LAPORAN TUGAS BESAR 1

IF2211 STRATEGI ALGORITMA

SEMESTER II TAHUN 2024/2025

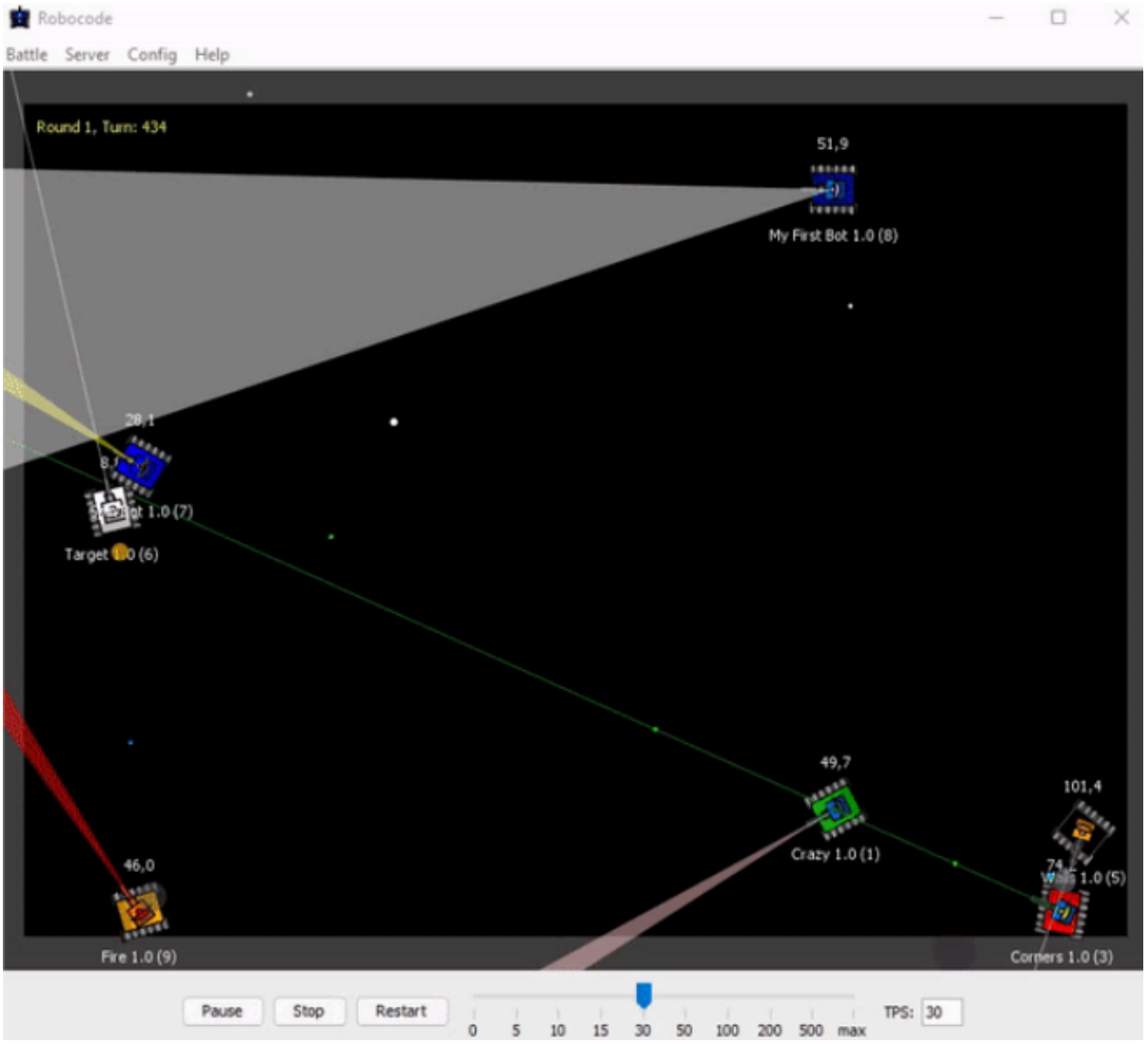
Disusun Oleh:
Kelompok TankRakus

Andi Farhan Hidayat	(13523128)
Andri Nurdianto	(13523145)
Muhammad Farrel Wibowo	(13523153)

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

Daftar Isi

Bab 1: Deskripsi Tugas



Gambar 1. Robocode Tank Royale

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.

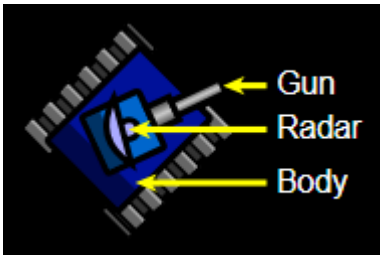
6. Tabrakan

   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.

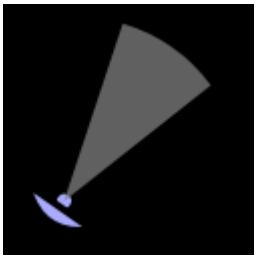
   Bagian paling lambat adalah tubuh tank, yang dalam kondisi terbaik dapat berputar hingga 10 derajat per giliran. Namun, ini bergantung pada kecepatan bot saat ini. Semakin cepat bot bergerak, semakin lambat kemampuannya untuk berbelok.

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

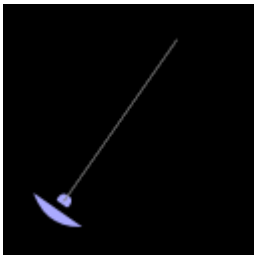
10. Pemindaian

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

2.1. Algoritma Greedy

Algoritma *Greedy* adalah algoritma yang paling sering digunakan karena sederhana untuk memecahkan persoalan optimasi, yaitu persoalan yang mencari solusi optimal. Terdapat dua macam persoalan optimasi, yaitu maksimasi (*maximation*) dan minimasi (*minimization*). Algoritma *Greedy* merupakan algoritma yang memecahkan persoalan secara langkah per langkah (*step-by-step*) sedemikian sehingga setiap langkah:

- Mengambil pilihan terbaik yang dapat diperoleh pada saat itu tanpa memperhatikan konsekuensi kedepannya, menerapkan prinsip *"take what you can get now!"*.
- "Berharap" bahwa dengan memilih optimum lokal pada setiap langkah akan berakhir dengan optimum global.

