Laporan Tugas Besar 1 IF 2211

Strategi Algoritma

Penerapan Algoritma Greedy dalam Robocode Tank Royale

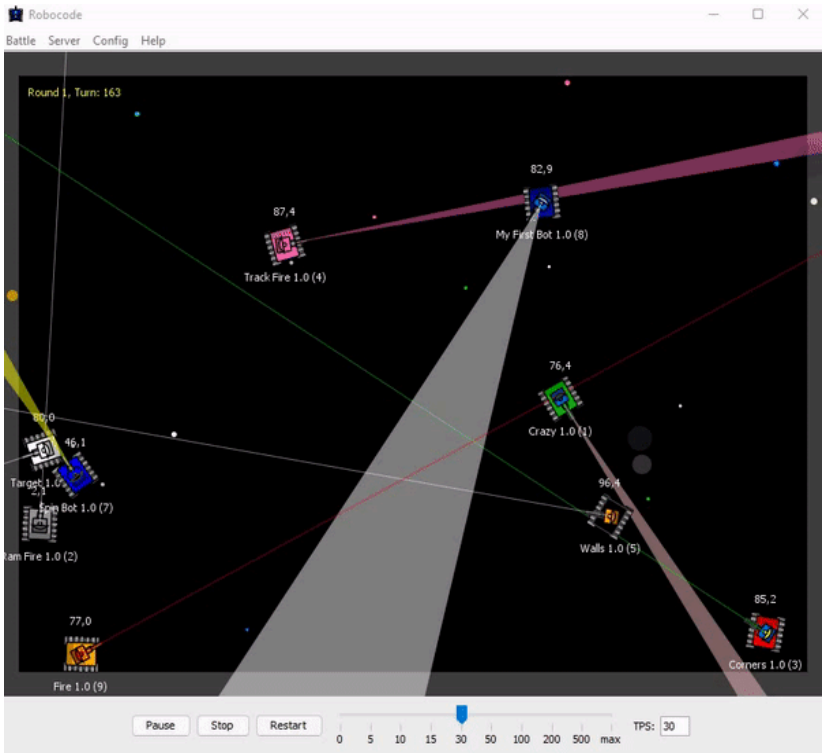
Disusun Oleh Kelompok Cisli, terdiri dari:

- 13523003 - Dave Daniell Yanni
- 13523033 - Alvin Christopher Santausa
- 13523036 - Yonatan Edward Njoto

Daftar Isi

BAB I

Deskripsi Tugas



Gambar 1. Visualisasi 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.

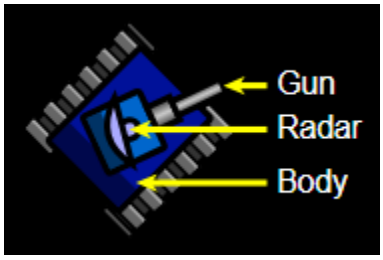
6. Tabrakan

   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 II

Landasan Teori

2.1 Algoritma Greedy

Algoritma greedy adalah salah satu metode paling sederhana dan populer untuk menyelesaikan masalah optimasi. Masalah optimasi sendiri terbagi menjadi dua jenis, yaitu maksimasi (maximization) dan minimasi (minimization).

Dalam algoritma greedy, solusi dibangun secara bertahap dengan membuat keputusan di setiap langkah. Meskipun ada banyak pilihan yang dapat dievaluasi, algoritma ini selalu memilih opsi terbaik pada saat itu (optimum lokal) tanpa mempertimbangkan konsekuensi jangka panjang. Selain itu, keputusan yang telah dibuat tidak dapat diubah kembali. Tujuan utama dari pendekatan ini adalah agar setiap langkah menuju solusi optimal secara keseluruhan (optimum global).

Komponen utama dalam algoritma greedy meliputi:

1. **Himpunan kandidat (C)**: Kumpulan elemen yang dapat dipilih dalam proses pembentukan solusi (misalnya simpul atau sisi dalam graf, tugas, pekerjaan, koin, benda, atau karakter).
2. **Himpunan solusi (S)**: Kumpulan elemen yang telah dipilih sebagai bagian dari solusi.
3. **Fungsi solusi**: Mengevaluasi apakah himpunan kandidat yang dipilih sudah menghasilkan solusi yang valid.
4. **Fungsi seleksi**: Memilih elemen dari himpunan kandidat berdasarkan strategi greedy tertentu yang bersifat heuristik.
5. **Fungsi kelayakan**: Memeriksa apakah elemen yang dipilih dapat dimasukkan ke dalam himpunan solusi tanpa melanggar batasan yang ada.
6. **Fungsi objektif**: Bertujuan untuk maksimasi atau minimasi nilai solusi yang diperoleh.

Meskipun sederhana dan cepat, algoritma greedy memiliki beberapa kelemahan:

1. **Tidak Selalu Menjamin Solusi Optimal**
    - Algoritma ini hanya mempertimbangkan keuntungan lokal tanpa melihat efek jangka panjang.
    - Contoh: Dalam **Coin Change Problem**, algoritma greedy tidak selalu menghasilkan solusi optimal jika sistem mata uang tidak memiliki properti greedy. Misalnya, untuk nominal koin {1, 3, 4} dan jumlah 6, greedy akan memilih {4, 1, 1}, sedangkan solusi optimal adalah {3, 3} karena pemilihan

secara greedy mengambil nilai terbesar terlebih dahulu.

2. **Bergantung pada Struktur Masalah**
    - Hanya efektif jika masalah memiliki **greedy choice** (bisa dipilih maksimum dan minimum)
    - Jika tidak, pendekatan lain seperti **dynamic programming** yang mungkin lebih baik.

3. **Tidak Dapat Mengoreksi Kesalahan**
    - Setelah membuat keputusan, algoritma tidak dapat kembali (backtrack) untuk memperbaikinya jika ternyata pilihan tersebut salah.
    - Berbeda dengan **dynamic programming**, yang menyimpan hasil perhitungan submasalah untuk menghindari perhitungan ulang dan menghasilkan solusi optimal.

Dengan begitu, algoritma ini tidak cocok untuk masalah yang membutuhkan eksplorasi solusi secara menyeluruh, seperti **Traveling Salesman Problem (TSP)** atau **knapsack problem** dengan kapasitas tertentu.

