LAPORAN TUGAS BESAR #1
PENGEMBANGAN ALGORITMA UNTUK PERMAINAN TANK ROYALE

IF2211– Strategi Algoritma

Kelompok "*Lagi Lagi Juaranya*"

Dosen:

Dr. Ir. Rinaldi, M.T.

Anggota Kelompok:

Raudhah Yahya Kuddah	13122003
Juan Sohuturon Arauna Siagian	18222086
Muhammad Adli Arindra	18222089

PROGRAM STUDI TEKNIK INFORMATIKA

SEKOLAH TEKNIK ELEKTRO DAN INFORMATIKA

INSTITUT TEKNOLOGI BANDUNG

2024

Daftar Isi

Deskripsi Tugas

Robocode Tank Royale merupakan permainan berbasis pemrograman yang menantang peserta untuk merancang dan mengembangkan bot berbentuk tank yang beroperasi secara otomatis di dalam arena pertempuran. Dalam permainan ini, bot-bot tersebut akan bertarung hingga hanya tersisa satu pemenang saja. Setiap bot dikendalikan oleh kode program yang menentukan segala jenis pergerakan, pengambilan keputusan, serta strategi serangan berdasarkan sifat yang sudah ditentukan sebelumnya.

Pada tugas besar ini, mahasiswa diwajibkan untuk mengembangkan empat bot menggunakan bahasa pemrograman C# dengan menerapkan strategi Greedy. Pendekatan Greedy berfokus pada pengambilan keputusan optimal secara lokal pada setiap langkah dengan tujuan memperoleh solusi yang mendekati optimal secara keseluruhan. Dalam konteks ***Robocode Tank Royale***, strategi ini diterapkan dalam berbagai aspek, termasuk pemilihan target tembakan, pengelolaan energi, serta pergerakan taktis dalam arena pertempuran.

Kompleksitas permainan ini terletak pada keterbatasan sumber daya, seperti berkurangnya energi akibat tembakan maupun tabrakan, serta adanya batasan waktu dalam setiap giliran (*turn timeout*). Selain itu, pengambilan keputusan strategis, seperti menentukan kapan harus menyerang lawan atau menghindari serangan, menjadi faktor krusial dalam meningkatkan peluang kemenangan bot. Oleh karena itu, penerapan strategi Greedy harus mempertimbangkan keseimbangan antara risiko dan keuntungan dalam setiap tindakan yang dilakukan.

Laporan ini akan menguraikan bagaimana persoalan ini dapat dimodelkan ke dalam elemen-elemen algoritma Greedy, serta mengeksplorasi berbagai alternatif strategi yang dapat diterapkan dalam permainan. Analisis lebih lanjut akan dilakukan guna mengevaluasi efektivitas masing-masing strategi serta mengidentifikasi solusi terbaik yang dapat diimplementasikan untuk mengembangkan bot yang kompetitif.

Landasan Teori

1. Penjelasan Teori Algoritma *Greedy*

Algoritma greedy adalah kelas algoritma yang membuat pilihan optimal secara lokal di setiap langkah dengan harapan menemukan solusi optimal secara global. Algoritma ini menggunakan prosedur pemecahan masalah untuk secara bertahap membangun solusi kandidat guna mendekati optimum global, dengan memperoleh solusi optimal secara lokal yang semakin baik pada setiap tahap. Secara umum, algoritma greedy tidak dapat menjamin solusi optimal global, tetapi dapat menghasilkan solusi optimal secara lokal yang cukup baik dalam waktu yang wajar dan dengan upaya komputasi yang lebih sedikit.

Salah satu contoh penerapan algoritma greedy adalah Dijkstra's Algorithm, yang digunakan untuk menemukan jalur terpendek dari satu simpul sumber ke semua simpul lain dalam graf berbobot yang tidak memiliki bobot negatif. Algoritma ini dimulai dengan menginisialisasi jarak semua simpul sebagai ∞, kecuali simpul sumber yang bernilai 0. Pada setiap langkah, simpul dengan jarak minimum yang belum dikunjungi dipilih, kemudian jarak ke simpul-simpul tetangganya diperbarui berdasarkan bobot sisi yang menghubungkan. Proses ini diulangi hingga semua simpul telah dikunjungi atau jalur terpendek ditemukan. Dijkstra's Algorithm bekerja secara optimal karena selalu memilih simpul dengan jarak minimum secara lokal, yang juga menghasilkan solusi optimal secara global dalam graf berbobot non-negatif.