Terdapat penggunaan elemen-elemen penentu strategi pada algoritma *greedy*, yaitu:

1. Himpunan kandidat C yang berisi kandidat pilihan pada setiap langkah (contohnya simpul/sisi di dalam graf, job, tugas, koin, benda, karakter, dsb.).
2. Himpunan solusi S berisi kandidat yang sudah dipilih.
3. Fungsi solusi yang menentukan himpunan kandidat yang dipilih sudah memberikan solusi atau tidak.
4. Fungsi seleksi (*selection function*) yang memilih kandidat berdasarkan strategi *greedy* tertentu. Strategi ini bersifat heuristik.
5. Fungsi kelayakan (*feasible*) memeriksa kandidat yang dipilih dapat dimasukkan ke dalam himpunan solusi (layak atau tidak).
6. Fungsi objektif yang memaksimumkan atau meminimumkan.

Dengan elemen-elemen tersebut, maka algoritma greedy melibatkan pencarian sebuah himpunan bagian S dari himpunan kandidat C yang dalam hal ini S harus memenuhi beberapa kriteria yang ditentukan, yaitu S menyatakan suatu solusi dan S dioptimasi oleh fungsi objektif.

Algoritma *Greedy* tidak beroperasi secara menyeluruh terhadap semua kemungkinan solusi yang ada. Algoritma ini juga memungkinkan fungsi seleksi yang berbeda, sehingga harus dipilih fungsi yang tepat jika ingin menghasilkan solusi yang optimal. Oleh sebab itu, pada sebagian persoalan, algoritma *greedy* tidak selalu berhasil memberikan solusi yang optimal global, tetapi memberikan solusi sub-optimal.

2.2. Robocode

Robocode adalah permainan pemrograman yang bertujuan membuat kode bot. Bot adalah tank yang harus bersaing dengan tank lain di arena pertempuran virtual. "Pemain" permainan ini adalah pemrogram bot yang tidak akan memiliki pengaruh langsung pada permainan saat sedang berjalan. Pemain ("Robocoder") harus menulis program yang membentuk otak tank. Program tersebut memberi tahu bagaimana tank harus berperilaku dan bereaksi terhadap peristiwa yang terjadi di arena pertempuran. Kode tersebut memberi tahu tank terkait cara bergerak di medan perang dan menghindari tembakan dari musuh saat bergerak dalam situasi yang berbeda. Pada saat yang sama, tank harus memindai tank musuh dengan memutar radarnya dan menembakkan senjatanya ke arah tank musuh. Semua bot (tank) mendapatkan skor dan penempatan tergantung pada seberapa baik kinerja mereka di medan perang. Tujuan dari permainan ini adalah untuk mendapatkan skor dan peringkat setinggi-tingginya.

2.3. Heuristic

Heuristic adalah metode pemecahan masalah dengan mencari solusi yang cepat dan efektif. Heuristic digunakan dengan mengetahui terlebih dahulu kondisi-kondisi berdasarkan tujuan yang diinginkan, heuristic mengeliminasi beberapa kemungkinan terlebih dahulu agar tidak perlu eksplorasi lebih lanjut dalam penanganannya. Metode ini dibuat untuk memecahkan masalah secara langsung dengan mengabaikan apakah metode tersebut benar secara matematis. Berikut adalah beberapa alasan heuristic dapat membantu pemecahan masalah:

1. Efisiensi: Heuristik dapat secara signifikan mengurangi waktu dan sumber daya yang dibutuhkan untuk memecahkan masalah, terutama ketika ruang pencarian sangat besar, atau masalahnya terlalu rumit.
2. Kesederhanaan: Metode heuristik sering kali lebih mudah diimplementasikan dan dipahami daripada algoritma yang rumit, sehingga lebih mudah diakses oleh khalayak yang lebih luas.
3. Fleksibilitas: Heuristik dapat diterapkan ke berbagai domain masalah, sering kali dengan sedikit modifikasi, sehingga menjadikannya alat pemecahan masalah yang serbaguna.
4. Ketahanan: Dalam situasi di mana data yang tersedia tidak lengkap, tidak pasti, atau tidak jelas, heuristik tetap dapat memberikan solusi yang bermanfaat, sehingga dapat diterapkan pada masalah di dunia nyata.

Heuristic berguna dalam situasi yang kompleks atau memiliki ruang pencarian yang luas, karena mampu menyederhanakan proses pencarian solusi. Keunggulan heuristic adalah efisiensi, kesederhanaan, fleksibilitas, dan ketahanan sehingga cocok digunakan dalam pemecahan masalah terutama dalam pemrograman, terlebih ketika data yang tersedia terbatas atau tidak pasti. Meskipun tidak selalu memberikan solusi optimal, heuristic tetap menjadi pendekatan yang efektif untuk pengambilan keputusan dan pemecahan masalah di dunia nyata.

Bab 3: Aplikasi Strategi Greedy

3.1. Mapping Elemen Algoritma Greedy

3.1.1. Himpunan Kandidat (C)

a. Energi

   Mencari poin dengan mengutamakan energi, jika energi masih banyak kemungkinan untuk bertahan menjadi nomor 1 akan semakin besar dan mendapatkan poin dari survival skor.

b. Jarak dengan Lawan

   Semakin besar jarak dengan lawan, lawan akan semakin sulit mengenai tank yang dimiliki karena bisa dihindari.

c. Arah Pergerakan

   Arah pergerakan akan memengaruhi penargetan tank lawan sehingga tank bisa menghindar dari peluru lawan ataupun tank lawan secara keseluruhan. Selain itu, arah pergerakan dapat digunakan untuk mendekati lawan untuk melakukan pertempuran ataupun menjauhi tank lawan untuk menghindari pertempuran

d. Daya Tembak Peluru

   Daya tembak peluru yang dikeluarkan akan mengakibatkan energi lawan berkurang, selain itu daya tembak akan memulihkan energi tank berjumlah 3 kali lipat dari energi yang dikeluarkan jika mengenai tank lawan

e. Sapuan Radar

   Sapuan radar dapat mengetahui lokasi musuh, sapuan radar dapat membantu untuk menghindari tank lawan ataupun menargetkan tank lawan dan bertempur dengan tank lawan.

f. Jumlah Lawan

   Jumlah lawan bisa memperbesar poin dengan mencari lawan dengan energi terendah dan mendapatkan bonus apabila berhasil membunuh (mendapatkan *hit* terakhir). Poin yang didapat bisa 20% dari damage jika berhasil membunuh menggunakan peluru ataupun 30% dari damage jika berhasil membunuh dengan menggunakan tabrakan.

3.1.2. Himpunan Solusi (S)

Aksi-aksi yang dilakukan setiap *turn* (bergerak, memindai, dan menembak) beserta arah dan besarannya.