2.2 Cara Kerja Program

Cara Kerja Program Robocode

Robocode adalah permainan berbasis pemrograman di mana pemain merancang bot dalam bentuk tank virtual yang bertarung dalam sebuah arena. Pertarungan berlangsung dalam mode **Battle Royale**, di mana semua bot saling bertarung hingga hanya tersisa satu yang menjadi pemenang.

Struktur Permainan Robocode

- **Ronde dan Giliran**
    i. Pertempuran berlangsung dalam beberapa ronde, yang terbagi lagi ke dalam sejumlah giliran.

    ii. Setiap giliran memiliki batas waktu eksekusi yang berkisar antara **30 hingga 50 milidetik**.

- **Energi Bot**
    i. Setiap bot memulai pertempuran dengan **100 poin energi**.

ii. Energi akan berkurang jika terkena tembakan atau mengalami tabrakan dengan bot lain.

- **Sistem Peluru dan Kerusakan**
  i. Peluru dengan bobot lebih besar memiliki kecepatan lebih rendah tetapi menghasilkan **kerusakan lebih besar**.
  ii. Bot yang terkena peluru akan kehilangan energi sesuai dengan kekuatan tembakan lawan.

- **Panas Meriam (Gun Heat)**
  i. Bot tidak dapat menembak jika suhu meriamnya terlalu tinggi.
  ii. Perlu menunggu waktu pendinginan sebelum bisa menembak kembali.

- **Tabrakan dan Dampaknya**
  i. Jika bot bertabrakan dengan dinding atau bot lain, akan terjadi **penurunan energi**.
  ii. Semakin sering menabrak, semakin besar risiko kekalahan.

- **Bagian Tubuh Tank**
  i. **Body**: Komponen utama yang menentukan pergerakan tank.
  ii. **Gun**: Meriam yang digunakan untuk menembak musuh.
  iii. **Radar**: Sensor yang mendeteksi keberadaan lawan dalam arena.

- **Pergerakan dan Manuver**
  i. Bot dapat bergerak maju dan mundur dengan kecepatan yang dipengaruhi oleh momentum sebelumnya.
  ii. Bot juga dapat **berbelok** untuk mengubah arah.

- **Sistem Pemindaian (Radar Scan)**
  i. Radar bot dapat mendeteksi musuh dalam **jangkauan hingga 1200 piksel**.
  ii. Deteksi ini berguna untuk menentukan strategi menyerang atau bertahan.

- **Sistem Skor dan Poin**
  i. Poin diberikan berdasarkan beberapa faktor, seperti **damage yang diberikan ke lawan, bertahan lebih lama, serta aksi strategis lainnya**.
  ii. Semakin efektif bot dalam menyerang dan bertahan, semakin tinggi perolehan skornya.

Cara bot melakukan aksinya

Untuk dapat melakukan suatu pergerakan bot memerlukan instruksi yang disediakan melalui API, ada juga versi onlinenya di online-documentation-dotnet

Metode Dasar

Run()

Start()

Go()

Metode Kontrol Gerakan Bot

Maju dan Mundur:

Forward(double distance)

Back(double distance)

SetForward(double distance)

SetBack(double distance)

Berbelok:

TurnLeft(double degrees)

TurnRight(double degrees)

SetTurnRight(double degrees)

SetTurnLeft(double degrees)

Metode Kontrol Senjata dan Radar

Menggerakkan Senjata:

TurnGunLeft(double degrees)

TurnGunRight(double degrees)

SetTurnGunLeft(double degrees)

SetTurnGunRight(double degrees)

Menggerakkan Radar:

TurnRadarLeft(double degrees)

TurnRadarRight(double degrees)

SetTurnRadarLeft(double degrees)

SetTurnRadarRight(double degrees)

Pengaturan Relasi:

SetAdjustGunForRobotTurn(boolean independent)

SetAdjustRadarForGunTurn(boolean independent)

Metode Menembak dan Interaksi dengan Musuh

Menembak:

Fire(double power)

SetFire(double power)

FireBullet(double power)

Informasi Meriam:

GetGunHeat()

Metode Informasi Status Bot

Posisi dan Kecepatan:

getX()

getY()

getVelocity()

Arah dan Orientasi:

getHeading()

getGunHeading()

getRadarHeading()

Status Lainnya:

getEnergy()

getBattleTime()

getGunTurnRemaining()

getRadarTurnRemaining()

Metode Pemrosesan Event (Event Handling)

Deteksi Lawan:

onScannedBot(ScannedBotEvent event)

Ditembak atau Bertabrakan:

onHitByBullet(HitByBulletEvent event)

onHitWall(HitWallEvent event)

onHitBot(HitBotEvent event)

Peluru dan Kerusakan:

onBulletHit(BulletHitEvent event)

onBulletMissed(BulletMissedEvent event)

onBulletHitBullet(BulletHitBulletEvent event)

Kemenangan dan Kematian:

onRobotDeath(RobotDeathEvent event)

onWin(WinEvent event)

onDeath(DeathEvent event)

Metode Lainnya

Kontrol Pergerakan:

Stop()

Resume()

isRunning()

Kalkulasi Jarak dan Sudut:

DistanceTo(double x, double y)

BearingTo(double x, double y)

NormalizeRelativeAngle(double angle)

Cara Implementasi Algoritma Greedy ke dalam bot

Algoritma greedy merupakan teknik yang memilih langkah terbaik untuk diambil sekarang, oleh karena itu kita dapat menambahkannya pada fungsi start robot dan juga event handler.

Cara Menjalankan bot

1. Instalasi Robocode Tank Royale

1. Unduh dan pasang Robocode Tank Royale dari repositori resmi GitHub.
2. Pastikan sistem telah menginstal Java, C#, dan .NET versi yang sesuai.

2. Membuat Bot

1. Kembangkan bot dengan metode untuk bergerak, menembak, dan mendeteksi lawan.
2. Terapkan strategi greedy dalam setiap aksi bot untuk pengambilan keputusan optimal.