Contoh lain dari algoritma greedy adalah masalah Fractional Knapsack, yang bertujuan untuk memaksimalkan nilai total barang yang dapat dimasukkan ke dalam tas dengan kapasitas tertentu. Dalam pendekatan greedy, barang diurutkan berdasarkan rasio nilai terhadap berat secara menurun, lalu dimasukkan ke dalam tas sebanyak mungkin sesuai kapasitas yang tersedia. Jika kapasitas tas masih tersisa namun barang berikutnya tidak dapat dimasukkan sepenuhnya, maka sebagian dari barang tersebut diambil hingga kapasitas tas penuh. Pendekatan ini optimal karena dalam Fractional Knapsack, diperbolehkan untuk mengambil sebagian dari barang, sehingga memilih barang dengan rasio terbaik pada setiap langkah akan menghasilkan solusi optimal secara global.

Secara keseluruhan, algoritma greedy adalah metode yang efektif dalam menyelesaikan masalah yang memiliki sifat greedy choice property dan optimal substructure. Pada kasus seperti Dijkstra's Algorithm dan Fractional Knapsack, greedy mampu memberikan solusi optimal global dengan efisiensi tinggi. Namun, dalam beberapa kasus seperti masalah Knapsack, Travelling Salesman Problem (TSP), atau Graph Coloring, pendekatan greedy tidak selalu memberikan solusi optimal sehingga diperlukan metode lain seperti dynamic programming atau backtracking. Oleh karena itu, sebelum menerapkan algoritma greedy, penting untuk memastikan bahwa sifat masalah sesuai dengan pendekatan ini agar solusi yang dihasilkan benar-benar optimal.

2. Cara Kerja *Tank Royale*

Robocode Tank Royale adalah permainan pemrograman di mana pemain menulis kode untuk mengendalikan tank virtual yang bertarung dalam pertempuran otomatis. Nama Robocode berasal dari versi orisinal permainan ini dan merupakan singkatan dari "Robot Code." Sementara itu, nama "Tank Royale" menggambarkan mekanisme permainan di mana pertarungan berlangsung di medan perang, dan tank akan bertarung hingga tersisa satu pemenang, layaknya permainan Battle Royale. Robocode Tank Royale dirancang untuk membantu pemain mengembangkan keterampilan pemrograman sambil bersenang-senang. Setiap tank dikendalikan oleh kode yang ditulis dalam bahasa pemrograman C#. Permainan ini terdiri dari beberapa komponen utama, yaitu game server yang menjalankan simulasi pertempuran, bot clients yang berisi kode untuk mengendalikan tank, dan antarmuka pengguna (UI) yang menampilkan jalannya pertandingan secara real-time. Bot berkomunikasi dengan server menggunakan protokol berbasis JSON melalui WebSocket, mengirimkan perintah seperti bergerak, menembak, atau berputar, serta menerima informasi tentang lingkungan sekitar mereka.

Cara kerja permainan dimulai dengan pemain menulis kode untuk bot mereka dan menjalankan game server, yang akan mengatur jalannya pertandingan. Bot terus berkomunikasi dengan server untuk mengirimkan perintah dan menerima data lingkungan, seperti posisi musuh dan status energi. Setiap bot memiliki radar untuk mendeteksi lawan dan dapat mengambil keputusan berdasarkan strategi yang diprogram. Selama pertempuran berlangsung, server akan menjalankan simulasi berdasarkan perintah yang diberikan bot dalam siklus waktu tertentu, memastikan bahwa semua tindakan dilakukan secara adil. Pertandingan berakhir ketika hanya satu bot yang bertahan atau ketika batas waktu tercapai, dan hasilnya akan ditampilkan dalam UI.

Setiap bot dalam Robocode Tank Royale memiliki kontrol atas beberapa aspek, seperti gerakan tank (maju, mundur, dan berputar), menembakkan peluru, serta menggunakan radar untuk memindai area sekitar dan mendeteksi lawan. Bot juga memiliki energi terbatas yang juga sekaligus menjadi *health point* dari bot tersebut, yang setiap kali menerima serangan atau menembakkan peluru dengan kekuatan besar, energinya berkurang. Jika energi habis, bot akan dinyatakan kalah dan tersingkir dari pertandingan. Oleh karena itu, strategi yang diterapkan sangat menentukan keberhasilan bot dalam bertahan dan mengalahkan lawan.

Strategi yang bisa diterapkan dalam permainan ini sangat bervariasi, tergantung pada algoritma yang digunakan. Ada pula strategi sederhana seperti menggerakkan bot secara acak untuk menghindari tembakan lawan, menunggu lawan masuk ke dalam jangkauan serangan sebelum menembak (*sniping*), atau bahkan mengecoh lawan dengan pola gerakan yang sulit ditebak. Dengan fleksibilitas dalam pemrograman bot, Robocode Tank Royale memberikan pengalaman yang menarik bagi pemain yang ingin mengembangkan keterampilan pemrograman dan menguji strategi dalam lingkungan permainan yang kompetitif.