3.1.3. Fungsi Solusi

a. Saat radar mendeteksi lawan, serang dengan daya tembak optimal
b. Saat energi bot rendah, hindari dan menjauh dari pertempuran
c. Saat bot mengalami *gun heat*, hindari dan menjauh dari pertempuran
d. Saat radar tidak mendeteksi lawan, *sweeping* radar dilakukan
e. Saat tertembak, ubah arah pergerakan
f. Saat lawan mendekat, ubah arah pergerakan untuk menghindari (*ramming*) lawan

3.1.4. Fungsi Seleksi

a. Targetkan dan eliminasi lawan dengan energi terendah dalam deteksi radar
b. Targetkan dan eliminasi lawan dengan jarak terdekat

c. Targetkan dan eliminasi lawan dengan arah pergerakan yang menjauhi bot (dengan anggapan arah radar sesuai dengan arah pergerakan).
d. Saat tertembak, bergerak secara zigzag sambil *sweeping* radar area sekitar dan targetkan lalu eliminasi lawan dengan energi yang berkurang sesuai dengan perhitungan damage yang diterima.
e. Saat jumlah pemain masih di atas 50% hindari area yang berkerumun (mode defensif). Ketika di bawah 50%, masuki area pertempuran (mode ofensif).
f. Saat lawan sangat dekat dan masuk area dengan kepastian 100% terkena tembakan, targetkan dan tembakkan peluru dengan daya tembak 50% dari total energi.
g. Saat menargetkan lawan, tembakkan hanya pada arah dengan kemungkinan terkena tertinggi hasil perhitungan perkiraan posisi lawan berdasarkan kecepatan dan arahnya, serta kecepatan dari peluru yang ditembakkan.
h. Saat menargetkan lawan, tembakkan peluru dengan energi seoptimal mungkin tetapi tidak menyebabkan *gun heat*.

3.1.5. Fungsi Kelayakan (feasible)

Pemeriksaan pada aksi bot sehingga:

a. Pergerakan bot tidak menabrak dinding
b. Bot tidak menembak ketika energi tersisa sedikit (< 10)
c. Bot tidak menembak secara berlebihan pada target yang tidak pasti

3.1.6. Fungsi Objektif

Tujuan utama bot dalam *Robocode Tank Royale* adalah mencapai total skor setinggi-tingginya, skor dipengaruhi oleh:

a. *Bullet Damage*: Tambahan skor sebesar damage
b. *Bullet Damage Bonus*: Tambahan skor sebesar 20% damage
c. *Survival Score*: Tambahan 50 skor tiap bot lain mati
d. *Last Survival Bonus*: Tambahan 10 skor dikali banyaknya musuh
e. *Ram Damage*: Tambahan skor sebesar 2 kali damage

3.2. Eksplorasi Alternatif Solusi

3.2.1. Greedy berdasarkan Energi Bot

Bot akan memprioritaskan bertahan hidup ketika energi berada di bawah ambang batas tertentu (misalnya 30). Bot akan cenderung menjauh dari kerumunan dan menghindari pertempuran. Strategi ini meningkatkan peluang mendapatkan survival score dan last survival bonus. Namun secara skor damage peluru menjadi lebih rendah karena sedikit berpartisipasi dalam pertempuran.

3.2.2. Greedy berdasarkan Energi Lawan

Bot akan memilih lawan dengan energi paling rendah yang terdeteksi dalam radar untuk segera dieliminasi. Strategi ini akan memaksimalkan peluang mendapatkan bullet damage bonus dan ram damage bonus. Namun berisiko jika bot mendekati lawan dalam

kondisi berbahaya (banyak musuh di sekitar atau dalam mode defensif).

3.2.3. Greedy berdasarkan Jarak
Bot akan menargetkan lawan yang paling dekat atau menjauh dari lawan yang terlalu dekat. Kelebihan yang dimiliki antara lain adalah dapat meningkatkan akurasi tembakan dan menghindari tabrakan.

3.2.4. Greedy berdasarkan Kepadatan Area (Crowd Avoidance)
Bot akan secara aktif menghindari area dengan banyak musuh. Bot akan dapat menghindari risiko besar terkena peluru dari banyak arah sehingga peluang untuk bertahan sampai akhir akan semakin besar.

3.2.5. Greedy Hybrid Offense-Defense
Menggabungkan pendekatan ofensif dan defensif, dengan strategi yang berganti tergantung pada kondisi (jumlah pemain, energi bot, suhu meriam). Strategi ini adaptif terhadap situasi pertempuran.

3.2.6. Greedy Damage Peluru
Bot fokus menembak dengan daya tembak maksimal yang aman untuk memaksimalkan *bullet damage* dan *bonus*. Cocok untuk strategi agresif. Namun strategi ini dapat membuat meriam cepat panas atau energi cepat habis jika tidak dikontrol.

3.2.7. Greedy Damage Ramming
Bot menabrak musuh, terutama yang diam atau berenergi rendah, untuk mendapat ram damage dan bonus. Efektif jarak dekat, namun berisiko terkena damage sendiri atau dikepung musuh.

3.2.8. Greedy Damage Obstacle (Dinding)
Bot mencoba menggiring musuh ke dinding agar terkena *wall damage*, sambil menghindari tabrakan sendiri. Efektif melawan musuh dengan navigasi buruk, tapi sulit diterapkan secara konsisten.

3.3. Analisis Efisiensi dan Efektivitas

Strategi	Efektifitas	Kompleksitas Waktu	Catatan
Greedy berdasarkan Energi Bot	Menjaga survival	O(1)	Sangat efisien, cocok untuk early game
Greedy berdasarkan Energi Lawan	Mengejar kill & bonus	O(n)	Agresif, harus hati-hati terhadap risiko

Greedy berdasarkan Jarak	Akurasi & penghindaran	O(n)	Akurat, cocok untuk bot dengan aim bagus
Greedy berdasarkan Kepadatan Area (Crowd Avoidance)	Menghindari kerusuhan	O(n)	Perlu sistem area atau radius
Greedy Hybrid Offense-Defense	Adaptif terhadap kondisi	O(n)	Efektif, tetapi lumayan kompleks
Greedy Damage Peluru	Fokus maksimalkan damage tembakan	O(n)	Efektif jika akurat, tapi risiko overheat
Greedy Damage Ramming	Menabrak musuh untuk bonus	O(1)	
Greedy Damage Obstacle (Dinding)	Giring musuh ke dinding	O(n)	Efektif situasional, sulit konsisten

3.4. Strategi Greedy Pilihan

(ini niatnya mau dibuat dulu klasifikasi solusi2 yang ada (dikelompokkan), trus dijadiin satu, contoh yang baru kepikiran:

1. Greedy Damage Peluru

   Salah satu faktor dari skoring adalah berdasarkan damage peluru. Greedy dengan maksimasi damage peluru maka akan memaksimasi skor bot pula. Selain itu, energy yang didapatkan dari peluru yang berhasil mengenai bot lawan dapat digunakan kembali untuk menembak lawan lainnya sehingga menghasilkan poin lebih besar pula. *Energy* yang semakin bertambah seiring dengan peluru yang berhasil memberikan damage pada lawan juga mengakibatkan bot dapat bertahan lebih lama sehingga menerima skor survival pula yang semakin menambah total skor bot.

