaitu dengan cara mengambil pilihan yang terbaik yang dapat diperoleh pada saat itu tanpa memperhatikan konsekuensi ke depan (prinsip "*take what you can get now*!") dan "berharap" bahwa dengan memilih optimum lokal pada setiap langkah akan berakhir dengan optimum global.

2.2.Cara Kerja Program Bot

Robocode adalah game pemrograman di mana pemain membuat bot virtual yang bertempur di medan perang melawan bot lainnya. Setiap bot dalam Robocode diimplementasikan menggunakan bahasa pemrograman. Program bot di Robocode ini terdiri dari beberapa komponen utama yang saling berinteraksi, seperti pergerakan, pemindaian musuh, penembakan, dan penghindaran tembakan.

Bot pada program ini bergerak di medan pertempuran dengan menggunakan perintah-perintah yang mengontroll arah dan jarak pergerakan. Selain itu, bot juga memiliki radar yang memindai lingkungan di sekitarnya untuk mendeteksi keberadaan musuh. Radar ini berputar secara berkala, biasanya dengan pola 360 derajat, untuk mengambil tindakan dengan menembak atau menghindar dari serangan musuh. Setiap bot memiliki metode tersendiri yang dapat diprogram untuk mengatur perilakunya.

Bot melakukan berbagai aksi sesuai dengan logika yang diprogramkan padanya. Salah satunya adalah scan atau pemindaian, yang dilakukan untuk mendeteksi posisi musuh di sekitar bot. Ketika radar bot mendeteksi adanya bot musuh, bot akan memutuskan apakah akan menembak atau bergerak untuk menghindari serangan. Fire atau menembak adalah aksi selanjutnya yang dapat dilakukan. Tembakan biasanya dihitung berdasarkan kecepatan dan arah musuh agar tembakan tepat sasaran. Selain menembak dan memindai, bot juga bisa diatur dengan perintah move agar menghindari tembakan atau mencari posisi yang lebih menguntungkan untuk menyerang.

Pergerakan ini sangat penting untuk menghindari serangan musuh. Bot dapat bergerak ke samping atau berputar untuk menghindari tembakan yang datang dengan menganalisis arah dan kecepatan tembakan musuh. Algoritma untuk bergerak sering sekali melibatkan pengambilan keputusan berdasarkan keadaan medan pertempuran saat itu.

Untuk menjalankan bot dalam Robocode, yang perlu dilakukan pertama kali adalah menulis kode bot (implementasi algoritma bot). Setelah kode bot selesai ditulis, bot perlu dikompilasi menjadi file *executable*. Setelah itu, barulah bot dapat diuji dengan menjalankannya dalam simulator pertempuran Robocode. Simulator ini akan dapat menunjukkan bagaimana setiap bot bertarung dan bertindak dalam medan perang. Jika bot tidak berfungsi seperti yang seharusnya, kita dapat melakukan modifikasi pada code untuk melakukan perbaikan.

Algoritma *greedy* dalam implementasi bot adalah pendekatan dimana bot membuat keputusan berdasarkan pilihan terbaik yang tersedia pada saat itu, tanpa memikirkan konsekuensi jangka panjang. Dalam konteks Robocode, algoritma greedy dapat digunakan untuk membuat keputusan cepat dan efisien dalam pertempuran. Sebagai contoh, dalam mengimplementasikan algoritma greedy untuk pergerakan, bot bisa memilih untuk bergerak ke arah yang paling aman berdasarkan jarak terdekat dari musuh. Jika bot mendeteksi bahwa ada tembakan yang datang ke arahnya, bot akan menghindar dengan bergerak menjauhi tembakan tanpa memikirkan strategi yang lebih besar (tidak diam di tempat untuk menentukan langkah selanjutnya. Contoh lain adalah pada saat melakukan targeting, bot juga bisa menggunakan pendekatan greedy dengan menembak musuh yang berada dalam jarak dekat tanpa mempertimbangkan apakah itu langkah terbaik dalam waktu yang lama.

BAB III
APLIKASI STRATEGI GREEDY

3.1.Elemen Algoritma Greedy

Berikut merupakan proses pemetaan persoalan Robocode Tank Royale menjadi elemen-elemen algoritma *Greedy*, yang terdiri atas himpunan kandidat, himpunan solusi, fungsi solusi, fungsi seleksi, fungsi kelayakan, dan fungsi objektif.

Himpunan kandidat adalah himpunan yang berisi semua kandidat yang dapat dipilih pada setiap langkah, sedangkan himpunan solusi berisi kandidat yang telah dipilih. Fungsi solusi berfungsi untuk menentukan apakah himpunan kandidat yang dipilih sudah memberikan solusi, kemudian fungsi seleksi memilih kandidat berdasarkan strategi greedy tertentu. Strategi *greedy* ini bersifat heuristik. Kemudian, fungsi kelayakan berfungsi untuk memeriksa apakah kandidat yang terpilih dapat dimasukkan dan layak dimasukkan ke dalam himpunan solusi. Terakhir, fungsi objektif merupakan semacam target akhir dari algoritma *greedy*.

3.1.1 Himpunan Kandidat (C)

Himpunan kandidat pada Robocode ini merupakan semua pilihan yang bisa diambil bot pada setiap turn.

a. Maju
b. Mundur
c. Putar
d. Putar Radar
e. *Fire*
f. Berhenti/diam

3.1.2 Himpunan Solusi (S)

Himpunan solusi yang terdefinisi pada permainan Robocode ini merupakan aksi-aksi yang dapat digunakan untuk membentuk strategi bot selama pertandingan berlangsung. Aksi-aksi ini harus dapat dilakukan pada saat itu agar dapat masuk dalam himpunan solusi ini. Contohnya, jika bot pada saat itu tidak memiliki energi yang cukup untuk menembak, maka kandidat aksi menembak tidak termasuk dalam himpunan solusi.

3.1.3 Fungsi Solusi

Pada penyusunan algoritma *greedy* dalam Robocode ini, fungsi solusi adalah fungsi yang menentukan apakah suatu aksi atau kombinasi aksi ini merupakan sebuah opsi aksi yang valid pada saat itu.

a. Aksi dianggap valid jika tidak melanggar aturan game (tidak menabrak dinding secara sengaja, tidak melakukan aksi yang merugikan skor)
b. Serangan valid jika peluru dapat mengenai target dengan probabilitas tinggi

3.1.4 Fungsi Seleksi *(selection function)*

Fungsi seleksi dalam algoritma greedy di Robocode ini merupakan fungsi yang memilih aksi terbaik yang sebaiknya (akan) dilakukan pada saat itu berdasarkan pertimbangan heuristik tertentu. Dari himpunan solusi yang terdefinisi, fungsi seleksi kemudian memilih solusi (aksi) mana yang akan dieksekusi berdasarkan pendekatan heuristik yang diadopsi.

3.1.5 Fungsi Kelayakan *(feasibility function)*

Fungsi kelayakan pada Robocode ini merupakan fungsi yang memastikan aksi yang dipilih sebagai solusi untuk diterapkan tetap sesuai dengan kondisi permainan. Beberapa fungsi kelayakan pada program ini adalah sebagai berikut.

a. Menghindari dinding dan musuh saat bergerak
b. Menembak hanya jika energi cukup
c. Tidak terlalu banyak menggunakan peluru berat jika lawan masih banyak

3.1.6 Fungsi Objektif

Fungsi objektif pada Robocode ini dapat didefinisikan sebagai tujuan utama dari algoritma greedy dalam permainan, dalam hal ini adalah untuk memaksimalkan skor akhir dengan menyerang lawan seefektif mungkin serta bertahan selama mungkin untuk mendapatkan *survival score* yang tinggi. Selain itu, menargetkan musuh yang paling lemah untuk meningkatkan peluang menang juga dapat menjadi salah satu fungsi objektif pada permainan ini.

3.2.Eksplorasi Alternatif Solusi *Greedy*

3.2.1 Belle *(Greedy Center Guard)*

Belle menerapkan algoritma yang mengontrol jalur tengah dan siap menyerang balik. Pada algoritma ini, bot akan menuju ke tengah arena pada awal permainan. Setelah itu, jika ada yang menembak ke arahnya (bot ini mendapatkan hit), maka bot akan bergerak mengejar bot tersebut sambil menembak ke arahnya. Jika tidak, ia akan berputar dan melakukan scanning. Jika hanya terdapat bot di dekatnya dalam jarak tertentu, maka ia akan menyerangnya tanpa bergerak dari posisinya dengan kekuatan penuh. Urutan algoritma *greedy*-nya adalah:

1. Mulai bergerak ke tengah
2. Apakah tertembak?
3. Jika iya, putar ke arah tembakan dan bergerak mengejar dan menyerang yang menembak.
4. Jika tidak, putar scan, kemudian jika terdapat bot lain dalam jarak 200 piksel, tembak.