Aplikasi Strategi *Greedy*

Bagian ini berisikan penjelasan tentang elemen-elemennya (himpunan kandidat, himpunan solusi, fungsi solusi, fungsi seleksi, fungsi kelayakan, fungsi objektif)

1. Variasi Strategi *Greedy*

Dalam tugas ini, kami mengembangkan empat variasi bot dengan strategi yang berbeda-beda untuk permainan Robocode Tank Royale. Setiap bot dirancang dengan pendekatan unik, baik dalam pergerakan, serangan, dan juga taktik bertahan agar dapat bersaing secara efektif di medan perang. Tujuan dari variasi ini adalah untuk mengeksplorasi berbagai metode strategi dan menganalisis bagaimana masing-masing pendekatan mempengaruhi kinerja bot selama pertandingan. Keempat bot akan diuji dengan cara ditandingkan satu sama lain dalam serangkaian simulasi. Hasil pertandingan akan dievaluasi berdasarkan daya tahan bot dalam pertempuran serta efektivitasnya dalam menyerang lawan. Dari evaluasi ini, kami akan menentukan strategi mana yang paling optimal. Berikut adalah keempat bot yang telah dikembangkan:

a. Bot 1

   Pada awalnya, bot bergerak secara terus-menerus dengan kecepatan maksimum sambil berputar ke kiri dalam jumlah besar untuk menghindari serangan lawan. Strategi utamanya adalah menghindari tembakan dan bertahan selama mungkin di arena. Jika jumlah musuh tersisa satu, bot mengaktifkan strategi *ramming*, yaitu untuk terus memindai posisi musuh dan segera bergerak ke arahnya untuk menabrakkan diri. Saat bertabrakan, bot juga menembakkan peluru dengan kekuatan yang disesuaikan berdasarkan energi lawan. Strategi ini diulang terus-menerus hingga musuh berhasil dikalahkan. Dengan pendekatan ini, bot memprioritaskan kelangsungan hidup di awal pertempuran dan menjadi agresif dalam duel satu lawan satu, memastikan kemenangan dengan menggabungkan serangan fisik dan tembakan yang efektif.

b. Bot 2

   Mirip seperti Bot 1, bot ini juga menggunakan strategi bertahan di awal permainan dan beralih ke mode agresif ketika jumlah musuh berkurang. Pada awalnya, bot bergerak ke tepi arena dan bergerak sepanjang sisi-sisi arena. Dengan memanfaatkan gerakan maju dan rotasi 90 derajat, bot mencoba untuk tetap bergerak sambil memindai lawan. Jika jumlah musuh tersisa kurang dari tiga, bot mengaktifkan mode agresif, yang artinya ia berhenti melakukan pola gerakan awal dan

mulai berfokus pada pertempuran langsung. Dalam mode ini, bot lebih sering berputar dan menyesuaikan posisi tembakan terhadap lawan. Saat mendeteksi musuh, bot bergerak mendekati target dan menyerang dengan kekuatan tembakan yang disesuaikan berdasarkan energi lawan. Jika terjadi tabrakan dengan musuh sebelum mode agresif, bot mundur atau maju untuk menghindari posisi yang menguntungkan lawan. Namun, dalam mode agresif, bot langsung mengarahkan meriamnya ke lawan dan menembak sebelum mendekat lebih jauh untuk memberikan tekanan. Pendekatan ini memastikan bot memiliki ketahanan tinggi di awal permainan dan menjadi lebih ofensif ketika jumlah musuh semakin sedikit, memaksimalkan peluang bertahan hingga akhir pertempuran.

c. Bot 3 - SimpleMan

   Bot ini mengandalkan komputasi matematika untuk melakukan *tracking* pada lawan. Hal ini mengakibatkan sekalinya radar mengunci lawan, radar akan selalu mengikuti pergerakan lawan dengan prediksi sudut berdasarkan kecepatan dan arah dari bot yang sedang dipindai. Setelah bot mengunci suatu lawan, bot akan mendekat untuk kemudian mulai menembak ketika jarak antara bot dan lawan sudah cukup dekat.

d. Bot 4 - Jago1

   Sama seperti SimpleMan, Jago1 mengandalkan kemampuan untuk memindai lawan. Namun, diberikan juga kalkulasi *firepower* berdasarkan kondisi bot lawan. Hal ini dikarenakan ada kondisi-kondisi khusus yang menguntungkan jika menggunakan *firepower* tertentu. Misalnya ketika jarak cukup dan selisih sudut antara bot lawan dan bot pemain cukup kecil (mengarah ke arah yang sama), maka menembakkan peluru secara cepat akan memiliki potensi mengenai lawan. Jika lawan sedang berhenti / kecepatan kecil juga memungkinkan pemain untuk menembakkan peluru yang berat.