   Maksimasi damage peluru berarti *energy* yang ada dialokasikan dan difokuskan untuk menembak. Oleh sebab itu, dilakukan aksi-aksi preventif untuk menjaga agar energy bot tidak terbuang sia-sia. Seperti dilakukan prosedur yang menghindari wall dan menjauh dari musuh yang menabrak atau tertabrak oleh bot.

   Fire damage pada bot ini diatur cukup maksimum sehingga memaksimasi skor dan energy yang didapatkan dari peluru yang berhasil mengenai lawan. Selain itu, diperlukan pula mekanisme untuk memaksimasi kemungkinan peluru terkena oleh lawan, yaitu dengan melakukan perhitungan kecepatan peluru, kecepatan lawan, jarak

lawan, arah, dan posisi dengan lawan untuk memperkirakan kemungkinan terbesar lokasi lawan selanjutnya.

Jarak juga berperan penting dalam greedy ini, sehingga bot perlu mekanisme untuk meminimasi jarak antara bot dengan bot lawan sehingga kemungkinan peluru mengenai bot lawan semakin besar.

2. Greedy Kill Musuh

   Saat membunuh/menghancurkan musuh, bot akan mendapatkan skor bonus. Berdasarkan hal tersebut, dilakukan greedy untuk maksimasi jumlah menghancurkan bot lain sehingga dapat memaksimasi total skor bot pula. Greedy ini memerlukan mekanisme targeting sehingga dapat memastikan bahwa bot yang telah ditarget akan dihancurkan sehingga mendapatkan bonus poin kill.

   Selain itu, untuk memaksimasi jumlah kill, cara tercepat untuk melakukan kill adalah dengan melakukan penembakan terus menerus pada bot lawan. Oleh sebab itu, perlu alokasi yang baik pada energy sehingga tidak terbuang sia-sia. Mekanisme penghindaran dinding dan bot lawan yang menabrak atau tertabrak oleh bot ini tentu dapat meminimasi energy yang terbuang sia-sia.

   Kecepatan dan damage peluru tentu menjadi hal yang penting. Tanpa kecepatan maka bot target yang sudah sekarat bisa dihancurkan oleh bot lainnya. Tanpa damage yang cukup, menghancurkan bot lawan akan memakan waktu yang lama sehingga akan mengurangi jumlah bot yang dapat dihancurkan. Oleh sebab itu, fire damage bot ini diatur agar seimbang antara kecepatan dan *damage*-nya, yaitu bernilai 2.

   Kecepatan dalam eksekusi tentu merupakan hal yang penting. Oleh sebab itu, bot ini akan langsung menargetkan bot pertama yang terdeteksi radar. Selain itu, diperlukan mekanisme minimasi jarak antara bot dengan bot lawan untuk memperbesar kemungkinan damage yang dapat diberikan.

3. Greedy Damage (Ramming dan Peluru).

   Poin yang didapat dari damage peluru dapat ditambahkan dengan poin yang didapat ketika menabrak tank lawan. Dengan menggunakan damage ramming dan peluru, kemungkinan mendapatkan poin bonus dengan kedua damage bisa dicapai.

   Dengan mendekatnya bot ke tank lawan saat pertempuran, kemungkinan damage dari peluru mengenai lawan akan bertambah. Lalu, damage *raming* bisa didapat dengan menabrakan diri ke tank lawan. *Fire damage* pada bot ini dibuat besar karena ketika mendekati lawan kemungkinan terkenanya akan besar sehingga akan ada energi yang dipulihkan oleh bot tersebut.

   Namun, ada kelemahan yang bisa terlihat ketika ingin memaksimalkan damage dari *ramming*, yaitu ketika bot menuju lawan, bot akan bergerak secara linear. Hal itu menyebabkan kemungkinan survival time akan berkurang karena sudah mati terlebih dahulu.

4. Greedy Survival Time

Dengan bertahan sampai akhir, dari setiap bot yang mati terlebih dahulu kita mendapatkan poin tambahan apalagi jika bertahan hingga akhir round maka akan mendapatkan poin bonus. pada strategi pilihan ini bot akan dirancang untuk menghindari pertempuran secara langsung dan cenderung untuk menghindar baik ketika menabrak, ditabrak maupun ditembak dengan metode zigzag serta dengan inisialisasi gerakan awal adalah memutari wall dan bergerak sepanjang wall.

Walaupun bot ini dirancang untuk meghindari pertempuran secara langsung, ketika bot ditembak maupun ditabrak gun akan mengarah pada arah peluru dan arah asal tabrakan dan menembakan peluru berdasarkan jarak untuk pengeluaran energi yang lebih efisien. Karena apabila peluru terkena sasaran maka akan mendapat pengembalian energi untuk keberlangsungannya survival.

Tentunya strategi-strategi tersebut diaplikasikan dengan beberapa syarat tambahan seperti mode hemat energi ketika berada dalam state energi tertentu yang membuatnya lebih jarang menembak dan lari dari lawan lebih jauh untuk menyelamatkan nyawa.

Bab 4: Implementasi dan Pengujian

4.1. Strategi Greedy Utama (TankRakus)

a. Struktur data

No.	Nama Variable	Tipe Data	Keterangan
1.	enemySpeed	double	Kecepatan musuh
2.	isEnemyScanned	boolean	Kondisi apakah target musuh telah di-*scan*
3.	lastEnemyX	double	Koordinat X terakhir bot target
4.	lastEnemyY	double	Koordinat Y terakhir bot target
5.	enemyDirection	double	Arah pergerakan musuh
6.	wallMargin	integer	Batas jarak antara bot dengan dinding
7.	radarTurn	double	Sudut tujuan perputaran radar
8.	distanceToEast	double	Jarak bot dengan dinding timur
9.	distanceToWest	double	Jarak bot dengan dinding barat
10.	distanceToNorth	double	Jarak bot dengan dinding utara
11.	distanceToSouth	double	Jarak bot dengan dinding selatan
12.	centerX	double	Posisi tengah arena secara horizontal
13.	centerY	double	Posisi tengah arena secara vertikal
14.	angleToCenter	double	Bearing antara bot dengan titik tengah arena
15.	hitBotX	double	Koordinat X bot yang tertabrak
16.	hitBotY	double	Koordinat Y bot yang tertabrak
17.	bearingToEnemyBot	double	Bearing antara bot dengan posisi bot lawan yang menabrak/tertabrak