3. Menjalankan Simulasi

1. Tes kompilasi kode bot dengan menjalankan ./<nama bot>.cmd atau ./<nama bot>.sh di direktori yang berisi file .cmd untuk proses kompilasi.
2. Jalankan simulator menggunakan file .jar dari aplikasi utama.
3. Boot folder bot, lalu masukkan ke dalam arena.

4. Uji bot dengan berbagai strategi untuk menemukan pendekatan yang paling efektif.

4. Optimasi dan Penyempurnaan

1. Analisis hasil pertempuran dan identifikasi kelemahan yang perlu diperbaiki.
2. Sesuaikan parameter dalam strategi greedy guna meningkatkan performa dan daya tahan bot dalam pertandingan.

BAB III

Aplikasi Strategi Greedy pada Bot

3.1 Elemen Strategi Greedy pada Robocode

Himpunan Kandidat

Semua tindakan yang dapat dilakukan oleh bot pada setiap langkah, seperti:

- Bergerak maju/mundur
- Mengatur kecepatan pergerakan
- Berputar 0-360 derajat
- Menembak
- Menggerakan radar dan tembakan
- dll (semua fungsi yang ada di API)

Himpunan Solusi

Urutan tindakan yang dilakukan oleh bot dalam pertandingan untuk mencapai tujuan (bertahan hidup, mengalahkan lawan, mendapatkan poin yang besar).

Fungsi Solusi

Apakah rangkaian tindakan telah mencapai kondisi kemenangan atau mendapatkan poin terbesar.

Fungsi Seleksi

Memilih tindakan terbaik berdasarkan heuristic tertentu, seperti:

- Menembak jika musuh berada dalam jangkauan
- Menghindari peluru berdasarkan pola gerakan lawan
- Bergerak menuju posisi yang lebih aman

Fungsi Kelayakan

Memeriksa apakah suatu tindakan dapat dilakukan dalam kondisi tertentu, seperti:

- Bot tidak bisa menembak jika cooldown masih berlaku
- Bot tidak bisa bergerak lebih jauh jika ada batas arena
- Bot harus memiliki cukup energi untuk melakukan tindakan tertentu

Fungsi Objektif

Mengoptimalkan skor kemenangan dengan cara:

- Meminimalkan terkena serangan
- Membunuh musuh dengan "ramming"
- Memaksimalkan jumlah tembakan yang mengenai lawan
- Menggunakan energi secara efisien untuk bertahan lebih lama

3.2 Aplikasi Greedy Pada Bot

Aplikasi Strategi Greedy pada CisliCrookLVL1

Himpunan Kandidat, Himpunan Solusi, Fungsi Solusi, Fungsi Kelayakan, Fungsi Objektif sama dengan penjelasan secara general pada **3.1 Elemen Strategi Greedy pada Robocode**

Fungsi Seleksi

Memilih tindakan terbaik berdasarkan heuristic tertentu:

- Menembak jika musuh berada dalam jarak tertentu (<500 unit)
- Menghindari peluru musuh dengan bergerak secara random membentuk setengah lingkaran
- Pemilihan target terdekat jika ada beberapa lawan yang terdeteksi

Heuristik Greedy: Strategi Dinamis Berbasis Situasi

Dalam perancangan **CisliCrookLv1**, strategi greedy digunakan untuk mengambil keputusan dalam setiap langkah permainan berdasarkan informasi yang diperoleh dari arena. Berikut adalah pemetaan elemen algoritma greedy:

1. **Strategi pemilihan target**
   - Bot akan terlebih dahulu melakukan *scanning* area sekitar kemudian memilih musuh terdekat untuk meningkatkan kemungkinan peluru terkena musuh sehingga dapat menggunakan energi secara efisien
   - Bot akan menembak jika Id bot sama dengan bot yang dianggap terdekat dengan kriteria jarak terdekat atau terscan sebanyak 2x

2. **Strategi penyesuaian kekuatan penembakan**
    - Bot menembak dengan kekuatan penuh (3 unit) jika musuh berada dalam jarak dekat < 100 unit untuk meningkatkan kemungkinan mengenai target.
    - Jika musuh dalam jarak menengah 100 - 300 unit, bot menyesuaikan kekuatan tembakan menjadi 2 unit.
    - Jika musuh dalam jarak yang jauh (> 300 unit), bot menyesuaikan kekuatan ke 1 unit.
    - Jika terlalu jauh (> 500 unit) bot tidak akan menembak untuk menghemat energi karena kemungkinan kena sangatlah kecil
3. **Strategi Menghindar**
    - Bot sering terus bergerak secara random dengan lintasan setengah lingkaran untuk menghindari tembakan-tembakan musuh agar musuh kesulitan mengunci target.
    - Bot mengganti arah jika menabrak tembok

Analisis Efisiensi dan Efektivitas Alternatif Solusi

Strategi	**Keunggulan**	**Kelemahan**	**Efisiensi**	**Efektivitas**
Strategi Pemilihan Target	Efisiensi energi karena meningkatkan kemungkinan tembakan mengenai musuh yang lebih dekat	Membutuhkan waktu untuk menentukan target	Tinggi	Sedang
Strategi penyesuaian kekuatan penembakan	Efektif untuk pertempuran jarak dekat dan berguna untuk menghemat energi	Cepat menghabiskan energi	Sedang	Tinggi
Strategi Menghindar	Sulit diprediksi oleh musuh	Tidak efektif karena bergerak secara random, bisa saja	Tinggi	Sedang

		tertarak musuh/tembok		

Aplikasi Strategi Greedy pada CisliCrookLv10

Dalam perancangan CisliMafiaBossLv100, strategi greedy digunakan untuk mengambil keputusan dalam setiap langkah permainan berdasarkan informasi yang diperoleh dari arena. Berikut adalah pemetaan elemen algoritma greedy:

Himpunan Kandidat, Himpunan Solusi, Fungsi Solusi, Fungsi Kelayakan, Fungsi Objektif sama dengan penjelasan secara general pada **3.1 Elemen Strategi Greedy pada Robocode**

Fungsi Seleksi

Bot memilih tindakan yang memberikan keuntungan terbesar dalam situasi tertentu dengan mempertimbangkan faktor-faktor berikut:

a. **Menembak dengan Efisiensi Energi**
b. **Menghindari Serangan Musuh**

Heuristik Greedy: Strategi Dinamis Berbasis Situasi

Bot menyesuaikan perilaku berdasarkan posisi, jarak musuh, dan tingkat energi untuk mencapai keseimbangan antara agresivitas dan pertahanan.