2. Analisis Strategi

Keempat bot ini memiliki pendekatan yang berbeda dalam bertarung, masing-masing dengan kelebihan dan kekurangan yang unik. **Bot 1** mengutamakan kelangsungan hidup dengan menghindari serangan di awal permainan. Pola gerakannya yang terus berputar dengan kecepatan tinggi membuatnya sulit terkena tembakan lawan. Namun, strategi ini memiliki kelemahan karena pola gerakan yang bisa diprediksi oleh bot dengan sistem targeting canggih. Saat tersisa satu lawan, Bot 1 mengubah pendekatan menjadi agresif dengan strategi ramming dan tembakan yang disesuaikan. Pendekatan ini efektif melawan lawan yang

tidak cukup gesit untuk menghindari tabrakan, tetapi bisa dikalahkan oleh bot dengan sistem tracking dan firepower yang lebih baik.

Bot 2 memiliki strategi yang mirip dengan Bot 1 dalam hal bertahan di awal permainan, tetapi dengan pendekatan yang lebih terstruktur. Dengan bergerak di sepanjang tepi arena, bot ini berusaha mengurangi kemungkinan diserang dari berbagai arah. Mode agresif yang diaktifkan saat jumlah musuh berkurang menunjukkan adaptasi terhadap situasi pertempuran. Namun, pergerakan di tepi arena bisa menjadi pola yang mudah diprediksi dan dapat dimanfaatkan oleh bot dengan prediksi tembakan yang akurat. Meskipun begitu, kemampuan bot ini untuk berubah menjadi agresif saat situasi memungkinkan membuatnya lebih fleksibel dibandingkan Bot 1.

Berbeda dari kedua bot sebelumnya, **Bot 3 (SimpleMan)** lebih berorientasi pada sistem tracking yang mengunci pergerakan lawan. Dengan menggunakan prediksi berbasis kecepatan dan arah musuh, bot ini mampu menargetkan lawan dengan lebih akurat dibandingkan Bot 1 dan Bot 2. Namun, fokus utamanya adalah mendekati musuh sebelum menyerang, yang membuatnya rentan terhadap bot yang memiliki pola gerakan tidak terduga atau yang mampu menyerang dari jauh sebelum SimpleMan bisa mendekat. Jika lawan memiliki strategi penghindaran yang baik atau melakukan gerakan zig-zag, prediksi tembakan bot ini bisa menjadi kurang efektif.

Sementara itu, **Bot 4 (Jago1)** mengombinasikan tracking lawan dengan optimasi firepower. Dengan mempertimbangkan kondisi lawan sebelum menembak, bot ini dapat menyesuaikan serangannya untuk memaksimalkan efektivitas. Jika lawan berada dalam posisi yang menguntungkan untuk ditembak, Jago1 akan menggunakan peluru cepat, sedangkan jika lawan bergerak lambat atau diam, bot ini akan menggunakan peluru dengan kekuatan lebih besar. Strategi ini menjadikannya bot yang paling cerdas dalam hal manajemen serangan. Namun, seperti SimpleMan, keefektifan Jago1 bergantung pada akurasi tracking-nya. Jika lawan memiliki pola gerakan acak atau strategi penghindaran yang baik, Jago1 bisa kesulitan mendaratkan tembakan yang akurat.

Dalam perbandingan langsung, **Jago1 memiliki keunggulan dalam strategi serangan yang disesuaikan, membuatnya lebih efektif dalam pertarungan jangka panjang dibandingkan bot lainnya**. Namun, **Bot 1 memiliki daya tahan yang lebih baik di awal pertempuran** dan bisa bertahan lebih lama jika lawan tidak mampu mengenainya. **Bot 2 unggul dalam fleksibilitas strategi** dengan peralihan dari defensif ke agresif, sedangkan **Bot 3 memiliki sistem tracking yang kuat tetapi rentan terhadap lawan dengan pergerakan tidak terduga**. Jika keempat bot bertarung dalam satu arena, Jago1

memiliki peluang terbaik dalam serangan jangka panjang, tetapi Bot 1 bisa memenangkan pertarungan terakhir dengan strategi ramming jika bertahan hingga akhir.