b. Fungsi

No.	Nama Fungsi	Tipe Data Keluaran	Keterangan
1.	isTooCloseWall()	boolean	Menentukan apakah bot sudah terlalu dekat dengan dinding atau tidak

c. Prosedur

No.	Nama Variable	Keterangan
1.	Main(string[] args)	Prosedur utama bot
2.	SetColors()	Prosedur untuk mendeklarasikan warna bot
3.	Run()	Prosedur saat program bot dijalankan, terdapat looping untuk dilakukan tiap turn
4.	AvoidWall()	Prosedur berisi aksi untuk mengarah ke pusat arena.
5.	FirePredict()	Memperkirakan posisi musuh selanjutnya untuk kemudian dilakukan penembakan pada arah prediksi musuh tersebut. Prediksi dilakukan dengan data kecepatan peluru, arah pergerakan musuh, kecepatan musuh, dan jarak dengan musuh.
6.	OnHitBot()	Prosedur aksi menjauh ketika menabrak atau ditabrak musuh
7.	OnScannedBot()	Melakukan scan pada bot yang ada di sekitar kemudian melakukan penembakan pada bot pertama yang dilihat. Pada scan bot ini juga menyimpan data kecepatan, posisi, dan arah dari bot target.

d. Pseudocode

```
BEGIN TankRakus

  DEFINE isEnemyScanned AS BOOLEAN
  DEFINE lastEnemyX, lastEnemyY, enemySpeed, enemyDirection AS
DOUBLE
  DEFINE wallMargin AS INTEGER = 100

  FUNCTION Main()
    INITIALIZE TankRakus
    CALL Start()
```

```
FUNCTION SetColors()
  SET BodyColor □ DarkBlue
  SET TurretColor □ OrangeRed
  SET RadarColor □ Cyan
  SET BulletColor □ Gold
  SET ScanColor □ GreenYellow
  SET TracksColor □ Gray
  SET GunColor □ Crimson

FUNCTION Run()
  CALL SetColors()
  SET AdjustRadarForGunTurn □ TRUE
  SET AdjustRadarForBodyTurn □ TRUE
  SET AdjustGunForBodyTurn □ TRUE

  SET isEnemyScanned □ FALSE
  SET TargetSpeed □ 5

  WHILE IsRunning DO
    IF IsTooCloseToWall() THEN
      CALL AvoidWall()

    IF isEnemyScanned THEN
      SET radarTurn □ RadarBearingTo(lastEnemyX, lastEnemyY)
      CALL SetTurnRadarLeft(radarTurn)
      CALL SetTurnLeft(BearingTo(lastEnemyX, lastEnemyY))
      CALL FirePredict()
      SET isEnemyScanned □ FALSE
    ELSE
      PRINT "Not Locking"
      CALL SetTurnRadarRight(20)

    CALL Go()

FUNCTION IsTooCloseToWall() RETURNS BOOLEAN
  SET distanceToEast □ ArenaWidth - X
  SET distanceToWest □ X
  SET distanceToNorth □ ArenaHeight - Y
  SET distanceToSouth □ Y

  RETURN (distanceToNorth < wallMargin OR
          distanceToSouth < wallMargin OR
          distanceToEast < wallMargin OR
```

```
            distanceToWest < wallMargin)

  FUNCTION AvoidWall()
    SET centerX □ ArenaWidth / 2
    SET centerY □ ArenaHeight / 2
    SET angleToCenter □ BearingTo(centerX, centerY)
    CALL SetTurnLeft(angleToCenter)

  FUNCTION OnHitBot(e)
    SET hitBotX □ e.X
    SET hitBotY □ e.Y
    SET bearingToEnemyBot □ BearingTo(hitBotX, hitBotY)

    IF e.IsRammed THEN
                                                          CALL
SetTurnLeft(NormalizeRelativeAngle(bearingToEnemyBot + 90))
    ELSE
                                                          CALL
SetTurnLeft(NormalizeRelativeAngle(bearingToEnemyBot + 180))

  FUNCTION OnScannedBot(e)
    SET isEnemyScanned □ TRUE
    SET lastEnemyX □ e.X
    SET lastEnemyY □ e.Y
    SET enemySpeed □ e.Speed
    SET enemyDirection □ e.Direction

  FUNCTION FirePredict()
    SET distance = DistanceTo(lastEnemyX, lastEnemyY)
    PRINT "Distance: " + distance

    IF distance > 700 THEN
      RETURN

    SET firePower □ 3
    SET bulletSpeed □ CalcBulletSpeed(firePower)
    SET timeToHit □ distance / bulletSpeed
    SET directionRad □ enemyDirection * PI / 180

  SET predictedX □ lastEnemyX + enemySpeed * COS(directionRad)
* timeToHit
  SET predictedY □ lastEnemyY + enemySpeed * SIN(directionRad)
* timeToHit
```

```
    SET gunTurn  GunBearingTo(predictedX, predictedY)
    CALL SetTurnGunLeft(gunTurn)
    CALL Fire(firePower)

END TankRakus
```

4.2. Alternatif Pertama (Bothan)
Strategi: Optimasi skor berdasarkan damage peluru
strategi utama: g
strategi sekunder: a, e, f, h
(pake pseudocode)

a. Struktur data

No.	Nama Variable	Tipe Data	Keterangan
1.	isLocking	boolean	Kondisi apakah sedang melakukan locking bot musuh
2.	isEnemyScanned	boolean	Kondisi apakah target musuh telah di-*scan*
3.	lastEnemyX	double	Koordinat X terakhir bot target
4.	lastEnemyY	double	Koordinat Y terakhir bot target
5.	enemyDirection	double	Arah pergerakan musuh
6.	targetId	integer	ID target musuh
7.	lastEnemySeenTurn	integer	Turn terakhir pada saat target di-*scan*
8.	targetLostThreshold	integer	Threshold turn sehingga bot dapat dinyatakan hilang / mati untuk melakukan restart targeting
9.	wallMargin	integer	Batas jarak antara bot dengan dinding
10.	radarTurn	double	Sudut tujuan perputaran radar
11.	distanceToEast	double	Jarak bot dengan dinding timur
12.	distanceToWest	double	Jarak bot dengan dinding barat
13.	distanceToNorth	double	Jarak bot dengan dinding utara