3.2.2 Ariel *(Greedy Center Guard)*

Ariel menerapkan algoritma yang selalu mengincar musuh terdekat dengan serangan kuat. Pada algoritma ini, bot akan scan, memilih bot yang terdekat dan bergerak zig-zag menujunya sambil menyerang dengan serangan terkuat hingga bot tersebut sudah mati. Gerakan zig-zag ini bertujuan untuk membuat bot lebih susah untuk dikejar atau ditargetkan oleh musuh. Setelah itu, bot akan scan kembali untuk menyerang target terdekat dengan langkah yang sama. Urutan algoritma *greedy*-nya adalah:

1. Bot melakukan scanning penuh
2. Pilih bot yang terdekat
3. Menyerang ke arah bot dengan kekuatan tertinggi sambil bergerak zig-zag
4. Apakah bot sudah mati?
5. Jika iya, balik ke langkah pertama
6. Jika tidak, serang bot lagi
7. Kembali ke langkah 4

3.2.3 Elsa (*Greedy Swift Shooter*)

Elsa menerapkan algoritma yang menyerang cepat lalu kabur sebelum diserang balik. Pada algoritma ini, bot akan bergerak di arena dalam pola persegi. Algoritma ini bertujuan untuk menjaga bot tetap bergerak dan menghindari serangan musuh secara aktif. Elsa memiliki mekanisme untuk menembak musuh yang terdeteksi saat bergerak dan menghindari benturan dengan bot lain atau tembok. Urutan algoritma *greedy*-nya adalah sebagai berikut:

1. Hitung jarak gerak maksimum (moveAmount) berdasarkan ukuran arena.
2. Putar bot agar sejajar dengan dinding terdekat.
3. Maju sejauh moveAmount.
4. Putar 90 derajat dan ulangi langkah 3 untuk bergerak dalam pola persegi.
5. Jika mendeteksi musuh, tembak dengan kekuatan 2.
6. Jika dalam mode "peek", lakukan rescan radar.
7. Jika bertabrakan dengan bot lain, mundur atau maju sesuai sudut tabrakan.
8. Jika terkena tembakan, belok 45 derajat dan maju 50 unit.
9. Jika menabrak dinding, mundur sedikit, lalu putar 90 derajat dan lanjutkan pola gerakan.

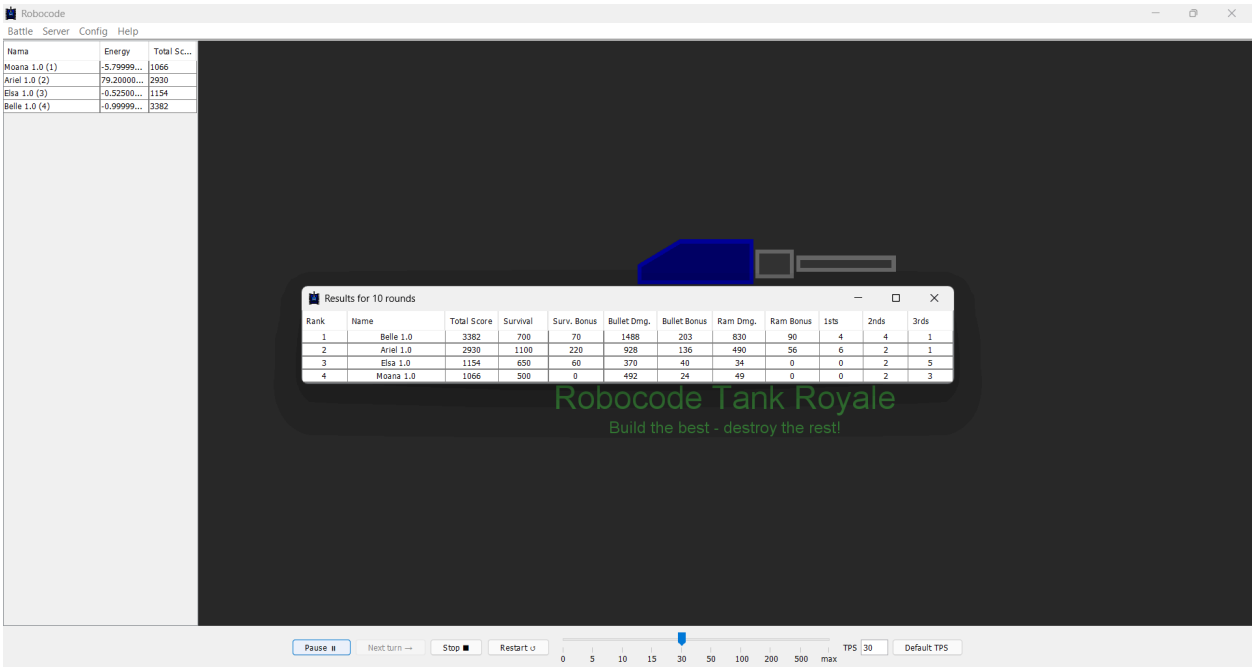
3.2.4 Moana (*Wave Hunter*)

Moana menerapkan algoritma yang bergerak secara sinusoidal, menjaga jarak terhadap musuh, dan mengeluarkan tembakan berdasarkan jarak musuh. Pada algoritma ini, bot Moana bergerak dalam pola gelombang sinus.

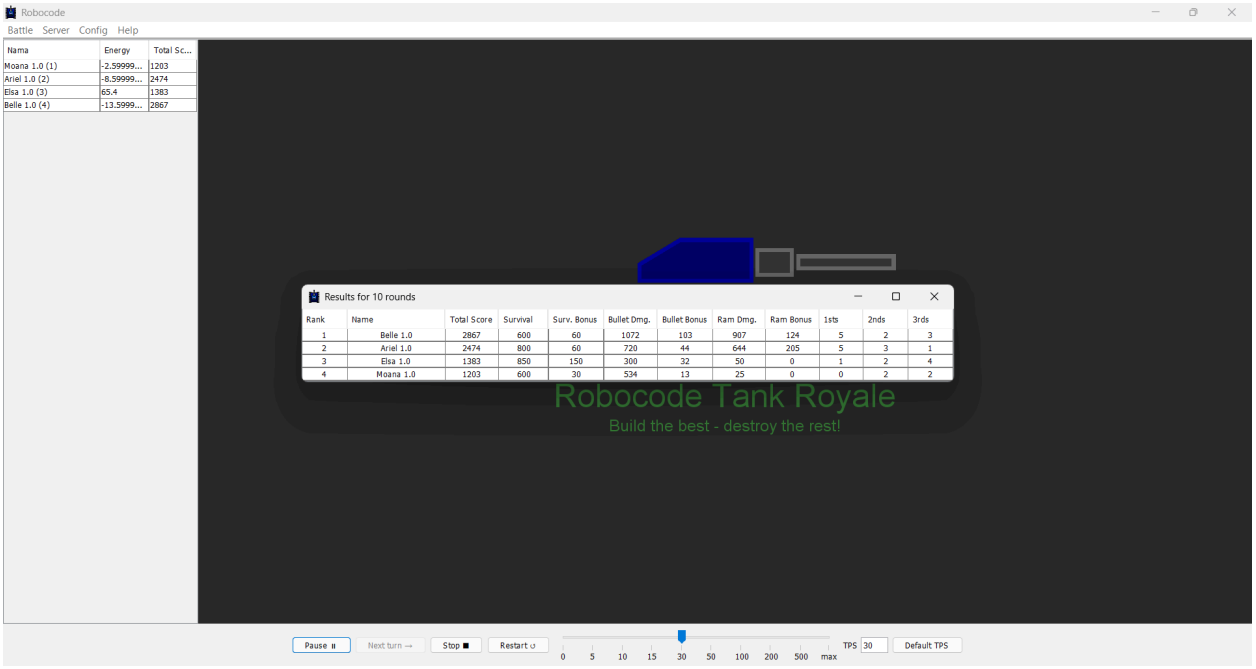
Pola ini memungkinkan Moana menghindari serangan sambil tetap mempertahankan mobilitas tinggi. Selain itu, Moana memiliki mekanisme untuk menembak musuh berdasarkan jarak dan menghindari benturan dengan tembok serta serangan lawan. Urutan algoritma *greedy*-nya adalah sebagai berikut:

1. Bergerak maju dengan pola gelombang sinus, memutar sedikit berdasarkan fungsi sinus.
2. Bot melakukan scanning
3. Jika bot mendeteksi musuh, maka tembak musuh
4. Jika musuh terlalu dekat, bot mundur. Jika terlalu jauh, bot maju
5. Jika terkena tembakan, bot menghindar
6. Jika menabrak dinding, bot akan mundur, belok 90 derajat, dan mengikuti pola dari nomor 1

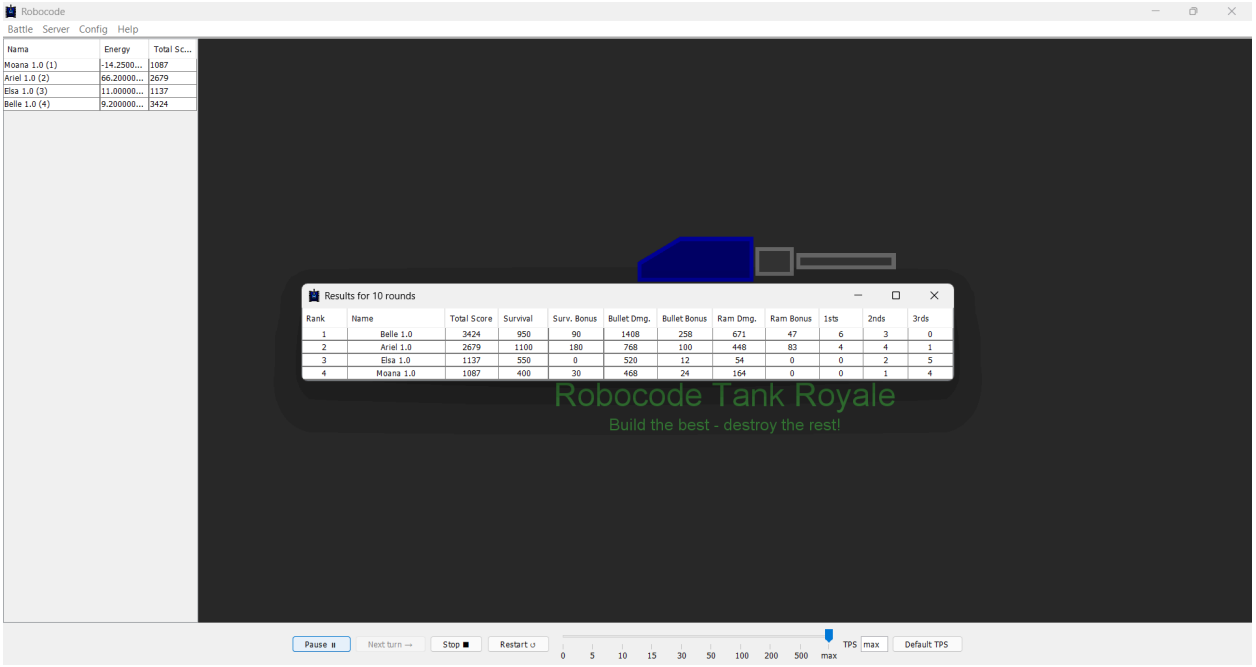
3.3 Analisis Efisiensi dan Efektivitas Solusi *Greedy*