Implementasi dan Pengujian

1. Implementasi Solusi Alternatif

Bagian ini berisikan *pseudo-code* untuk masing-masing bot yang telah dikembangkan, dengan tujuan menjelaskan strategi yang diterapkan dalam pergerakan, serangan, dan pertahanan tanpa menampilkan kode asli. Pseudo-code ini memberikan gambaran umum tentang bagaimana setiap bot mengambil keputusan selama pertempuran, seperti cara mereka mendeteksi lawan, menghindari serangan, atau menentukan waktu yang tepat untuk menembak. Berikut ini adalah *pseudo-code* untuk keempat bot yang telah dikembangkan:

a. Bot 1

```
running := true
agresif := false

while (running) do {
    if (enemyCount == 1) then agresif = true;
    if (agresif) then {
        TurnLeft()
    } else {
        Forward()
        TurnLeft()
    }

    if (botScanned) then {
        if (agresif) then {
            TurnToEnemy()
            Forward()
        } else {
            Fire()
        }
    }

    if (botRammed) then {
        if (agresif) then {
            TurnToEnemy()
            Forward()
            Fire()
        } else {
            TurnLeft()
        }
    }
}
```

b. Bot 2

```
running := true
agresif := false

TurnGunRight()
TurnRight()

while (running) do {
    if (enemyCount < 3) then agresif = true;

    if (agresif) then {
         TurnGunToFront()
         TurnLeft()
    } else {
         Forward()
         TurnRight()
    }

    if (botScanned) then {
         if (agresif) then {
              TurnToEnemy()
              Forward()
         } else {
              Fire()
         }
    }

    if (botRammed) then {
         if (agresif) then {
              TurnToEnemy()
              Forward()
              Fire()
         } else {
              if (facingEnemy) then Backward()
              else Forward()
         }
    }
}
```

c. Bot 3 - SimpleMan

```
running := true
agresif := false

TurnGunRight()
TurnRight()

while (running) do {
        if (enemyCount < 3) then agresif = true;

        if (agresif) then {
          TurnGunToFront()
          TurnLeft()
          } else {
          TurnLeft(2)
        }

        if (botScanned) then {
          foundEnemy = true
          enemyX = scannedBot.X
          enemyY = scannedBot.Y
          enemyDir = scannedBot.Direction
          enemySpeed = scannedBot.Speed

          if (agresif) then {
                  TurnToEnemy()
                  Forward()
          } else {
                  Fire(1)
          }
        }

        if (foundEnemy) then {
          newEnemyX = enemyX + cos(enemyDir)
          newEnemyY = enemyY + cos(enemyDir)
          TurnLeft(BearingTo(newEnemyX, newEnemyY))

          if (DistanceTo(enemyX, enemyY) >= 125) then {
                  Forward(10)
          } else {
               Backward(10)
          }
          foundEnemy = false
        }

        Go()
}
```

d. Bot 4 - Jago1

```
running := true
agresif := false
randomizer := Random()

TurnGunRight()
TurnRight()

while (running) do {
        if (enemyCount < 3) then agresif = true

        if (agresif) then {
          TurnGunToFront()
          TurnLeft()
        } else {
          Hunting()
          ClearEnemyMemory()
          TurnLeft(2)
        }

        if (botScanned) then {
          foundEnemy = true
          enemyX = scannedBot.X
          enemyY = scannedBot.Y
          enemyDir = scannedBot.Direction
          if (enemySpeed != -11) then
                enemyDeltaSpeed = scannedBot.Speed - enemySpeed
          enemySpeed = scannedBot.Speed
        }

        if (foundEnemy) then {
          newEnemyX = enemyX + cos(enemyDir) * (enemyDeltaSpeed + enemySpeed)
          newEnemyY = enemyY + sin(enemyDir) * (enemyDeltaSpeed + enemySpeed)
          TurnLeft(BearingTo(newEnemyX, newEnemyY))
          Forward(10)

          enemyDir1 = enemyDir
          enemyDir2 = (enemyDir + 180) % 360
          dirDelta1 = abs(CalcBearing(enemyDir1))
          dirDelta2 = abs(CalcBearing(enemyDir2))
          distanceToEnemy = DistanceTo(newEnemyX, newEnemyY)

          if (dirDelta1 <= 20 OR dirDelta2 <= 20) then {
                if (distanceToEnemy <= 100 AND distanceToEnemy >= 70) then Fire(3)
                else if (distanceToEnemy <= 125) then Fire(2)
                else if (distanceToEnemy <= 175) then Fire(1)
          }

          if (distanceToEnemy < 70) then {
                if (abs(enemySpeed) < 3) then Fire(10)
                else if (dirDelta1 <= 20 OR dirDelta2 <= 20) then Fire(7.5)
          }
          foundEnemy = false
        }

        Go()
}
```

```
procedure Hunting() {
        if (DistanceRemaining == 0) then
          Forward(randomizer.next(-100, 100))
        TurnLeft(2)
}

procedure ClearEnemyMemory() {
        enemyDeltaSpeed = 0
        enemyDir = -1
        enemySpeed = -11
        enemyX = -1
        enemyY = -1
}
```