1. **Strategi Menembak Berdasarkan Jarak**
    - Bot menembak berdasarkan jarak secara descending, jika musuh berjarak jauh maka bot akan menggunakan energi lebih sedikit untuk menembak.
2. **Strategi Menghindar**
    - Bot akan mengarahkan diri sedekat mungkin dengan 90 derajat dengan arah musuh sehingga pergerakan maju mundur untuk menghindari efektif.

Analisis Efisiensi dan Efektivitas Alternatif Solusi

Strategi	Keunggulan	Kelemahan	Efisiensi	Efektivitas

Menembak Secara Efisien	Memanfaatkan energi secara optimal dalam setiap serangan.	Bisa kurang agresif jika energi terlalu dijaga.	Sedang	Tinggi
Menghindari Serangan Musuh	Meminimalkan kerusakan dan meningkatkan daya tahan bot.	Bisa terlalu pasif dan sulit melakukan serangan yang efektif.	Tinggi	Sedang

Aplikasi Strategi Greedy pada CisliMafiaLv50

Himpunan Kandidat, Himpunan Solusi, Fungsi Solusi, Fungsi Kelayakan, Fungsi Objektif sama dengan penjelasan secara general pada **3.1 Elemen Strategi Greedy pada Robocode**

Fungsi Seleksi

Bot memilih tindakan yang memberikan keuntungan terbesar dalam situasi tertentu dengan mempertimbangkan faktor-faktor berikut:

a. **Efisiensi Penggunaan Energi untuk Menembak Musuh**
b. **Menghindari Tembakan Musuh**
c. **Menyerang Musuh secara Agresif**

Heuristik Greedy: Strategi Dinamis Berbasis Situasi

Dalam perancangan **CisliMafiaLv50**, strategi greedy digunakan untuk mengambil keputusan dalam setiap langkah permainan berdasarkan informasi yang diperoleh dari arena. Berikut adalah pemetaan elemen algoritma greedy:

1. **Strategi penyesuaian kekuatan penembakan**
   - Bot menembak dengan kekuatan penuh (3 unit) jika musuh berada dalam jarak dekat < 100 unit untuk meningkatkan kemungkinan mengenai target.
   - Jika musuh dalam jarak menengah 100 - 300 unit, bot menyesuaikan kekuatan tembakan menjadi 2 unit.
   - Jika musuh dalam jarak yang jauh (> 300 unit), bot menyesuaikan kekuatan ke 1 unit.

- Jika terlalu jauh (> 500 unit) bot tidak akan menembak untuk menghemat energi karena kemungkinan kena sangatlah kecil

2. **Strategi Menghindar**
   - Bot sering terus bergerak secara random dengan lintasan setengah lingkaran untuk menghindari tembakan-tembakan musuh agar musuh kesulitan mengunci target.
   - Bot mengganti arah jika menabrak tembok
3. **Strategi Target Lock, Mendekat, dan Menabrak**
   - Untuk meningkatkan akurasi penembakan, Bot akan mengunci bot musuh terdekat
   - Bot akan mendekati bot musuh yang terkunci untuk meningkatkan kemungkinan peluru kena
   - Bot akan menabrak bot musuh secara terus menerus hingga musuh hancur

Analisis Efisiensi dan Efektivitas Alternatif Solusi

Heuristik	Kelebihan	Kekurangan	Efisiensi	Efektivitas
Strategi penyesuaian kekuatan penembakan	Efektif untuk pertempuran jarak dekat dan berguna untuk menghemat energi	Cepat menghabiskan energi	Sedang	Tinggi
Strategi Menghindar	Sulit diprediksi oleh musuh	Tidak efektif karena bergerak secara random, bisa saja tertarak musuh/tembok	Tinggi	Sedang
Strategi Target Lock,	Tingkat akurasi	Dapat menjadi sasaran bersama	Efisiensi	Efektivitas

Mendekat, dan Menabrak	penembakan tinggi	dengan bot yang dikunci		

Aplikasi Strategi Greedy pada CisliMafiaBossLv100

Dalam perancangan CisliMafiaBossLv100, strategi greedy digunakan untuk mengambil keputusan dalam setiap langkah permainan berdasarkan informasi yang diperoleh dari arena. Berikut adalah pemetaan elemen algoritma greedy:

Himpunan Kandidat, Himpunan Solusi, Fungsi Solusi, Fungsi Kelayakan, Fungsi Objektif sama dengan penjelasan secara general pada **3.1 Elemen Strategi Greedy pada Robocode**

Fungsi Seleksi

Bot memilih tindakan yang memberikan keuntungan terbesar dalam situasi tertentu dengan mempertimbangkan faktor-faktor berikut:

d. **Menembak dengan Efisiensi Energi**
e. **Menghindari Serangan Musuh**
f. **Menyesuaikan Posisi Secara Optimal**

Heuristik Greedy: Strategi Dinamis Berbasis Situasi

Bot menyesuaikan perilaku berdasarkan posisi, jarak musuh, dan tingkat energi untuk mencapai keseimbangan antara agresivitas dan pertahanan.

1. Menembak Secara Efisien

- Jika musuh berada dalam jarak dekat (<100 unit), bot menembak dengan kekuatan penuh untuk meningkatkan peluang mengenai target.
- Pada jarak menengah (100 - 200 unit), bot menyesuaikan kekuatan tembakan agar lebih hemat energi.
- Jika musuh terlalu jauh (>200 units), bot lebih fokus pada pergerakan sebelum menyerang.

2. Menghindari Serangan Musuh

- Jika musuh menembak (terdeteksi dari perubahan energi), bot segera mengubah arah untuk menghindari peluru.
- Bot sering mengubah kecepatan dan arah secara acak untuk menghindari pola tembakan musuh.
- Jika berada dalam jarak bahaya (<200 unit), bot segera menjauh atau mencari posisi aman.