14.	distanceToSouth	double	Jarak bot dengan dinding selatan
15.	centerX	double	Posisi tengah arena secara horizontal
16.	centerY	double	Posisi tengah arena secara vertikal
17.	angleToCenter	double	Bearing antara bot dengan titik tengah arena
18.	hitBotX	double	Koordinat X bot yang tertabrak
19.	hitBotY	double	Koordinat Y bot yang tertabrak
20.	bearingToEnemyBot	double	Bearing antara bot dengan posisi bot lawan yang menabrak/tertabrak

b. Fungsi

No.	Nama Fungsi	Tipe Data Keluaran	Keterangan
1.	isTooCloseWall()	boolean	Menentukan apakah bot sudah terlalu dekat dengan dinding atau tidak

c. Prosedur

No.	Nama Variable	Keterangan
1.	Main(string[] args)	Prosedur utama bot
2.	SetColors()	Prosedur untuk mendeklarasikan warna bot
3.	Run()	Prosedur saat program bot dijalankan, terdapat looping untuk dilakukan tiap turn
4.	AvoidWall()	Prosedur berisi aksi untuk mengarah ke pusat arena.
5.	OnHitBot()	Prosedur aksi menjauh ketika menabrak atau ditabrak musuh
6.	OnScannedBot()	Melakukan scan pada bot yang ada di sekitar kemudian melakukan locking pada bot pertama yang dilihat. Bot target tersebut akan disimpan ID-nya, sehingga jika hilang dari radar akan dilakukan

		pencarian terlebih dahulu hingga batas turn yang telah ditentukan. Pada scan bot ini juga menyimpan data kecepatan, posisi, dan arah dari bot target.
7.	OnBotDeath()	Memastikan apakah bot yang hancur adalah bot target, jika ya maka memanggil prosedur ResetTargeting()
8.	ResetTargeting()	Me-*restart* kondisi bot seperti awal (tidak melakukan targeting)

d. Pseudocode

```
BEGIN Bothan

  DEFINE isLocking, isEnemyScanned AS BOOLEAN
  DEFINE lastEnemyX, lastEnemyY, enemyDirection AS DOUBLE
  DEFINE targetId, lastEnemySeenTurn AS INTEGER
  DEFINE targetLostThreshold AS INTEGER = 50
  DEFINE wallMargin AS INTEGER = 100

  FUNCTION Main()
    INITIALIZE Bothan
    CALL Start()

  FUNCTION SetColors()
    SET BodyColor □ Black
    SET TurretColor □ Red
    SET RadarColor □ Cyan
    SET BulletColor □ Yellow
    SET ScanColor □ Green
    SET TracksColor □ Orange
    SET GunColor □ Magenta

  FUNCTION Run()
    CALL SetColors()
    SET AdjustRadarForGunTurn □ TRUE
    SET AdjustRadarForBodyTurn □ TRUE
    SET AdjustGunForBodyTurn □ TRUE

    SET isLocking □ FALSE
    SET isEnemyScanned □ FALSE
    SET TargetSpeed □ 5

    WHILE IsRunning DO
      IF IsTooCloseToWall() THEN
```

```
        CALL AvoidWall()

          IF isLocking AND (TurnNumber - lastEnemySeenTurn >
targetLostThreshold) THEN
        PRINT "Target lost - assuming dead or out of range"
        CALL ResetTargeting()

      IF isLocking AND isEnemyScanned THEN
        PRINT "Locking: " + targetId
        SET radarTurn = RadarBearingTo(lastEnemyX, lastEnemyY)
        CALL SetTurnRadarLeft(radarTurn)

        CALL SetTurnLeft(BearingTo(lastEnemyX, lastEnemyY))
                  CALL  SetTurnGunLeft(GunBearingTo(lastEnemyX,
lastEnemyY))
        CALL Fire(2)

        SET isEnemyScanned □ FALSE
      ELSE
        PRINT "Not Locking"
        CALL SetTurnRadarRight(20)

      CALL Go()

  FUNCTION IsTooCloseToWall() RETURNS BOOLEAN
    SET distanceToEast □ ArenaWidth - X
    SET distanceToWest □ X
    SET distanceToNorth □ ArenaHeight - Y
    SET distanceToSouth □ Y

    RETURN (distanceToNorth < wallMargin OR
            distanceToSouth < wallMargin OR
            distanceToEast < wallMargin OR
            distanceToWest < wallMargin)

  FUNCTION AvoidWall()
    SET centerX □ ArenaWidth / 2
    SET centerY □ ArenaHeight / 2
    SET angleToCenter □ BearingTo(centerX, centerY)
    CALL SetTurnLeft(angleToCenter)

  FUNCTION OnHitBot(e)
    SET hitBotX □ e.X
    SET hitBotY □ e.Y
```

```
    SET bearingToEnemyBot = BearingTo(hitBotX, hitBotY)

    IF e.IsRammed THEN
                                                                CALL
SetTurnLeft(NormalizeRelativeAngle(bearingToEnemyBot + 90))
    ELSE
                                                                CALL
SetTurnLeft(NormalizeRelativeAngle(bearingToEnemyBot + 180))

  FUNCTION OnScannedBot(e)
    IF NOT isLocking OR e.ScannedBotId == targetId THEN
      SET isLocking □ TRUE
      SET isEnemyScanned □ TRUE
      SET targetId □ e.ScannedBotId

      SET lastEnemySeenTurn □ TurnNumber
      SET lastEnemyX □ e.X
      SET lastEnemyY □ e.Y
      SET enemyDirection □ e.Direction

  FUNCTION OnBotDeath(e)
    IF e.VictimId == targetId THEN
      CALL ResetTargeting()

  FUNCTION ResetTargeting()
    SET isEnemyScanned □ FALSE
    SET isLocking □ FALSE
    SET targetId = -1

END Bothan
```

4.3. Alternatif Kedua (Kamikaze)
Strategi: Optimasi skor dengan mencari damage terbanyak
Strategi utama: mencari lawan dan menembaknya dalam posisi diam
Strategi sekunder: berjalan menuju lawan sambil menembakkan peluru ketika energi kurang dari 70

a. Struktur data

No.	Nama Variable	Tipe data	Keterangan
1.	e	bot	Berisi data yang ada pada tank lawan