2. Penjelasan Struktur Program Solusi

Program Jago1 adalah sebuah bot yang dikembangkan menggunakan Robocode Tank Royale API dalam bahasa C#. Program ini mengontrol bot dalam arena pertempuran dengan strategi berburu musuh, mendeteksi keberadaan musuh, dan menyerang menggunakan aturan tertentu berdasarkan jarak dan arah musuh.

2.1. Struktur Data yang Digunakan

a. Variabel Instance
- randomizer: Objek Random yang digunakan untuk menentukan gerakan bot secara acak.
- foundEnemy: Boolean yang menandakan apakah musuh telah terdeteksi.
- enemyDeltaSpeed: Double yang menyimpan perubahan kecepatan musuh dibandingkan dengan sebelumnya.
- enemyDir: Double yang menyimpan arah gerakan musuh dalam derajat.
- enemySpeed: Double yang menyimpan kecepatan musuh.
- enemyX, enemyY: Double yang menyimpan koordinat musuh dalam arena.

2.2 Penjelasan *Method*

a. Main()
- Method utama yang memulai eksekusi program dengan membuat objek Jago1 dan menjalankan bot menggunakan Start().

b. Jago1() (Constructor)
- Konstruktor memuat informasi bot dari file konfigurasi (Jago1.json) menggunakan BotInfo.FromFile().

c. Run() (Override dari Bot)
- Memanggil Construct() untuk inisialisasi bot.
- Loop utama berjalan selama bot masih aktif (IsRunning).
- Jika musuh ditemukan (foundEnemy == true), bot akan mengeksekusi Destroy() untuk menyerang.

- Jika tidak ada musuh, bot akan melakukan Hunting() untuk mencari musuh dan menghapus data musuh dengan ClearEnemyMemory().
- Bot juga mengubah arah dengan SetTurnLeft(2) dan menjalankan aksi dengan Go().

d. Construct()
   - Menginisialisasi randomizer untuk pergerakan acak.
   - Mengatur foundEnemy menjadi false.
   - Memanggil ClearEnemyMemory() untuk menghapus data musuh yang tersimpan sebelumnya.

e. ClearEnemyMemory()
   - Mengatur ulang informasi musuh ke nilai default seperti enemyDir = -1 dan enemySpeed = -11.

f. Destroy()
   - Memperkirakan posisi musuh berdasarkan arah dan kecepatan dengan menggunakan fungsi Math.Cos() dan Math.Sin() untuk menghitung koordinat baru.
   - Mengarahkan bot ke posisi musuh dengan SetTurnLeft(BearingTo(newEnemyX, newEnemyY)).
   - Menggerakkan bot maju dengan SetForward(10).
   - Menentukan apakah bot harus menembak berdasarkan arah dan jarak musuh. Jika musuh dalam jarak tertentu, bot akan menembakkan peluru dengan daya yang berbeda:
     - Jarak 70 - 100 → Tembakan kuat (SetFire(3))
     - Jarak 100 - 125 → Tembakan sedang (SetFire(2))
     - Jarak 125 - 175 → Tembakan lemah (SetFire(1))
     - Jika musuh sangat dekat (< 70) dan kecepatan rendah, bot menembakkan peluru maksimal (Fire(10)).
     - Jika musuh masih dalam sudut serangan, bot dapat menembak dengan kekuatan menengah (Fire(7.5)).

g. Hunting()
   - Jika bot tidak memiliki pergerakan yang tersisa (DistanceRemaining == 0), maka bot akan bergerak maju atau mundur dalam jarak acak antara -100 hingga 100.
   - Bot juga akan sedikit berbelok (SetTurnLeft(2)) untuk menghindari pola pergerakan yang mudah ditebak.

h. OnScannedBot(ScannedBotEvent e) (Event Handler)
   - Event ini dipanggil ketika bot mendeteksi musuh.
   - Bot menyimpan koordinat musuh (enemyX, enemyY), arah (enemyDir), dan kecepatan (enemySpeed).
   - Jika sudah memiliki data kecepatan sebelumnya, maka bot menghitung perubahan kecepatan (enemyDeltaSpeed).