Gambar 2 Uji Pertarungan 4 Bot yang Telah Dibuat (Belle, Ariel, Elsa, dan Moana) yang pertama



Gambar 3 Uji Pertarungan 4 Bot yang Telah Dibuat (Belle, Ariel, Elsa, dan Moana) yang kedua



Gambar 4 Uji Pertarungan 4 Bot yang Telah Dibuat (Belle, Ariel, Elsa, dan Moana) yang ketiga

Pada algoritma pertama, bot Belle, menyerang dengan sangat baik sebagai bentuk *defense* dan juga *offense*, karena ia akan menunggu untuk terkena serangan atau tabrakan terlebih dahulu untuk menyerang, tetapi juga akan menyerang dengan *full* ketika terserang. Tetapi, berbeda dari yang direncanakan, ternyata sistem RoboCode tidak dapat mengidentifikasi penyerang dari *bullet* yang terkena Belle, sehingga pada implementasinya, kami membuat Belle menyerang dan mengejar bot yang paling dekat dengannya pada saat itu, dengan jarak <= 100 piksel, mengasumsikan bahwa yang paling dekatlah yang menyerang Belle ini. Perubahan ini

juga sebenarnya bertujuan untuk membuat bot Belle bergerak ketika terserang agar tidak menjadi target yang gampang bagi yang lainnya.

Bot Ariel setelah diuji merupakan yang paling efisien kedua. Hal ini karena bot Ariel juga merupakan sebuah bot yang agresif, yang selalu menyerang musuh terdekat dengan kekuatan penuh. Ariel juga mengimplementasikan gerakan zig-zag ketika sedang bergerak, serta ia juga menyalakan radar setiap saat untuk mendeteksi musuh. Namun, Ariel kalah dengan Belle karena ia memanfaatkan *turn*-nya untuk bergerak zig-zag, yang menyebabkannya untuk bergerak lebih lambat daripada Belle dalam mengejar musuh. Walaupun begitu, gerakan zig-zag ini juga memiliki keuntungannya sendiri, yaitu membuat Ariel lebih sulit untuk ditarget musuh. Dalam implementasi bot Ariel ini, perbandingan antara jarak bot satu dengan lainnya menghabiskan banyak waktu karena bot yang berada di arena juga banyak. Hal ini menyebabkan penggunaan *turn* serta waktu bertarung tidak efisien. Oleh karena itu, kami memutuskan untuk membuat parameter perbandingan jarak syarat untuk menyerang menjadi minimal 100 piksel dari tempat Ariel.

Bot Elsa merupakan bot yang menduduki posisi ketiga dalam kategori paling efisien. Hal tersebut dapat terjadi karena bot Elsa tidak seagresif bot Belle dan bot Ariel yang lebih fokus menyerang musuh secara aktif. Bot Elsa mengandalkan strategi pergerakan pola persegi, bot ini akan berbelok setiap kali menabrak tembok. Dengan pola ini, bot Elsa dapat bergerak secara terstruktur di sekitar arena tanpa berpindah secara acak. Meskipun lebih pasif dibandingkan bot lain, strategi ini membuat bot Elsa lebih stabil dan bertahan lebih lama di arena karena tidak terlalu sering mengambil risiko dalam pertarungan. Salah satu keunggulan utama Elsa adalah efisiensi energinya. Berbeda dengan Ariel dan Belle yang terus-menerus menyerang, Elsa hanya akan menembak ketika menemukan musuh, sehingga penggunaannya lebih hemat energi. Selain itu, pola pergerakan yang sederhana membuat bot ini lebih stabil dan tidak mudah terkena serangan musuh yang bergerak agresif. Namun, meskipun memiliki keunggulan dalam stabilitas dan efisiensi, Elsa juga memiliki beberapa kelemahan. Karena tidak aktif mencari musuh, bot ini bisa kehilangan kesempatan menyerang lawan yang memiliki energi rendah. Selain itu, pola persegi yang digunakan dalam pergerakannya dapat diprediksi oleh lawan yang cerdas, yang bisa memanfaatkan titik belokan Elsa untuk menyerangnya.

Bot Moana merupakan bot yang menduduki posisi keempat (terakhir) dalam kategori paling efisien. Hal tersebut terjadi karena mengandalkan pergerakan sinusoidal yang meningkatkan peluangnya untuk menghindari tembakan musuh. Namun, bot Moana ini akan mundur dan selalu menjaga jarak minimal 100px dari bot lawan agar tidak ditabrak hingga mati. Selain itu, bot ini akan menembak bot musuh dengan pertimbangan jarak. Pertimbangan jarak yang dimaksud adalah, jika bot musuh berjarak kurang dari 300px, bot Moana akan menembak dengan kekuatan berdaya 3, jika bot berjarak di antara 300px sampai 600px, bot Moana akan menembak dengan kekuatan berdaya 2, dan jika bot musuh berjarak lebih jauh lagi, bot Moana akan menembak dengan kekuatan berdaya 1. Penerapan ini memungkinkan bot Moana untuk menyeimbangkan antara efisiensi energi dan

efektivitas serangan. Bot Moana akan mengalami kekalahan telak apabila bertemu bot-bot yang sangat agresif seperti Belle dan Ariel.

3.4 Pemilihan Strategi Greedy dan Alasan

Strategi greedy yang dipilih untuk menjadi bot utama adalah **bot Belle**. Hal ini karena bot Belle menunjukkan hasil pertarungan yang paling optimal di antara keempat bot yang telah kami rancang. Bot ini menyatukan antara taktik *offense* dan *defense* yang baik, serta menyerang bot lain secara agresif. Belle ini juga sangat efektif dalam meraih tujuan utama dari bot, yaitu mendapatkan skor terbanyak, baik itu dari *ramming* atas agresivitasnya ataupun dari skor lainnya. Oleh karena itu, bot Belle merupakan bot paling optimal untuk dijadikan bot utama.

BAB IV
IMPLEMENTASI DAN PENGUJIAN

4.1. Implementasi 4 Alternatif Solusi

4.1.1 *Pseudocode* Bot Belle

```
START
    INISIALISASI Warna bot (BodyColor = Kuning,
TurretColor = Hitam, RadarColor = Hitam,
BulletColor = Kuning, ScanColor = Kuning)

    GERAKKAN bot ke posisi tengah arena sebagai
center guard

    WHILE Bot masih berjalan (IsRunning):
        RESET flag isHit dan isScanned (isHit = 0,
isScanned = 0)

        IF bot belum terkena serangan:
            GERAKKAN bot ke posisi tengah arena
            Lanjutkan scanning radar untuk mencari
bot lain
        ELSE IF bot terkena serangan:
            BERPINDAH ke mode counterattack dan
terus mencari bot lain untuk dibalas tembak

END WHILE

FUNCTION MoveToCenter()
    // Tentukan posisi tengah arena sebagai target
    SET targetX = ArenaWidth / 2
    SET targetY = ArenaHeight / 2

    // Putar bot untuk menghadap target (tengah
arena)
    PUTAR bot ke target menggunakan
TurnToFaceTarget

    // Bergerak maju menuju target (tengah arena)
    MAJU menuju target (tengah arena)
END FUNCTION

FUNCTION TurnToFaceTarget(targetX, targetY)
    // Hitung bearing (sudut) menuju target
    HITUNG bearing sebagai sudut ke target
menggunakan BearingTo
```

```
    // Tentukan arah putaran berdasarkan nilai
bearing
    SET turnDirection sesuai dengan nilai bearing
(positif atau negatif)

    // Putar bot ke arah target
    PUTAR bot ke arah target menggunakan
SetTurnLeft(bearing)
END FUNCTION

EVENT OnScannedBot(ScannedBotEvent e)
    // Tandai bahwa bot telah discan
    SET isScanned = 1

    // Hitung perbedaan sudut antara arah bot dan
bot yang discan
    HITUNG angleDifference antara arah bot dan bot
yang discan
    HITUNG turningPoint untuk putaran radar

    // Hitung toleransi untuk memastikan scan radar
lebih akurat
    HITUNG toleransi untuk pengaturan arah putaran
radar

    // Putar radar untuk menyesuaikan arah bot yang
discan
    PUTAR radar ke titik yang dihitung

    // Jika bot belum terkena serangan dan jarak ke
bot yang discan <= 200
    IF bot belum terkena serangan DAN jarak ke bot
<= 200:
        PUTAR bot untuk menghadap bot yang discan
        TEMBAK bot dengan daya penuh (firepower =
3)
    ELSE:
        // Jika bot sudah terkena serangan atau
jarak lebih dekat
        PUTAR bot ke arah bot yang discan
        MAJU menuju bot yang discan

        // Jika jarak cukup dekat (<= 100 unit),
tembak bot
        IF jarak ke bot <= 100:
            TEMBAK bot dengan daya penuh (firepower
= 3)

        // Jika bot musuh dihancurkan (energi <=
0), berhenti mengejar dan kembali ke mode scanning
```

```
        IF e.Energy <= 0:
            RESET isHit (isHit = 0) // Hentikan
pengejaran setelah bot dihancurkan
            KEMBALI ke mode scanning
    END IF
END EVENT

EVENT OnHitByBullet(HitByBulletEvent e)
    // Tandai bahwa bot terkena peluru
    SET isHit = 1
END EVENT

EVENT OnHitBot(HitBotEvent e)
    // Tandai bahwa bot bertabrakan dengan bot lain
    SET isHit = 1
END EVENT
```