3. Menyesuaikan Posisi Secara Optimal

- Bot memprediksi posisi musuh berdasarkan kecepatan dan arah gerak mereka sebelum menembak.
- Jika energi rendah, bot menghindari konfrontasi langsung dan memprioritaskan pertahanan.
- Bot hanya menyerang jika posisi menguntungkan, misalnya saat musuh bergerak ke arah yang sulit untuk menghindar.

Analisis Efisiensi dan Efektivitas Alternatif Solusi

Berikut adalah analisis efisiensi dan efektivitas berbagai pendekatan yang bisa diterapkan dalam **CisliMafiaBossLv100**:

Strategi	Keunggulan	Kelemahan	Efisiensi	Efektivitas
Menembak Secara Efisien	Memanfaatkan energi secara optimal dalam setiap serangan.	Bisa kurang agresif jika energi terlalu dijaga.	Sedang	Tinggi
Menghindari Serangan Musuh	Meminimalkan kerusakan dan meningkatkan daya tahan bot.	Bisa terlalu pasif dan sulit melakukan serangan yang efektif.	Tinggi	Sedang
Menyesuaikan Posisi Secara Optimal	Memastikan bot berada di posisi terbaik sebelum menembak.	Berisiko terlalu banyak bergerak dan mengurangi efisiensi tembakan.	Sedang	Tinggi

Strategi Agresif Berbasis Jarak	Memberikan damage tinggi pada musuh dalam waktu singkat.	Menghabiskan energi dengan cepat, berisiko kalah dalam pertempuran panjang.	Rendah	Tinggi
Strategi Bertahan dan Konservatif	Memungkinkan bot bertahan lebih lama dalam pertempuran.	Bisa kalah karena terlalu pasif dan kurangnya damage terhadap lawan.	Tinggi	Rendah
Strategi Sniping Berbasis Prediksi	Akurasi tembakan lebih tinggi, serangan lebih efektif pada musuh yang bergerak.	Kurang efektif jika prediksi meleset atau lawan sering mengubah arah / strafing.	Sedang	Tinggi

BAB IV

Implementasi dan pengujian

Penjelasan struktur data, fungsi, dan prosedur yang digunakan pada bot dengan solusi greedy yang dipilih dalam komentar.

4.1 Aplikasi Strategi Greedy pada CisliCrookLv10

Penjelasan struktur data, fungsi, dan prosedur yang digunakan ada dalam comment yang di-highlight kuning.

```
program CisliCrookLv10


KAMUS GLOBAL
    int: turnCounter, TargetSpeed, TurnRate {hanya sebuah variable, kecepatan maju robot, kecepatan putar robot}
  double: Direction {Arah yang robot hadap}
  DirectionTo(x, y) → double {Arah dari robot ke input}
  BearingTo(x, y) → double {Arah bearing robot ke input}
  NormalizeRelativeAngle(x) → double {Menormalizekan angle input}
  DistanceTo(x, y) → double {Mencari jarak dari robot ke titik input}
  SetTurnGunRight(x) {Memutarkan Gun ke arah kanan}
  SetTurnGunLeft(x) {Memutarkan Gun ke arah kiri}
  Fire(x) {Fungsi untuk menembak}
  Go() {Untuk Menjalankan robot}
```

```
{Prosedur utama}

procedure Main()

  Kamus Lokal


  Algoritma

    set bot colors {Menentukan warna robot}

    do

      SetTurnGunRight(∞) {Mutar Gun selamanya}


          {If else digunakan untuk pergerakan robot. Robot akan maju atau mundur berdasarkan
turnCounter, untuk strafing}

      if(turnCounter MOD 40 = 0) then

        TargetSpeed = 6

      else if (turnCounter MOD 40 = 20) then

        TargetSpeed = -6

      turnCounter++ {}

      {Robot mulai jalan}

      Go()

    while (true)




{Sebuah function untuk mendapatkan fire power yang tepat. Fire power akan dihitung
berdasarkan seberapa jauh inputnya, tambah jauh maka akan return nilai yang lebih kecil. Ada
kasus khusus, jika energi rendah maka akan return nilai yang sangat kecil untuk
memprioritaskan hidup selama mungkin}

function getFirePower(double x, double y) –> double
```

```
  Kamus Lokal
    double: distance


  Algoritma
    distance ← DistanceTo(x, y)
    if(Energy < 10) then → 0.1
    if(distance < 100) then → 3
    else if(distance <200) then → 2.5
    else if(distance <300) then → 2
    else if(distance <400) then → 1.5
    else if(distance <500) then → 1
    else → 0.5



{Prosedur untuk melakukan sesuatu jika ada robot yang berada dalam scanner}
procedure OnScannedBot(input ScannedBotEvent e)
  Kamus Lokal
    double: angleBodyToEnemy, angleToEnemy, radarTurn, bulletPower
    getFirePower(double x, double y) → double
  Algoritma
     angleBodyToEnemy ← (360 + Direction - DirectionTo(e.X, e.Y)) MOD 360 {Arah robot ke scanner atau robot musuh}
    angleToEnemy ← Direction + BearingTo(e.X, e.Y) {Arah ke robot musuh}
        radarTurn ← NormalizeRelativeAngle(angleToEnemy - RadarDirection) {Arah yang dipelukan radar untuk bergerak ke arah musuh}
```

```
bulletPower ← getFirePower(e.X, e.Y)


{Handling supaya tracking lebih bagus}
if(radarTurn < 0) then radarTurn ← radarTurn - 10
else radarTurn ← radarTurn + 10


SetTurnGunLeft(radarTurn) {Memutarkan Gun ke arah musuh}
Fire(bulletPower)


{if else untuk membuat badan robot membuat 90 derajat dengan Gun}
if(angleBodyToEnemy < 360 AND angleToBodyEnemy > 275) then TurnRate ← -2
else if(angleBodyToEnemy < 265 AND angleToBodyEnemy > 180) then TurnRate ← 2
else if(angleBodyToEnemy < 180 AND angleToBodyEnemy > 95) then TurnRate ← -2
else if(angleBodyToEnemy < 85 AND angleToBodyEnemy > 0) then TurnRate ← -2
else  TurnRate ← 0
```