Program Jago1 adalah bot yang beroperasi dalam arena dengan strategi sederhana. Bot ini mencari musuh dengan pola gerakan acak, mendeteksi musuh dengan event OnScannedBot(), dan kemudian menyerang menggunakan strategi berbasis jarak serta kecepatan musuh.

3. Pengujian Bot

Results for 10 rounds

Rank	Name	Total Score	Survival	Surv. Bonus	Bullet Dmg.	Bullet Bonus	Ram Dmg.	Ram Bonus	1sts	2nds	3rds
1	Jago1 1.0	1667	450	90	662	108	302	54	10	0	0
2	Crazy 1.0	300	0	0	136	0	164	0	0	10	0

Results for 10 rounds

Rank	Name	Total Score	Survival	Surv. Bonus	Bullet Dmg.	Bullet Bonus	Ram Dmg.	Ram Bonus	1sts	2nds	3rds
1	Jago1 1.0	1213	350	70	534	88	170	0	8	0	0
2	Crazy 1.0	262	0	0	172	0	90	0	0	8	0

Results for 10 rounds

Rank	Name	Total Score	Survival	Surv. Bonus	Bullet Dmg.	Bullet Bonus	Ram Dmg.	Ram Bonus	1sts	2nds	3rds
1	Jago1 1.0	1376	400	80	600	96	174	26	9	0	0
2	Crazy 1.0	206	0	0	108	0	98	0	0	9	0

Dalam total 27 pertandingan melawan bot Crazy, seluruhnya dimenangkan oleh bot Jago1.

Results for 10 rounds

Rank	Name	Total Score	Survival	Surv. Bonus	Bullet Dmg.	Bullet Bonus	Ram Dmg.	Ram Bonus	1sts	2nds	3rds
1	Jago1 1.0	1454	350	30	530	94	424	27	4	0	0
2	Spin Bot 1.0	1123	500	60	464	33	66	0	2	2	2
3	Fire 1.0	718	300	30	348	30	10	0	0	2	1
4	Crazy 1.0	697	500	30	120	0	47	0	0	2	3

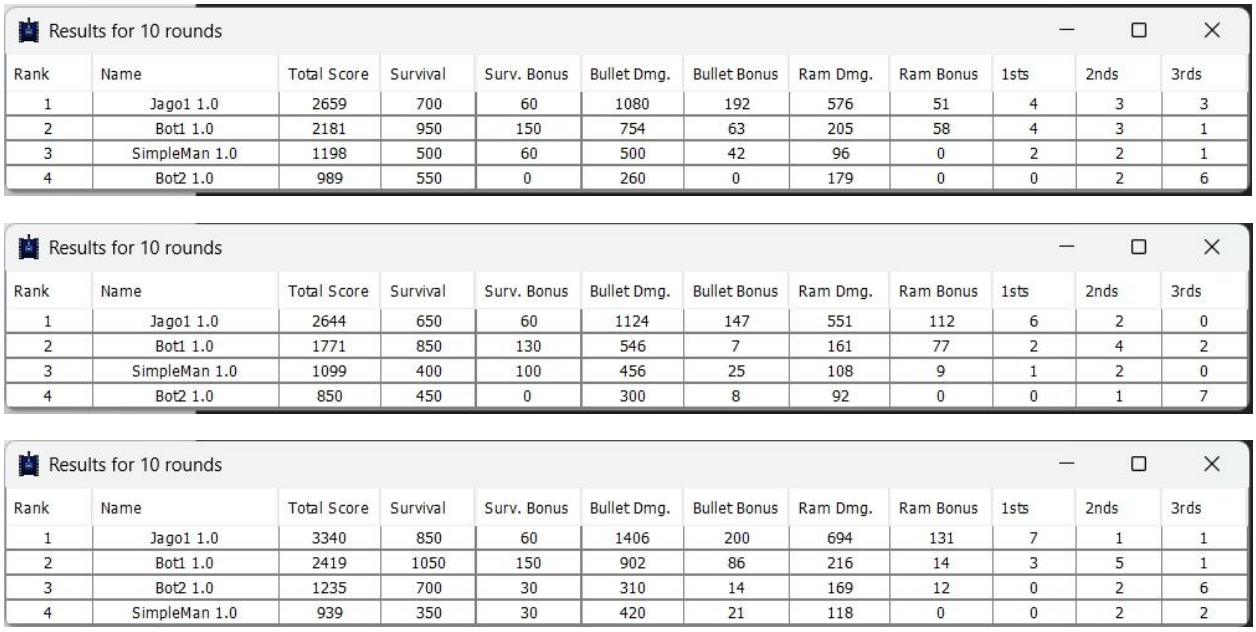
Results for 10 rounds

Rank	Name	Total Score	Survival	Surv. Bonus	Bullet Dmg.	Bullet Bonus	Ram Dmg.	Ram Bonus	1sts	2nds	3rds
1	Jago1 1.0	1354	250	0	584	40	418	61	4	0	2
2	Spin Bot 1.0	1036	450	60	400	29	77	19	2	2	0
3	Crazy 1.0	773	550	60	116	6	41	0	0	2	2
4	Fire 1.0	735	400	30	288	12	5	0	0	2	2