4.1.2 *Pseudocode* Bot Ariel

```
START
    INISIALISASI Warna bot (BodyColor = Hijau,
TurretColor = Hitam, RadarColor = Hijau,
BulletColor = Hijau, ScanColor = Hijau)

    WHILE Bot masih berjalan (IsRunning):
        Panggil fungsi ZigzagMovement() untuk
melakukan gerakan zigzag
        Putar radar ke kanan untuk terus mencari
bot lain (TurnRadarRight dengan nilai positif tak
terbatas)

END WHILE

FUNCTION ZigzagMovement()
    // Putar bot ke kanan sebesar 30 derajat *
turnDirection untuk membuat pola zigzag
    SetTurnRight(30 * turnDirection)

    // Tunggu sampai rotasi selesai (menggunakan
WaitFor dan memeriksa TurnRemaining)
    Tunggu sampai putaran selesai (TurnRemaining ==
0)

    // Bergerak maju 100 unit setelah putaran
selesai
    SetForward(100)

    // Ubah arah putar untuk langkah zigzag
```

```
berikutnya
    turnDirection *= -1
END FUNCTION

EVENT OnScannedBot(ScannedBotEvent e)
    // Tandai bahwa bot telah discan
    isScanned = 1

    // Hitung perbedaan sudut (angle) antara arah
bot dan bot yang discan
    angleDifference = Direction + BearingTo(e.X,
e.Y)
    turningPoint =
NormalizeRelativeAngle(angleDifference -
RadarDirection)

    // Hitung toleransi untuk perbedaan sudut radar
agar lebih akurat
    tolerance = Min(Atan(36.0 / DistanceTo(e.X,
e.Y)), 45)

    // Sesuaikan titik putar radar dengan toleransi
yang dihitung
    turningPoint += (turningPoint < 0 ? -tolerance
: tolerance)
    SetTurnRadarLeft(turningPoint)

    // Putar bot untuk menghadap target dan maju
menuju bot yang discan
    TurnToFaceTarget(e.X, e.Y)
    SetForward(DistanceTo(e.X, e.Y))

    // Jika jarak ke bot cukup dekat (<= 100) dan
bot masih hidup (energi > 0), tembak
    IF DistanceTo(e.X, e.Y) <= 100 AND e.Energy >
0:
        SetFire(3)
    END IF
END EVENT

FUNCTION TurnToFaceTarget(double x, double y)
    // Hitung bearing (sudut) ke target
    bearing = BearingTo(x, y)

    // Putar bot ke arah target
    SetTurnLeft(bearing)

    // Tunggu sampai rotasi selesai (menggunakan
WaitFor dan memeriksa TurnRemaining)
    Tunggu sampai putaran selesai (TurnRemaining ==
```

```
0)
END FUNCTION
```

4.1.3 *Pseudocode* Bot Elsa

```
START
    // Inisialisasi warna bot
    SET BodyColor = Biru
    SET TurretColor = Putih
    SET RadarColor = Biru Muda
    SET BulletColor = Cyan
    SET ScanColor = Biru Muda

    // Tentukan jarak pergerakan bot
    SET moveAmount = MAX(ArenaWidth, ArenaHeight) -
50
    SET peek = FALSE

    // Sesuaikan arah bot agar sejajar dengan
dinding
    MAJU menuju arah(Direction MOD 90)

    // Gerak maju ke tepi arena
    MAJU menuju tepi arena

    // Belok kanan untuk mulai patroli
    PUTAR bot ke arah kanan
    SET peek = TRUE

    // Loop utama selama bot masih berjalan
    WHILE Bot masih berjalan (IsRunning):
        SET peek = TRUE
        MOVE_FORWARD(moveAmount - 50)
        SET peek = FALSE
        TURN_RIGHT(90)
    END WHILE
END

EVENT OnScannedBot(ScannedBotEvent e)
    // Jika bot musuh terdeteksi, tembak dengan
kekuatan sedang
    TEMBAK bot lawan dengan daya 2

    // Jika sedang mengintip, lakukan pemindaian
ulang
    IF peek = TRUE:
        CALL Rescan()
    END IF
END EVENT
```

```
EVENT OnHitBot(HitBotEvent e)
    // Hitung sudut relatif ke bot yang ditabrak
    SET bearing = BearingTo(e.X, e.Y)

    // Jika bot musuh berada di depan, mundur
    IF bearing > -90 AND bearing < 90:
        MUNDUR 100 unit
    ELSE:
        // Jika bot musuh berada di belakang, maju
        MAJU 100 unit
    END IF
END EVENT

EVENT OnHitByBullet(HitByBulletEvent e)
    // Menghindari tembakan dengan berputar dan
bergerak maju
    BELOK KANAN
    MAJU 50 unit
END EVENT

EVENT OnHitWall(HitWallEvent e)
    // Mundur sedikit lalu berputar untuk
menghindari dinding
    MUNDUR 20 unit
    BELOK KANAN
END EVENT
```

4.1.4 *Pseudocode* Bot Moana

```
START
    // Inisialisasi warna bot
    SET BodyColor = Teal
    SET TurretColor = Deep Orange
    SET RadarColor = Amber
    SET BulletColor = Red
    SET ScanColor = Cyan

    // Loop utama selama bot masih berjalan
    WHILE Bot masih berjalan (IsRunning):
        CALL MoveInSineWave()
        PUTAR radar
    END WHILE
END

FUNCTION MoveInSineWave()
    MAJU 20 unit
    BELOK KANAN(10 * SIN(wavePhase))
```

```
    INCREMENT wavePhase BY 0.5
END FUNCTION

EVENT OnHitByBullet(HitByBulletEvent e)
    BELOK KANAN
    MAJU 150 unit
END EVENT

EVENT OnHitWall(HitWallEvent e)
    MUNDUR 50 unit
    BELOK KANAN
END EVENT

EVENT OnScannedBot(ScannedBotEvent e)
    // Tandai bahwa bot telah discan
    SET isScanned = 1

    // Hitung perbedaan sudut antara arah bot dan
bot yang discan
    HITUNG angleDifference antara arah bot dan bot
yang discan
    HITUNG turningPoint untuk putaran radar

    // Hitung toleransi untuk memastikan scan radar
lebih akurat
    HITUNG toleransi untuk pengaturan arah putaran
radar

    // Putar radar untuk menyesuaikan arah bot yang
discan
    PUTAR radar ke titik yang dihitung

    // Tentukan kekuatan tembakan berdasarkan jarak
    IF distance < 300:
        TEMBAK bot musuh dengan daya 3
    ELSE IF distance < 600:
        TEMBAK bot musuh dengan daya 2
    ELSE:
        TEMBAK bot musuh dengan daya 1
    END IF

    // Jika musuh terlalu dekat, mundur
    IF distance < 200:
        MUNDUR 100 unit
    // Jika musuh terlalu jauh, maju mendekat
    ELSE IF distance > 600:
        MAJU 100 unit
    END IF
END EVENT
```

```
FUNCTION TurnToFaceTarget(targetX, targetY)
    // Hitung bearing (sudut) menuju target
    HITUNG bearing sebagai sudut ke target
menggunakan BearingTo

    // Tentukan arah putaran berdasarkan nilai
bearing
    SET turnDirection sesuai dengan nilai bearing
(positif atau negatif)

    // Putar bot ke arah target
    PUTAR bot ke arah target menggunakan
SetTurnLeft(bearing)
END FUNCTION
```