4.2 Aplikasi Strategi Greedy pada CisliCrookLv1

Penjelasan struktur data, fungsi, dan prosedur yang digunakan ada dalam comment yang di-highlight kuning.

```
program CisliCrookLv1
 { Program bot level 1 untuk Robocode Tank Royale }

KAMUS GLOBAL
 int turnCounter = 0
 float lastNearestDistance = ∞
 int lastNearestBotId = -1
 int scanCounter = 0

{ jalankan bot }
procedure Main()
    Buat objek bot CisliCrookLv1
    Jalankan bot

{ ubah trajektori perjalanan berdasarkan turn }
procedure Run()
    Set warna bot: merah, putih, dan biru
    Set turnCounter = 0
    Set GunTurnRate = 15

    while IsRunning:
```

```
        if turnCounter % 128 = 0 then
            SetTurnRate(10)
            SetTargetSpeed(5)
        if turnCounter % 128 = 64 then
            SetTurnRate(-10)
            SetTargetSpeed(5)
        turnCounter ← turnCounter + 1
        Go()


{ saat kena scan, simpan lalu atur fire power dan tembak }
procedure OnScannedBot(input ScannedBotEvent e)
    distance ← sqrt((e.X - X)^2 + (e.Y - Y)^2)
    botSpeed ← abs(e.Speed)
    firePower ← 0

    if distance < lastNearestDistance or scanCounter >= 5 then
        lastNearestDistance ← distance
        lastNearestBotId ← e.ScannedBotId
        scanCounter ← 0
    else
        scanCounter ← scanCounter + 1

    if e.ScannedBotId = lastNearestBotId then
        if not (distance > 500 and botSpeed > 4) then
```

```
        if distance < 100 then
            firePower ← 3
        else if distance < 300 then
            firePower ← 2
        else
            firePower ← 1

        if firePower > 0:
            Fire(firePower)

{ saat kena scan, simpan lalu atur fire power dan tembak }
procedure OnHitByBullet(input HitByBulletEvent e)
    SetTurnRate(5)

procedure OnHitWall(input HitWallEvent e)
    SetTargetSpeed(-1 * TargetSpeed)
```

4.3 Aplikasi Strategi Greedy pada CisliMafiaLv50

Penjelasan struktur data, fungsi, dan prosedur yang digunakan ada dalam comment yang di-highlight kuning.

```
program CisliMafiLv50
 { Program bot level 50 untuk Robocode Tank Royale }

KAMUS GLOBAL
 int turnDirection = 1
 int turnCounter = 0

{ jalankan bot }
procedure Main()
    Buat objek bot CisliMafiLv50
    Jalankan bot

{ saat bot berjalan ubah trajektori perjalanan supaya lebih sulit ditebak dengan mengubah turnRate dan target speed }
procedure Run()
    Set warna bot: merah, oranye, kuning, hijau, biru, dan nila
    Set turnCounter = 0
    repeat
       if RadarTurnRemaining = 0.0 then
          SetTurnRadarRight(∞)
```

```
        if turnCounter % 128 = 0 then
            TurnRate ← 10
            TargetSpeed ← 5
        else if turnCounter % 128 = 64 then
            TurnRate ← -10
            TargetSpeed ← 5

        MaxSpeed ← 8
        turnCounter ← turnCounter + 1
        Go()
    until false

{ saat scanning ubah arah dan tembak sesuai fire power }
procedure OnScannedBot(input ScannedBotEvent e)
    int firePower
    double distance ← DistanceTo(e.X, e.Y)
    double angleToEnemy ← Direction + BearingTo(e.X, e.Y)
    double radarTurn ← NormalizeRelativeAngle(angleToEnemy - RadarDirection)

    double extraTurn ← 5

    if radarTurn < 0 then
        radarTurn ← radarTurn - extraTurn
    else
```

```
        radarTurn ← radarTurn + extraTurn


    SetTurnRadarLeft(radarTurn)
    TurnToFaceTarget(e.X, e.Y)


    if distance < 100 then
        firePower ← 3
    else if distance < 300 then
        firePower ← 2
    else
        firePower ← 1


    Fire(firePower)


{ ubah arah dengan set turn left }
procedure TurnToFaceTarget(input double x, input double y)
    double bearing ← BearingTo(x, y)
    if bearing >= 0 then
        turnDirection ← 1
    else
        turnDirection ← -1
    SetTurnLeft(bearing)
{ langsung balik arah saat kena wall }
procedure OnHitWall(input HitWallEvent e)
```

```
TargetSpeed ← -1 * TargetSpeed
```