2.	Energy	double	Jumlah energi yang dimiliki
3.	angleToEnemy	double	Menyimpan sudut ke tank lawan
4.	X	double	Koordinat X tank lawan
5.	Y	double	Koordinat Y tank lawan

b. Prosedur

No.	Nama Prosedur	Keterangan
1.	Run()	Prosedur utama yang dijalankan saat bot aktif. Berisi logika pewarnaan bot, orientasi awal, dan gerakan utama berulang.
2.	OnScannedBot()	Event ketika musuh terdeteksi oleh radar. Mengatur targeting, penembakan, dan penghindaran. Termasuk logika deteksi musuh diam dan pengaturan ulang radar.
3.	OnHitBot()	Event ketika bot menabrak musuh. Menghindar dari tabrakan, menyesuaikan arah, dan membalas.
4.	TurnToFaceTarget()	Event ketika bot lawan terdeteksi dan tank akan membelokan badan dan membidik ke arah target

c. Pseudocode

```
BEGIN Kamikaze

  FUNCTION Main()
    INITIALIZE Kamikaze
    CALL Start()

  FUNCTION SetColors()
    SET BodyColor □ Yellow
    SET TurretColor □ Blue
    SET RadarColor □ Green
    SET ScanColor □ Black
    SET BulletColor □ White

  FUNCTION Run()
    CALL SetColors()

    WHILE IsRunning DO
```

```
      CALL TurnGunLeft(360)

  FUNCTION OnScannedBot(e)
    IF Energy >= 70 THEN
      CALL TurnToFaceTarget(e.X, e.Y)
      CALL TurnGunLeft(GunBearingTo(e.X, e.Y))
      CALL Fire(5)
      CALL Rescan()
    ELSE
      CALL TurnToFaceTarget(e.X, e.Y)
      CALL TurnGunLeft(BearingTo(e.X, e.Y))
      CALL Forward(500)
      CALL Fire(5)
      CALL Back(20)

  FUNCTION OnHitBot(e)
    CALL TurnToFaceTarget(e.X, e.Y)
    IF e.IsRammed THEN
      CALL Fire(5)
      CALL Back(30)
      CALL Forward(20)
      CALL Fire(5)

  FUNCTION TurnToFaceTarget(x, y)
    SET angleToEnemy □ BearingTo(x, y)
    CALL TurnLeft(angleToEnemy)

END Kamikaze
```

4.4. Alternatif Ketiga (BotFrag)

Strategi: optimasi skor survival bonus untuk bertahan sebagai bot terakhir dengan menghindari pertempuran secara langsung dan menghindari apabila melihat bot lawan

a. Struktur data

Nama Variabel	Tipe Data	Keterangan
enemies	int	Menyimpan jumlah musuh di awal permainan.
moveAmount	double	Digunakan untuk menentukan jarak gerakan bot (maks dari panjang/tinggi arena).
turnDirection	int	Menyimpan arah belokan saat berpindah (1 atau -1),

		dipakai untuk zigzag atau penghindaran atau penentuan searah jarum jam atau berlawanan jarum jam
lowEnergyMode	bool	Menandakan apakah bot dalam mode energi rendah (< 20).
lastEnemyX, lastEnemyY	double	Menyimpan posisi X dan Y musuh terakhir untuk deteksi musuh diam.
stationaryEnemyCount	const int	Counter jumlah giliran musuh tetap di tempat (diam).

b. Prosedur

Prosedur	Deskripsi
Run()	Prosedur utama yang dijalankan saat bot aktif. Berisi logika pewarnaan bot, orientasi awal, dan gerakan utama berulang.
OnScannedBot()	Event ketika musuh terdeteksi oleh radar. Mengatur targeting, penembakan, dan penghindaran. Termasuk logika deteksi musuh diam dan pengaturan ulang radar.
OnHitBot()	Event ketika bot menabrak musuh. Menghindar dari tabrakan, menyesuaikan arah, dan membalas.
OnHitByBullet()	Event ketika bot terkena peluru. Menjalankan strategi penghindaran tergantung mode energi, serta mencoba membalas tembakan.

c. Pseudocode

```
BEGIN BotFrag

  DEFINE enemies AS INTEGER
  DEFINE moveAmount AS DOUBLE
  DEFINE turnDirection AS INTEGER = 1
  DEFINE lowEnergyMode AS BOOLEAN = FALSE

  DEFINE lastEnemyX, lastEnemyY AS DOUBLE = -1
  DEFINE stationaryEnemyCount AS INTEGER = 0
  DEFINE maxStationaryTurns AS INTEGER = 100
```

```
  FUNCTION Main()
    INITIALIZE BotFrag
    CALL Start()

  FUNCTION SetColors()
    SET BodyColor □ Black
    SET TurretColor □ HotPink
    SET RadarColor □ Teal
    SET BulletColor □ Red
    SET ScanColor □ LightPink
    SET TracksColor □ HotPink

  FUNCTION Run()
    CALL SetColors()

    SET moveAmount □ MAX(ArenaWidth, ArenaHeight)
    CALL TurnRight(Direction MOD 90)
    CALL Forward(moveAmount)

    SET enemies □ EnemyCount
    CALL TurnGunRight(90)

    WHILE IsRunning DO
      IF Energy < 20 THEN
        SET lowEnergyMode □ TRUE

      IF Direction MOD 90 ≠ 0 THEN
        IF turnDirection = -1 THEN
          CALL TurnLeft(Direction MOD 90)
        ELSE
          CALL TurnRight(Direction MOD 90)
      ELSE
        IF turnDirection = -1 THEN
          CALL TurnLeft(90)
        ELSE
          CALL TurnRight(90)

      CALL Forward(moveAmount)

  FUNCTION OnScannedBot(e)
    IF ABS(e.X - lastEnemyX) < 1 AND ABS(e.Y - lastEnemyY) < 1
THEN
      INCREMENT stationaryEnemyCount
```