4.2. Penjelasan Struktur Data, Fungsi, dan Prosedur yang digunakan pada Bot dengan Solusi Greedy yang Dipilih

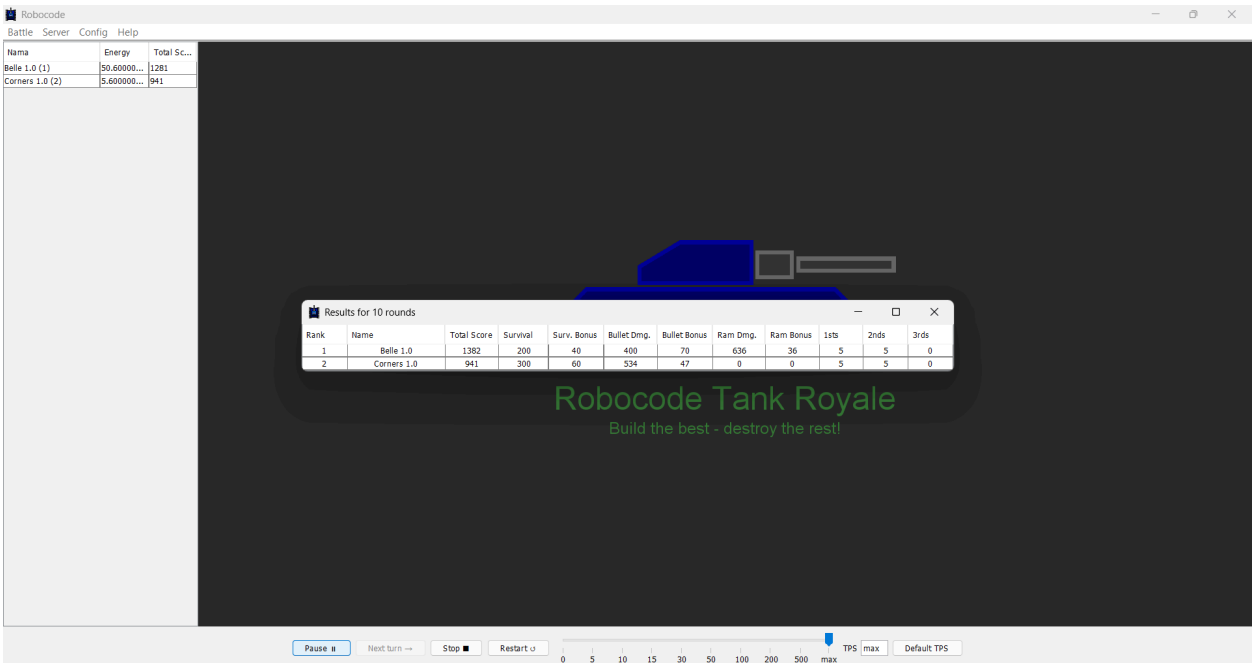
Tabel 1 Tabel Penjelasan Struktur Data, Fungsi, dan Prosedur pada Bot Belle

Komponen	**Tipe**	**Deskripsi**
turnDirection	Integer	Mengontrol arah putaran bot. Nilai positif memutar bot ke kanan, negatif memutar ke kiri.
isScanned	Integer	Menandakan apakah ada bot yang tercan. Nilai 1 berarti telah ada, 0 berarti belum.
isHit	Integer	Menandakan apakah bot telah terkena serangan atau bertabrakan dengan bot lain. Nilai 1 berarti telah terkena serangan, nilai 0 berarti tidak.
MoveToCenter	Prosedur	Memindahkan bot ke posisi tengah arena.
TurnToFaceTarget	Prosedur	Memutar bot untuk menghadap target berdasarkan koordinat (x,y) parameter dari prosedur.
Run	Prosedur	Prosedur utama yang

		dijalankan bot untuk bergerak ke tengah arena, memindai, dan mengejar musuh.
OnScannedBot	Event Handler	Menangani bot yang discan. Jika bot dalam jarak tembak, bot akan menembak dan/atau bergerak ke target.
OnHitByBullet	Event Handler	Menangani jika bot terkena peluru, mengubah status menjadi terkena serangan (isHit = 1).
OnHitBot	Event Handler	Menangani jika bot bertabrakan dengan bot lain, mengubah status menjadi terkena tabrakan (isHit =1).

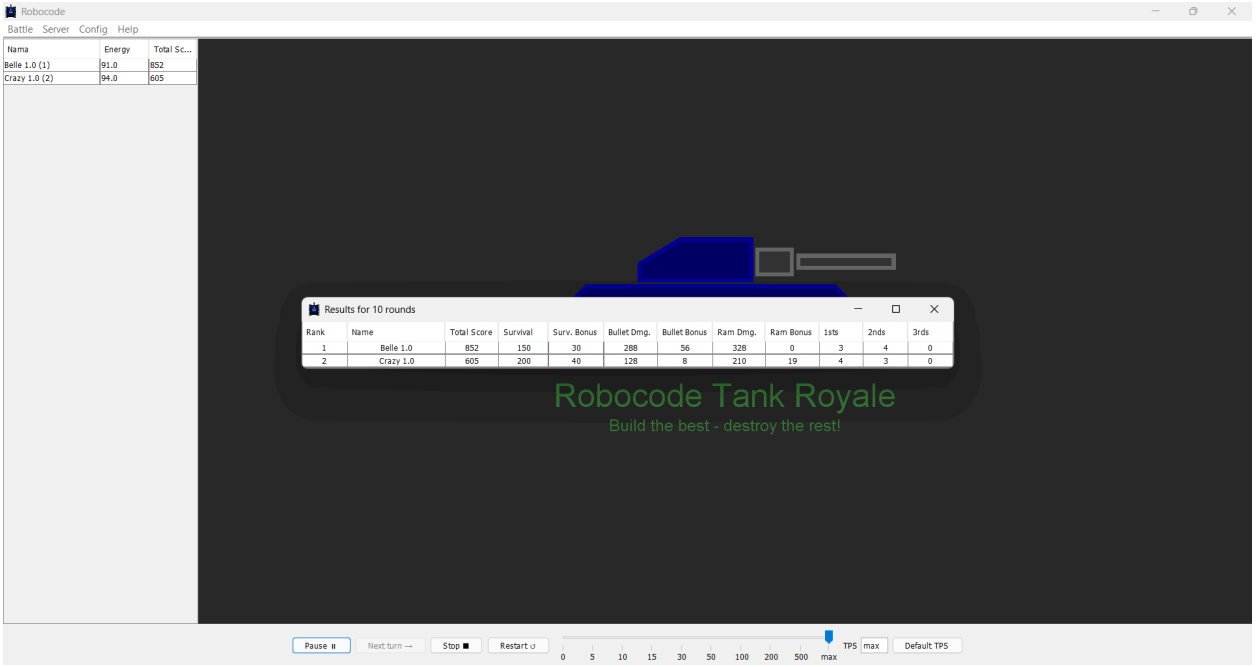
4.3. Pengujian Bot dengan 3 Bot Asisten

4.3.1 Belle melawan Corners



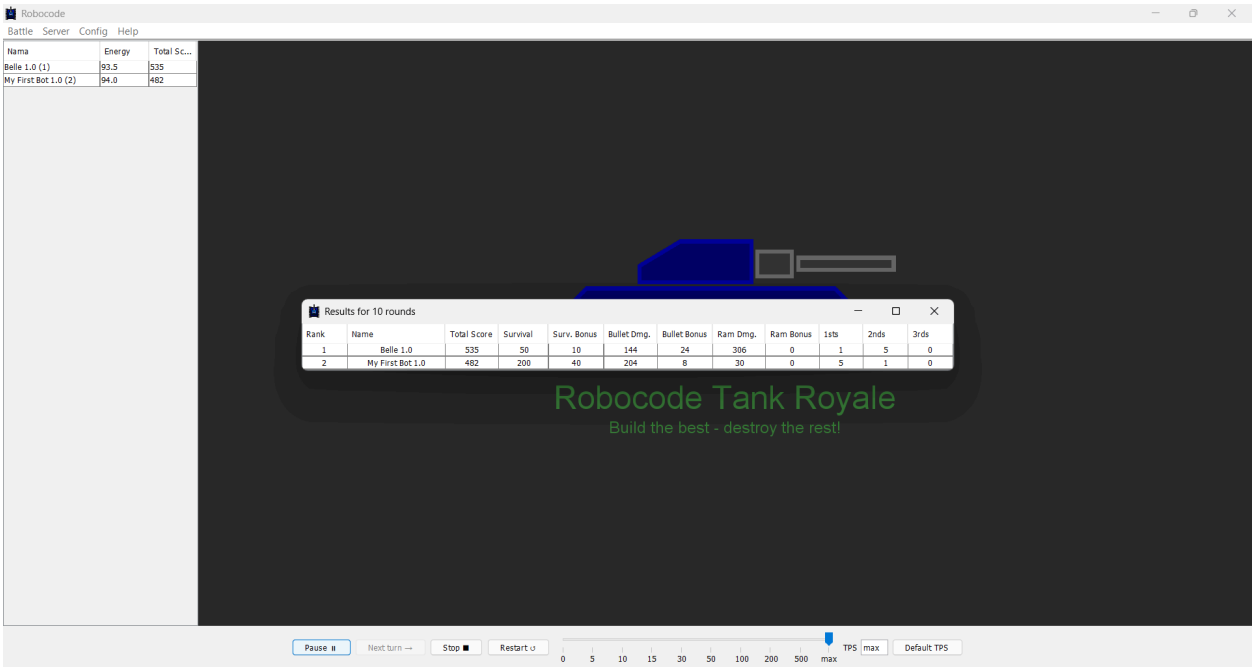
Gambar 5 Uji Pertarungan Bot Belle dengan Bot Asisten (Corners)