4.4 Aplikasi Strategi Greedy pada CisliMafiaBossLv100

Penjelasan struktur data, fungsi, dan prosedur yang digunakan ada dalam comment yang di-highlight kuning.

```
program CisliMafiaBossLv100
{ program mafia boss lv 100 }

KAMUS GLOBAL
    float enemyX, enemyY, enemyDistance
    float enemyEnergy, lastEnemyEnergy= 100
    Random random
    int moveDirection = 1
    int wallHitCount= 0
    float WALL_MARGIN = 100
    float DANGER_DISTANCE= 200

{ lakukan radar scanning dan bergerak dengan teknik (jalan sambil scanning dahulu) }
procedure Main()
    set bot colors
    set independent movement for gun, radar, and body
    while IsRunning:
        SetTurnRadarRight(∞)
        ExecuteMovement()
        Go()
```

```
{ hindari tabrakan dengan wall }

procedure ExecuteMovement()

 if IsNearWall():

        EvadeWall()

 else:

        EnhancedMovement()


{ Pergerakan yang baik dekat ke dinding supaya tidak mudah ditembak,

Jika energi musuh berkurang, maka anggap mereka menembak/ menerima tembakan jadi lakukan pergerakan untuk menghindarinya

Jika masih dalam jarak yang cukup dekat dengan musuh, hindari supaya tidak di “ram” yang mengurangi nyawa

Putari musuh supaya tidak mudah disasar }

procedure EnhancedMovement()

    if enemyDistance = 0:

        StayNearWalls()

   else:

        if lastEnemyEnergy > enemyEnergy and

        (lastEnemyEnergy - enemyEnergy) <= 3.0 and

        (lastEnemyEnergy - enemyEnergy) >= 0.1:

            EvadeBullet()

        elif enemyDistance < DANGER_DISTANCE:

            RunAway()

        else:

            OrbitEnemy()
```

```
            lastEnemyEnergy ← enemyEnergy


{ Jika dekat dinding kembalikan true }
function IsNearWall() → boolean
    -> X < WALL_MARGIN or Y < WALL_MARGIN or
       X > ArenaWidth - WALL_MARGIN or Y > ArenaHeight - WALL_MARGIN


{ Lakukan putaran (dipanggil saat dekat dinding) }
procedure EvadeWall()
   output("Evading Wall")
   SetTurnLeft(BearingTo(ArenaWidth / 1.5, ArenaHeight / 1.5))
   SetForward(100)


{ Jika sudah terlalu dekat jauhi dinding }
procedure StayNearWalls()
   if IsNearWall():
      EvadeWall()
{ hindari dinding jika dekat dinding
Jika tidak lakukan pergerakan yang berubah supaya sulit ditebak }
procedure EvadeBullet()
 if IsNearWall():
         EvadeWall()
 else:
         angleToEnemy ← BearingTo(enemyX, enemyY)
```

```
        evasionDirection ← if random.NextDouble() > 0.5 then 1 else -1
        SetTurnLeft(angleToEnemy + 90 * evasionDirection)
        SetForward(min(150, 300000 / (enemyDistance * enemyDistance + 1)))


{ hindari dinding jika dekat dinding
Jika tidak kabur dari musuh }
procedure RunAway()
 if IsNearWall():
        EvadeWall()
 else:
        angleToEnemy ← BearingTo(enemyX, enemyY)
        SetTurnLeft(angleToEnemy + 180)
        SetForward(150)




{ hindari dinding jika dekat dinding
Jika tidak lakukan pergerakan yang akan membuat kita berputar mengelilingi musuh }
procedure OrbitEnemy()
   if IsNearWall():
        EvadeWall()
   else:
      angleToEnemy ← BearingTo(enemyX, enemyY)
      SetTurnLeft(angleToEnemy + 90 * moveDirection)
```

```
        SetForward(150)
        if random.NextDouble() < 0.05:
            moveDirection ← moveDirection * -1


{ Lakukan prediksi untuk continuous scanning dan
memanfaatkan fungsi yang sudah dibuat di bawah untuk menembak }
procedure OnScannedBot(input ScannedBotEvent e)
   enemyX ← e.X
   enemyY ← e.Y
   enemyDistance ← DistanceTo(e.X, e.Y)
   enemyEnergy ← e.Energy
    angleToEnemy ← Direction + BearingTo(e.X, e.Y)
    radarTurn ← NormalizeAngle(angleToEnemy - RadarDirection)
    radarTurn ← radarTurn + if radarTurn < 0  then  -15  else  15
    SetTurnRadarLeft(radarTurn)
    speed ←  if  e.Energy = 0  then  0  else  e.Speed
    PredictEnemyAndFire(e.X, e.Y, speed, e.Direction, enemyDistance)


{ Bot bereaksi saat menabrak dinding dengan mundur, berbelok, dan kembali ke tengah jika
terlalu sering menabrak }
procedure OnHitWall()
            SetBack(50)
            SetTurnRight(90)
            wallHitCount ← wallHitCount + 1
```

```
        if wallHitCount > 2:
            SetTurnLeft(BearingTo(ArenaWidth / 1.5, ArenaHeight / 1.5))
            SetForward(200)
            wallHitCount ← 0



{ Menormalkan sudut agar berada dalam rentang -180° hingga 180° }
function NormalizeAngle(input angle)
        while angle > 180:
            angle ← angle - 360
        while angle < -180:
            angle ← angle + 360
        return angle



{ Memprediksi posisi masa depan musuh berdasarkan arah dan kecepatan, lalu menembak jika akurat }
procedure PredictEnemyAndFire(input x, y, speed, heading, distance)
        bulletPower ← if distance < 100 then 3.0
                    else if distance < 200 then 2.0
                    else if distance < 400 then 1.0
                    else 0.5
        bulletSpeed ← 20 - (3 * bulletPower)
        time ← (distance * 0.7) / bulletSpeed
```

```
futureX ← x + SIN(heading) * speed * time * 0.7
futureY ← y + COS(heading) * speed * time * 0.7
gunBearing ← GunBearingTo(futureX, futureY)
SetTurnGunLeft(gunBearing)
if ABS(gunBearing) < 5:
        Fire(bulletPower)
```

4.5 Pengujian

CisliMafiaBossLv100 vs CisliMafiaLv50

Results for 10 rounds

Rank	Name	Total Score	Survival	Surv. Bonus	Bullet Dmg.	Bullet Bonus	Ram Dmg.	Ram Bonus	1sts	2nds
1	CisliMafiaBossLv100 2.0	1664	450	110	852	155	96	0	9	1
2	CisliMafiaLv50 1.0	439	0	20	200	0	180	39	1	9

Hal ini disebabkan oleh pergerakan dari Mafia Boss yang lebih sulit untuk ditebak sehingga Crook Lv 10 yang tidak menerapkan heuristik prediksi untuk penembakan lebih boros energi dan menyebabkan dia kalah dahulu.

CisliMafiaBossLv100 vs CisliCrookLv10

Results for 10 rounds

Rank	Name	Total Score	Survival	Surv. Bonus	Bullet Dmg.	Bullet Bonus	Ram Dmg.	Ram Bonus	1sts	2nds
1	CisliMafiaBossLv100 2.0	1341	300	60	736	84	161	0	6	4
2	CisliCrookLv10 1.0	985	200	40	515	62	132	35	4	6

Strategi greedy dari mafia boss bisa mengalahkan Crook Lv 10

Hal ini disebabkan oleh pergerakan dari Mafia Boss yang lebih sulit untuk ditebak sehingga Crook Lv 10 yang tidak menerapkan heuristik prediksi untuk penembakan lebih boros energi dan menyebabkan dia kalah dahulu.