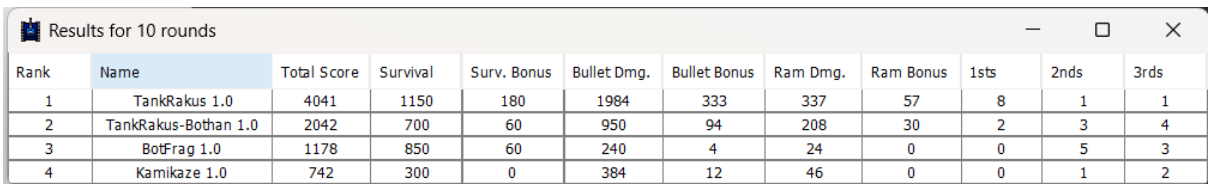
```
    ELSE
      SET stationaryEnemyCount □ 0

    SET lastEnemyX □ e.X
    SET lastEnemyY □ e.Y

    IF stationaryEnemyCount >= maxStationaryTurns THEN
      CALL Rescan()
      RETURN

    SET distance □ DistanceTo(e.X, e.Y)
    IF distance > 400 THEN
      CALL Rescan()
      RETURN

    SET bearingFromGun □ GunBearingTo(e.X, e.Y)
    CALL TurnGunLeft(bearingFromGun)

    SET power □ IF distance < 100 THEN 3 ELSE IF distance <
300 THEN 2 ELSE 1

    IF ABS(bearingFromGun) ≤ 4 THEN
      IF GunHeat = 0 AND Energy > 0.2 THEN
        SET firePower □ MIN(power, MAX(0.1, Energy - 0.1))
        CALL Fire(firePower)
      ELSE
        RETURN

    IF ABS(bearingFromGun) < 1 THEN
      CALL Rescan()

    IF NOT lowEnergyMode THEN
      SET turnDirection □ turnDirection * -1

  FUNCTION OnHitBot(e)
    SET bearing □ GunBearingTo(e.X, e.Y)
    CALL TurnGunLeft(bearing)
    SET turnDirection □ turnDirection * -1

    IF bearing > -90 AND bearing < 90 THEN
      CALL Back(100)
      CALL TurnLeft(90)
    ELSE
      CALL Forward(100)
```

```
      CALL TurnLeft(90)

    CALL Rescan()

  FUNCTION OnHitByBullet(e)
    IF Energy < 20 THEN
      SET lowEnergyMode □ TRUE
      IF turnDirection = 1 THEN
        CALL TurnRight(90)
        CALL Back(400)
      ELSE
        CALL TurnLeft(90)
        CALL Back(400)
    ELSE
      IF turnDirection = 1 THEN
        CALL TurnRight(90)
        CALL Back(200)
      ELSE
        CALL TurnLeft(90)
        CALL Back(200)

    CALL TurnRight(30)
    CALL Forward(100)
    CALL TurnLeft(30)

    SET bearingFromGun □ GunBearingTo(e.Bullet.X, e.Bullet.Y)
    CALL TurnGunLeft(bearingFromGun)
    CALL Rescan()
    SET turnDirection □ turnDirection * -1

END BotFrag
```

4.5. Pengujian

(semua bot 1 vs 1 vs 1 vs 1)

a. Percobaan 1



Results for 10 rounds

Rank	Name	Total Score	Survival	Surv. Bonus	Bullet Dmg.	Bullet Bonus	Ram Dmg.	Ram Bonus	1sts	2nds	3rds
1	TankRakus 1.0	4041	1150	180	1984	333	337	57	8	1	1
2	TankRakus-Bothan 1.0	2042	700	60	950	94	208	30	2	3	4
3	BotFrag 1.0	1178	850	60	240	4	24	0	0	5	3
4	Kamikaze 1.0	742	300	0	384	12	46	0	0	1	2

b. Percobaan 2

Results for 10 rounds

Rank	Name	Total Score	Survival	Surv. Bonus	Bullet Dmg.	Bullet Bonus	Ram Dmg.	Ram Bonus	1sts	2nds	3rds
1	TankRakus 1.0	4066	1200	210	1968	310	331	46	8	1	1
2	TankRakus-Bothan 1.0	1758	600	60	750	77	238	33	2	1	3
3	BotFrag 1.0	1514	850	30	569	11	29	24	0	6	3
4	Kamikaze 1.0	820	350	0	400	16	54	0	0	2	3

c. Percobaan 3

Results for 10 rounds

Rank	Name	Total Score	Survival	Surv. Bonus	Bullet Dmg.	Bullet Bonus	Ram Dmg.	Ram Bonus	1sts	2nds	3rds
1	TankRakus 1.0	4241	1400	270	2000	351	220	0	9	0	1
2	TankRakus-Bothan 1.0	1596	500	30	760	59	214	33	1	3	3
3	BotFrag 1.0	1085	600	0	453	7	25	0	0	4	2
4	Kamikaze 1.0	1055	500	0	480	32	43	0	0	3	4

4.6. Analisis Hasil Pengujian

4.6.1. Bot Utama (TankRakus)

Algoritma strategi greedy utama berhasil mendapatkan nilai optimal pada bullet damage. Hal tersebut juga cukup mempengaruhi survival point yang juga tinggi pada bot utama. Selain itu, terbukti bahwa strategi greedy utama berhasil mencapai nilai total skor tertinggi dengan strategi maksimasi damage peluru.

4.6.2. Bot Alternatif Pertama (Bothan)

Berhasil mencapai total nilai tinggi tetapi sayangnya memiliki nilai yang sedikit pada survival bonus.

4.6.3. Bot Alternatif Kedua (Kamikaze)

Berhasil mencapai ram damage yang lumayan baik, akan tetapi gagal mencapai nilai tinggi

4.6.4. Bot Alternatif Ketiga (BotFrag)

Berhasil mencapai nilai survival yang tinggi, tetapi tidak tertinggi

Bab 5: Kesimpulan dan Saran

5.1. Kesimpulan

Algoritma greedy mengutamakan target yang ingin dicapai tanpa melihat konsekuensi yang akan terjadi kedepannya. Algoritma ini dijalankan dengan harapan bahwa solusi yang dipilih merupakan solusi terbaik. Untuk beberapa kasus, algoritma ini memang bekerja dengan optimal dan dapat mencapai target yang dituju. Namun, ada kalanya algoritma ini tidak menghasilkan hasil akhir terbaik.

Algoritma *greedy* dapat digunakan dengan maksimal jika mengetahui bahwa algoritma ini memang akan menjadi solusi yang tepat dengan tercapainya target. Namun, ada kalanya kita, sebagai manusia, bukanlah seseorang yang dapat memperkirakan masa depan. Oleh sebab itu, rencanakan algoritma *greedy* sebaik mungkin dengan menentukan target terbaik, meskipun kita tidak mengetahui konsekuensi yang akan datang.

5.2. Saran

- Mempelajari lebih dalam mengenai algoritma yang akan digunakan.
- Membaca dan mempelajari dokumentasi robocode.
- Mempelajari dan mendalami bahasa C# yang digunakan dalam pembuatan bot dalam robocode.

Lampiran

1. Tabel Kelengkapan

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	✓	

2. Tautan *Repository* GitHub:
   https://github.com/andi-frame/Tubes1_TankRakus

3. Tautan Video:
   https://youtu.be/6QelF7q3jBM

Daftar Pustaka

Munir, Rinaldi. 2025. Algoritma Greedy (Bagian 1). Diakses melalui https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2024-2025/02-Algoritma-Brute-Force-(2025)-Bag1.pdf

Munir, Rinaldi. 2025. Algoritma Greedy (Bagian 1). Diakses melalui https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2024-2025/03-Algoritma-Brute-Force-(2025)-Bag2.pdf

Vidjikant, S., Sorokopud, O. 2024. Guide to Heuristics in Computer Science and Programming https://softjourn.com/insights/heuristic-programming

Robocode. 2024. What is Robocode?. Diakses melalui https://robocode-dev.github.io/tank-royale/articles/intro.html