4.3.2 Belle melawan Crazy



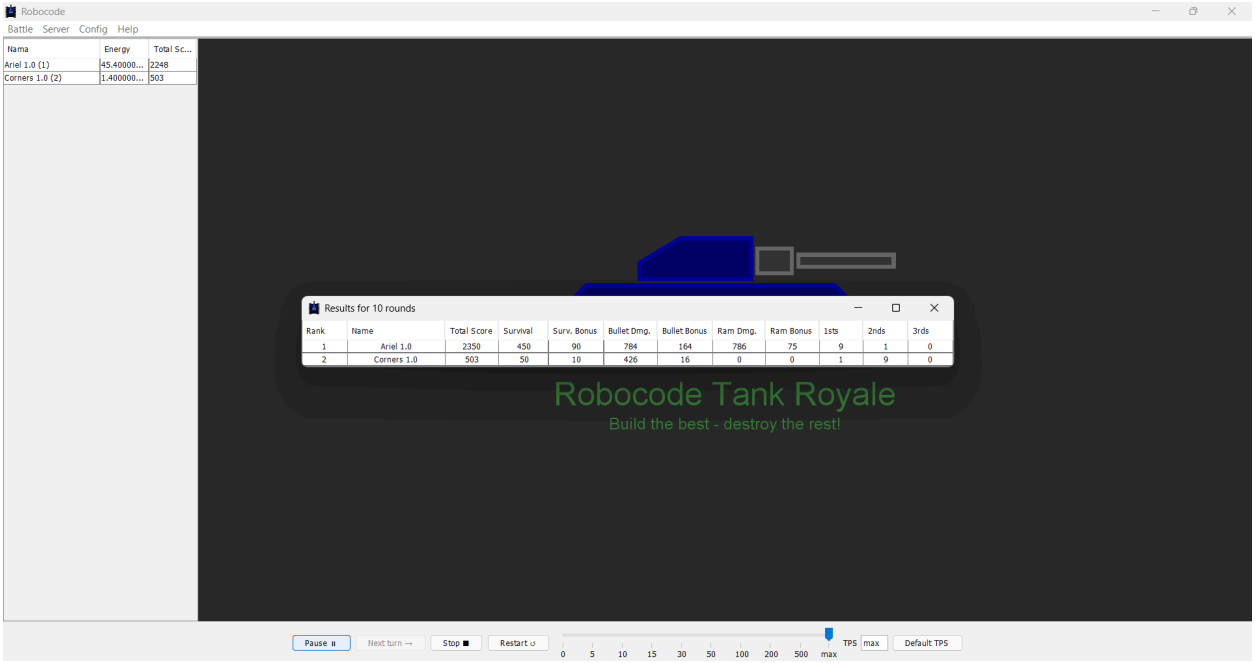
Gambar 6 Uji Pertarungan Bot Belle dengan Bot Asisten (Crazy)

4.3.3 Belle melawan MyFirstBot



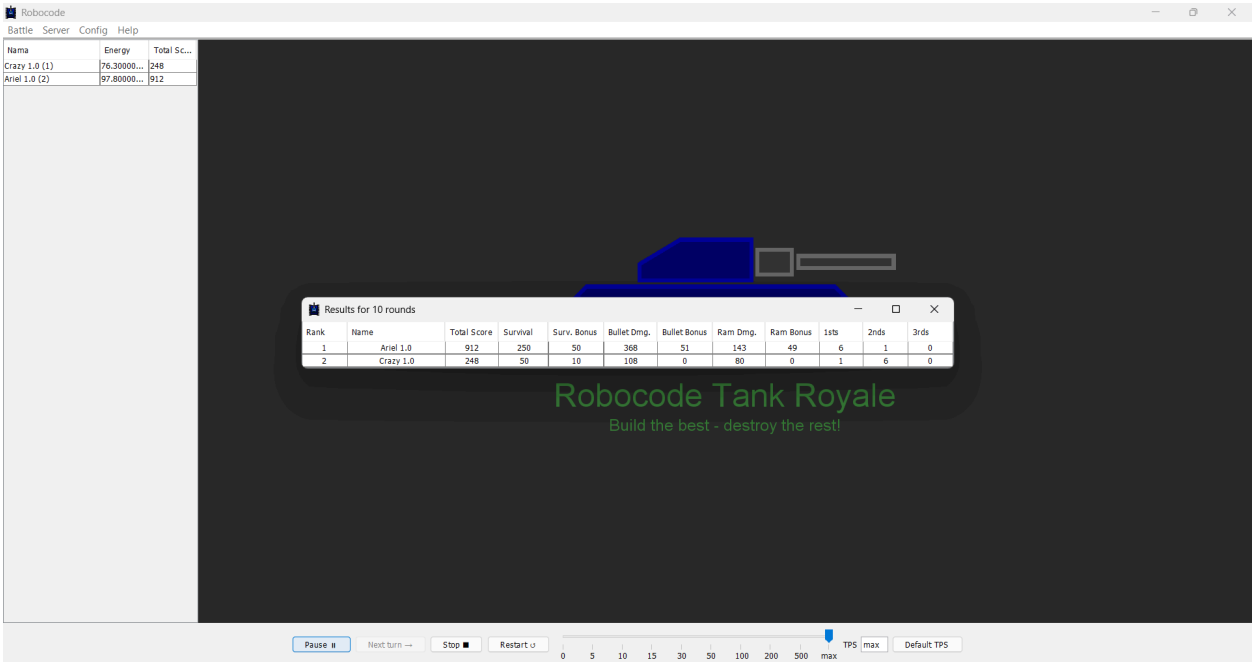
Gambar 7 Uji Pertarungan Bot Belle dengan Bot Asisten (MyFirstBot)

4.3.4 Ariel melawan Corners



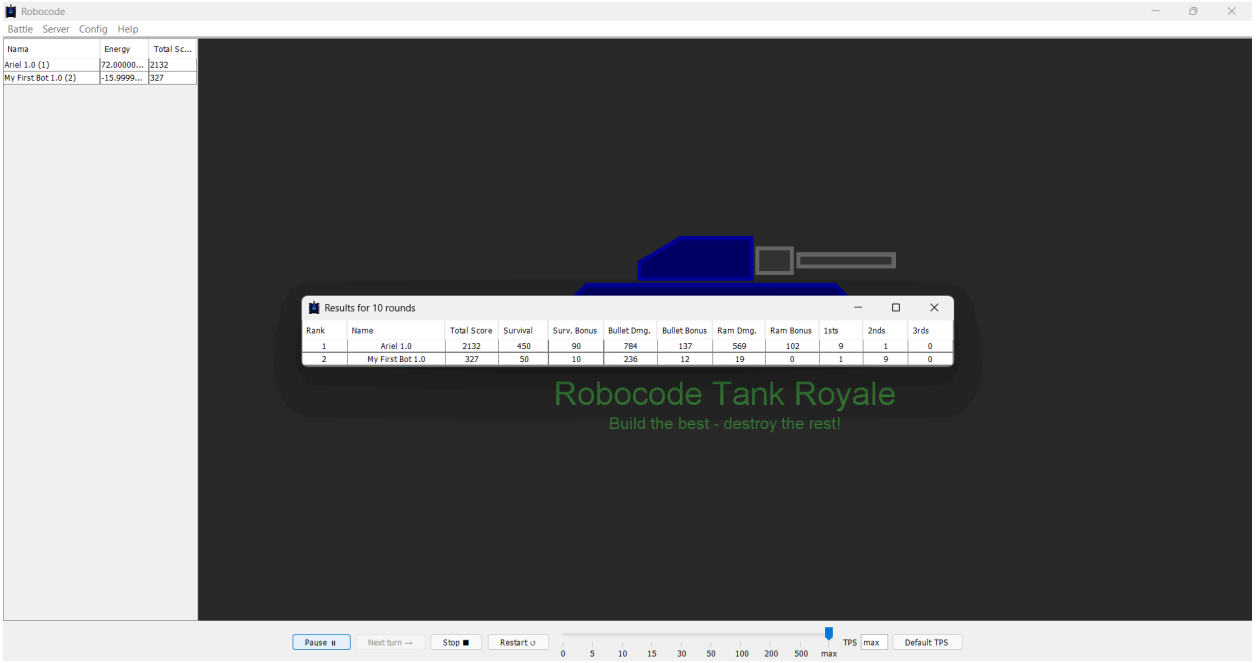
Gambar 8 Uji Pertarungan Bot Ariel dengan Bot Asisten (Corners)

4.3.5 Ariel melawan Crazy



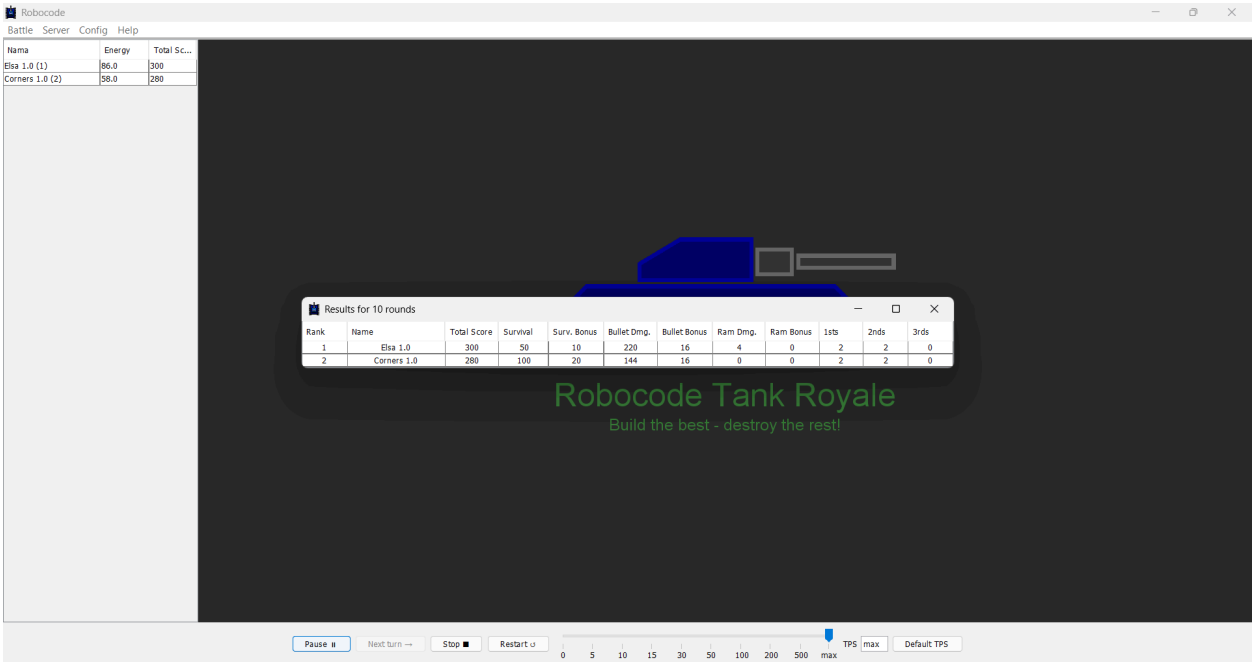
Gambar 9 Uji Pertarungan Bot Ariel dengan Bot Asisten (Crazy)