CisliMafiaBossLv100 vs CisliCrookLv1

Results for 10 rounds

Rank	Name	Total Score	Survival	Surv. Bonus	Bullet Dmg.	Bullet Bonus	Ram Dmg.	Ram Bonus	1sts	2nds
1	CisliMafiaBossLv100 2.0	807	250	50	448	36	23	0	6	3
2	ChisliCrookLv1 1.0	655	150	30	398	51	25	0	3	6

Strategi greedy dari mafia boss bisa mengalahkan Crook Lv 1

Hal ini disebabkan oleh pergerakan dari Mafia Boss yang lebih sulit untuk ditebak sehingga Crook Lv 1 yang tidak menerapkan heuristik prediksi untuk penembakan lebih boros energi dan menyebabkan dia kalah dahulu.

CisliMafiaLv50 vs CisliCrookLv10

Results for 10 rounds

Rank	Name	Total Score	Survival	Surv. Bonus	Bullet Dmg.	Bullet Bonus	Ram Dmg.	Ram Bonus	1sts	2nds
1	CisliMafiLv50 1.0	1078	300	60	388	64	229	37	6	3
2	CisliCrookLv10 1.0	589	100	20	323	20	126	0	3	6

Strategi greedy dari mafia Lv 50 bisa mengalahkan Crook Lv 1

Hal ini disebabkan oleh pergerakan dari Mafia Lv 50 lebih hemat dalam penggunaan energi. Bot Mafia Lv 50 tidak menembak jika jarak lebih dari 500. Sedangkan Crook Lv 10 terus menerus menembak di jarak jauh dan tembakannya tidak kena, menguras energinya sendiri

CisliMafiaLv50 vs CisliCrookLv1

Results for 10 rounds

Rank	Name	Total Score	Survival	Surv. Bonus	Bullet Dmg.	Bullet Bonus	Ram Dmg.	Ram Bonus	1sts	2nds
1	CisliMafiaLv50 1.0	1326	250	50	692	118	215	0	4	5
2	CisliCrookLv1 1.0	1154	150	30	770	22	144	37	5	4

Strategi greedy dari mafia Lv 50 bisa mengalahkan Crook Lv 1

Hal ini disebabkan oleh pergerakan dari Mafia Lv 50 lebih hemat dalam penggunaan energi. Bot Mafia Lv 50 tidak menembak jika jarak lebih dari 500. Sedangkan Crook Lv 1 terus menerus menembak di jarak jauh dan tembakannya tidak kena, menguras energinya sendiri

CisliCrookLv10 vs CisliCrookLv1

Results for 10 rounds

Rank	Name	Total Score	Survival	Surv. Bonus	Bullet Dmg.	Bullet Bonus	Ram Dmg.	Ram Bonus	1sts	2nds
1	CisliCrookLv10 1.0	498	200	40	199	26	32	0	4	1
2	ChisliCrookLv1 1.0	367	50	10	250	24	32	0	1	4

Strategi greedy dari Crook lv 10 bisa mengalahkan Crook Lv 1

Hal ini disebabkan oleh pergerakan dari Crook Lv 10 yang bergerak maju mundur sehingga banyak dari tembakan Crook Lv 1 tidak kena.

BAB V

Kesimpulan dan Saran

5.1 Kesimpulan

Implementasi algoritma greedy dalam Robocode sangat sesuai karena permainan ini menuntut keputusan cepat dan optimal di setiap giliran. Setiap aksi yang diambil harus memberikan keuntungan langsung, seperti menghindari peluru, menembak dengan akurasi tinggi, atau berpindah ke posisi yang lebih menguntungkan.

Dengan menggunakan strategi greedy, bot dapat secara adaptif memilih tindakan terbaik berdasarkan kondisi saat itu, seperti menghindari pertempuran atau bergerak ke posisi yang lebih strategis. Pendekatan ini memastikan bot selalu mencari solusi terbaik dalam jangka pendek dengan harapan dapat mencapai hasil optimal dalam jangka panjang.

Meskipun strategi greedy dapat memberikan hasil yang baik dalam banyak situasi, ada keterbatasan dalam menghadapi skenario kompleks di mana keputusan jangka pendek yang optimal tidak selalu menghasilkan kemenangan dalam jangka panjang. Oleh karena itu, mengombinasikan strategi greedy dengan algoritma lain, seperti pemrosesan berbasis probabilitas atau machine learning, dapat lebih meningkatkan efektivitas bot dalam bertarung.

5.2 Saran

Berdasarkan hasil analisis dan implementasi strategi greedy dalam Robocode, terdapat beberapa saran yang dapat diterapkan untuk meningkatkan efektivitas bot dalam pertempuran:

1. **Optimasi Algoritma Prediksi Musuh**
    - Saat ini, prediksi pergerakan musuh hanya mempertimbangkan kecepatan dan arah saat ini. Penggunaan metode prediksi berbasis pola atau machine learning dapat meningkatkan akurasi tembakan atau kita bisa menyimpan alur perjalanan musuh yang kita scan terus menerus.

2. **Peningkatan Manajemen Energi**
    - Pemilihan kekuatan tembakan berdasarkan jarak dapat diperbaiki dengan mempertimbangkan kondisi energi lawan dan jumlah musuh yang tersisa di arena. Ini dapat membantu bot bertahan lebih lama tanpa kehabisan energi terlalu cepat.

3. **Perbaikan Strategi Penghindaran Peluru**
    - Saat ini, bot menghindari peluru berdasarkan perubahan energi musuh. Strategi ini dapat diperbaiki dengan menerapkan pola pergerakan yang lebih tidak terduga, seperti pergerakan zig-zag atau teknik "wave surfing" yang lebih efektif dalam menghindari serangan.

4. **Evaluasi dan Pengujian dengan Berbagai Lawan**
    - Untuk mengukur keefektifan strategi greedy, bot harus diuji melawan berbagai jenis lawan dengan gaya bermain yang berbeda. Analisis terhadap kelemahan yang muncul dapat digunakan untuk menyempurnakan strategi lebih lanjut.

Lampiran

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

Github Repository

https://github.com/yonatan-nyo/Tubes1_Cisli

Video

https://youtu.be/SaNR7uZUdDk?si=dnAsOF849uVX6PXt

Daftar Pustaka

Munir, Rinaldi. 2025. “Algoritma Greedy (Bagian 1)”. https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2024-2025/04-Algoritma-Greedy-(2025)-Bag1.pdf (Diakses pada 23 Maret 2025)