Results for 10 rounds

Rank	Name	Total Score	Survival	Surv. Bonus	Bullet Dmg.	Bullet Bonus	Ram Dmg.	Ram Bonus	1sts	2nds	3rds
1	Jago1 1.0	1322	300	30	512	82	346	52	4	0	0
2	Spin Bot 1.0	930	400	30	400	48	52	0	1	2	1
3	Crazy 1.0	706	450	60	160	12	23	0	0	2	2
4	Fire 1.0	428	200	0	200	21	6	0	0	1	2

Dilakukan juga 17 kali pertarungan 3 lawan 1 melawan bot Spin Bot, Crazy, dan Fire. Bot Jago1 berhasil meraih kemenangan 12 kali juara 1, 2 kali juara 3, dan 3 kali juara 4.



Results for 10 rounds

Rank	Name	Total Score	Survival	Surv. Bonus	Bullet Dmg.	Bullet Bonus	Ram Dmg.	Ram Bonus	1sts	2nds	3rds
1	Jago1 1.0	2659	700	60	1080	192	576	51	4	3	3
2	Bot1 1.0	2181	950	150	754	63	205	58	4	3	1
3	SimpleMan 1.0	1198	500	60	500	42	96	0	2	2	1
4	Bot2 1.0	989	550	0	260	0	179	0	0	2	6

Results for 10 rounds

Rank	Name	Total Score	Survival	Surv. Bonus	Bullet Dmg.	Bullet Bonus	Ram Dmg.	Ram Bonus	1sts	2nds	3rds
1	Jago1 1.0	2644	650	60	1124	147	551	112	6	2	0
2	Bot1 1.0	1771	850	130	546	7	161	77	2	4	2
3	SimpleMan 1.0	1099	400	100	456	25	108	9	1	2	0
4	Bot2 1.0	850	450	0	300	8	92	0	0	1	7

Results for 10 rounds

Rank	Name	Total Score	Survival	Surv. Bonus	Bullet Dmg.	Bullet Bonus	Ram Dmg.	Ram Bonus	1sts	2nds	3rds
1	Jago1 1.0	3340	850	60	1406	200	694	131	7	1	1
2	Bot1 1.0	2419	1050	150	902	86	216	14	3	5	1
3	Bot2 1.0	1235	700	30	310	14	169	12	0	2	6
4	SimpleMan 1.0	939	350	30	420	21	118	0	0	2	2

Dalam tiga kali pertandingan melawan ketiga bot alternatif lainnya, dapat dilihat bahwa Jago1 selalu meraih skor tertinggi dibandingkan ketiga bot lainnya.

4. Analisis Pengujian

Dalam pertandingan 1v1, hampir dapat dijamin bahwa Jago1 akan memenangkan pertandingan. Hal ini dikarenakan kemampuan Jago1 untuk melakukan *tracking* dan mengikuti pergerakan musuh sehingga bisa mendapatkan sudut-sudut yang ideal untuk menembak.

Namun, pada pertandingan 1v4 dan selainnya, terlalu banyak faktor yang perlu diperhitungkan untuk mendapatkan hasil yang optimal. Seperti peluru-peluru lawan dan bentrokan dengan bot lain. Namun, Jago1 tetap bisa menavigasi dengan tidak menghabiskan energi untuk tembakan-tembakan yang tidak terjamin

Kesimpulan dan Saran

Dapat disimpulkan bahwa strategi *greedy* dapat diterapkan untuk permainan Robocode Tank Royale. Namun, diperlukan kemampuan matematika dan analisis untuk bisa menentukan komponen-komponen yang ingin diterapkan secara *greedy*. Diperlukan juga perumusan strategi di awal yang nantinya akan diimplementasikan secara *greedy*.

Hal yang sering dilupakan ketika menerapkan algoritma *greedy* adalah *greedy* tidak hanya berarti rangkaian if-else, *greedy* adalah alur / prosedur yang akan diterapkan untuk menghadapi suatu masalah. Dan semakin banyak kasus yang ditangani oleh algoritma, maka semakin baik juga performa program untuk menghadapi masalah yang diberikan.

Lampiran

Tautan GitHub