4.3.6 Ariel melawan MyFirstBot



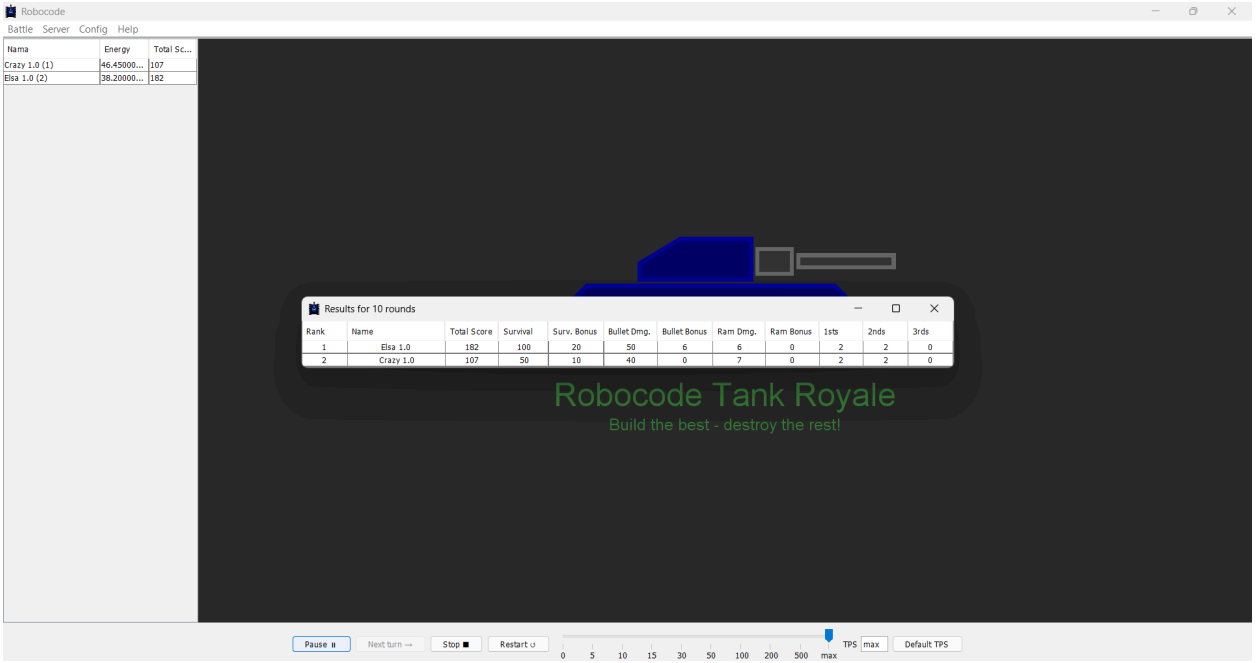
Gambar 10 Uji Pertarungan Bot Ariel dengan Bot Asisten (MyFirstBot)

4.3. Elsa melawan Corners



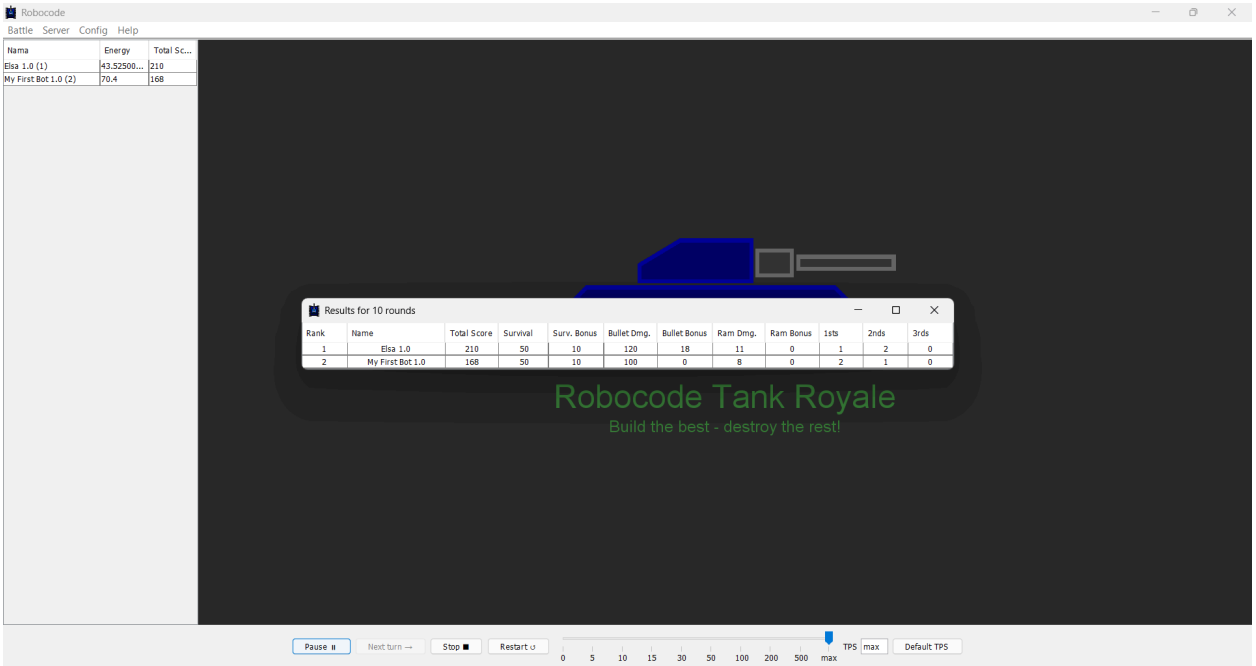
Gambar 11 Uji Pertarungan Bot Elsa dengan Bot Asisten (Corners)

4.3.5 Elsa melawan Crazy



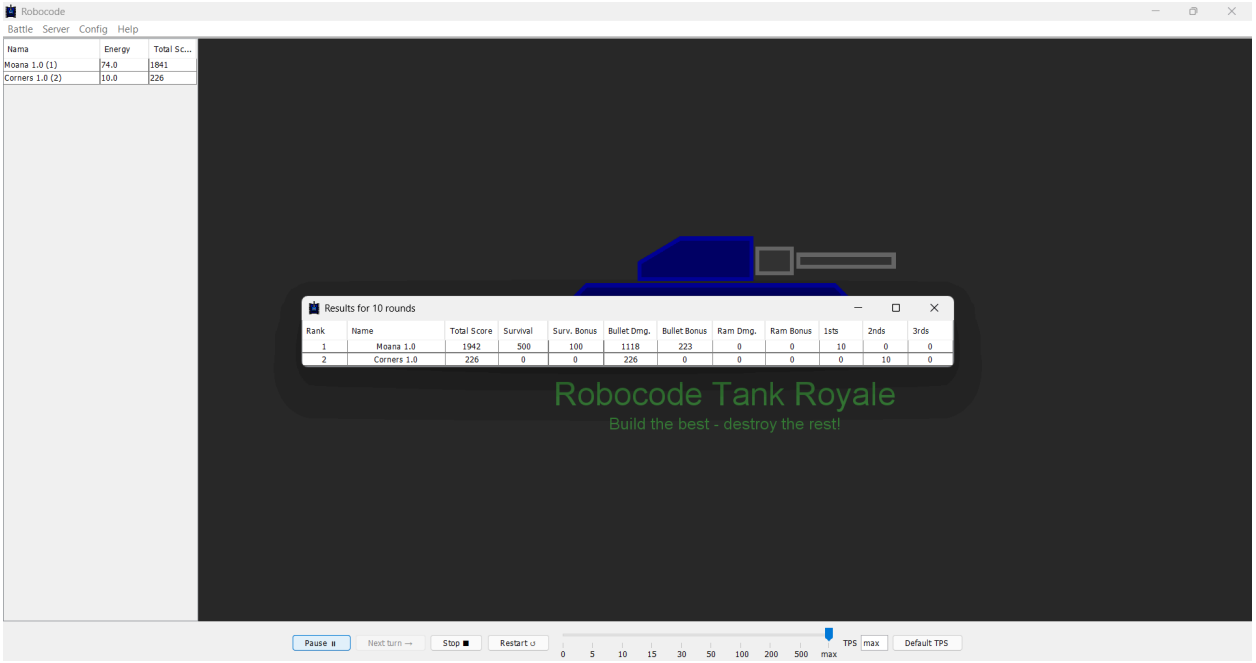
Gambar 12 Uji Pertarungan Bot Elsa dengan Bot Asisten (Crazy)

4.3.6 Elsa melawan MyFirstBot



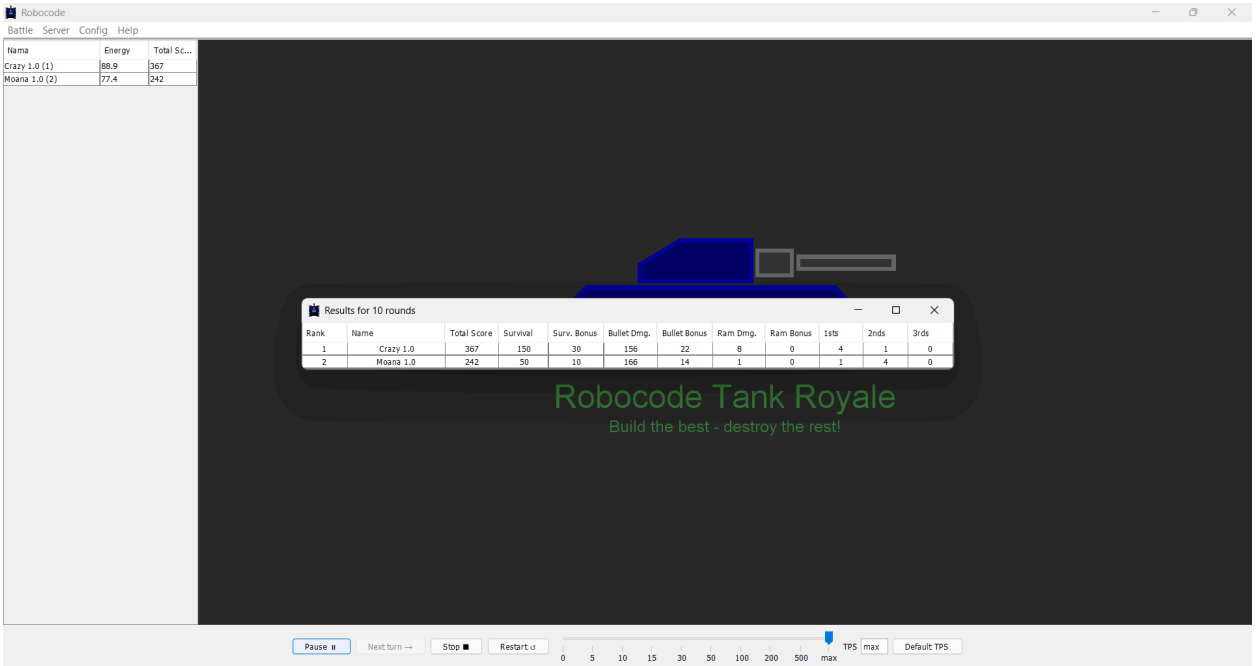
Gambar 13 Uji Pertarungan Bot Elsa dengan Bot Asisten (MyFirstBot)

4.3.4 Moana melawan Corners



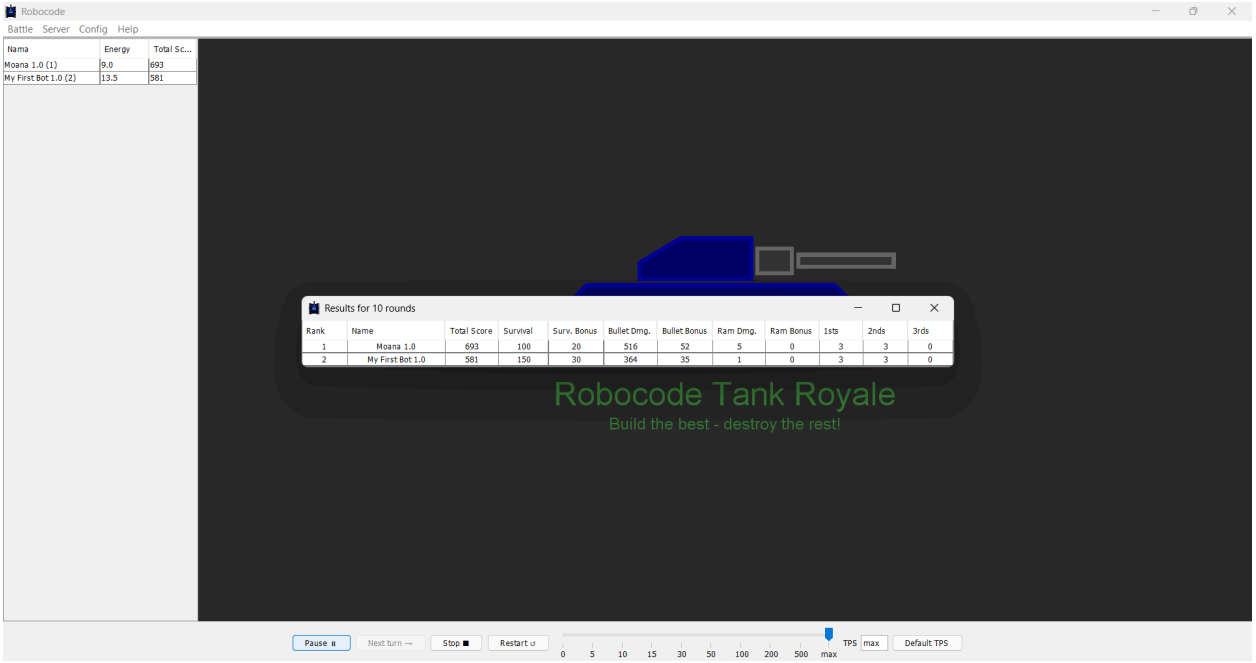
Gambar 14 Uji Pertarungan Bot Moana dengan Bot Asisten (Corners)

4.3.5 Moana melawan Crazy



Gambar 15 Uji Pertarungan Bot Moana dengan Bot Asisten (Crazy)

4.3.6 Moana melawan MyFirstBot



Gambar 16 Uji Pertarungan Bot Moana dengan Bot Asisten (MyFirstBot)

4.4.Analisis Hasil dari Pengujian

Dari hasil pengujian bot utama ini, dapat dilihat bahwa pada setiap pertandingan Belle memenangkan pertandingan dan mendapatkan hasil yang optimal karena algoritmanya yang menekan musuh jika terkena hit. Ia akan balas serangan dengan serangan full dan agresif. Ketika sedang tidak terkena serangan juga Belle hanya akan menyerang musuh di sekitarnya yang berada di jarak 200px darinya, jadi tidak menyia-nyiakan energi. Namun, tentunya hal ini juga bergantung pada placement awal dari bot karena Belle akan berusaha untuk menuju ke tengah terlebih dahulu jika belum kena *hit*.

BAB V
KESIMPULAN DAN SARAN

5.1 Kesimpulan

Dari hasil implementasi dan pengujian bot pada permainan Robocode Tank Royale dengan menggunakan algoritma *greedy*, bot Belle menunjukkan hasil yang paling optimal. Belle berhasil menggabungkan strategi *defense* dan *offense* dengan baik, yaitu dengan mengontrol posisi tengah arena dan menyerang balik dengan kekuatan penuh saat diserang. Meskipun demikian, bot Ariel dengan gerakan zig-zag juga menunjukkan performa yang cukup baik, namun sedikit lebih lambat karena memanfaatkan waktu untuk bergerak. Bot Elsa dan Moana menunjukkan strategi yang lebih menghindar dan mengandalkan pola gerakan, namun tidak seagresif Belle maupun Ariel dalam hal mendapatkan skor.

Pada pengujian, Belle berhasil mendapatkan skor tertinggi dan memenangkan pertandingan, membuktikan bahwa algoritma greedy yang dipilih cukup efektif dalam mendapatkan hasil yang optimal.

5.2 Saran

Untuk pengembangan lebih lanjut, saran yang dapat diberikan adalah:

1. **Optimasi Gerakan Bot**, memperbaiki pola pergerakan bot agar lebih dinamis dalam menghindari musuh dan tembakan dapat meningkatkan efisiensi pengumpulan skor.
2. **Penggunaan Energi**, mengatur penggunaan energi dengan lebih bijaksana agar lebih mengoptimalkan performa bot dalam pertempuran jangka panjang.
3. **Eksplorasi Algoritma Lain**, mengembangkan solusi yang lebih cerdas dengan algoritma pencarian yang lebih kompleks untuk meningkatkan efektivitas bot lebih lanjut.

DAFTAR PUSTAKA

Munir, R. (2025). *Tugas Besar 1 – IF2211 Strategi Algoritma*. Institut Teknologi Bandung. Diakses dari https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2024-2025/Tubes1-Stima-2025.pdf

Robocode Development Team. (n.d.). *Introduction to Robocode Tank Royale*. Diakses dari https://robocode-dev.github.io/tank-royale/articles/intro.html#the-name-tank-royale

Asisten Lab IRK (2025). *Spesifikasi Tugas Besar 1 Stima 2024/2025*. Diakses dari https://docs.google.com/document/d/14MCaRiFGiA6Ez5W8-OLxZ9enXyENcep7AzSH6sUHKM8/edit?tab=t.0

LAMPIRAN

Tautan *repository* Github : https://github.com/shanlie20/Tubes1_biskitop
Tautan video : https://youtu.be/2Uyx_Zcb3kw?si=-sf0-nBB5ifbTE6Y

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.	✔️	