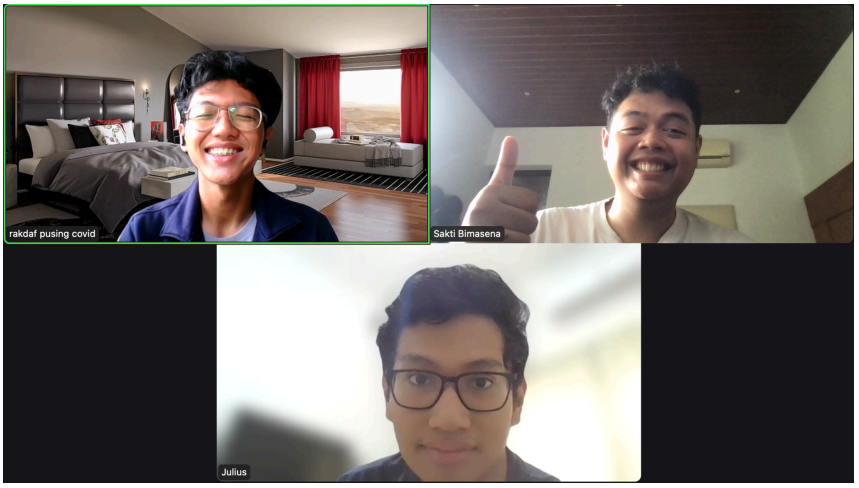
LAPORAN TUGAS BESAR 1

IF2211 STRATEGI ALGORITMA
Pemanfaatan Algoritma *Greedy* dalam pembuatan *bot* permainan Robocode Tank Royale



Disusun Oleh:

Raka Daffa Iftikhaar (13523018)

Julius Arthur (13523030)

Sakti Bimasena (13523053)

PROGRAM STUDI TEKNIK INFORMATIKA

SEKOLAH TEKNIK ELEKTRO DAN INFORMATIKA

INSTITUT TEKNOLOGI BANDUNG

JL. GANESA 10, BANDUNG 40132

2025

DAFTAR ISI

BAB I
DESKRIPSI TUGAS

Gambar 1.1 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.

BAB II
LANDASAN TEORI

2.1 Dasar Teori Algoritma Greedy

Algoritma Greedy adalah algoritma yang memecahkan persoalan secara step-by-step (langkah demi langkah). Strateginya adalah dengan mengambil pilihan terbaik sesuai strategi yang diperoleh pada saat itu tanpa memikirkan konsekuensi untuk kedepannya. Penggunaan algoritma greedy juga berharap bahwa pemilihan optimum lokal pada setiap langkah dapat menuju ke optimum global.

Langkah-langkah umum dalam algoritma greedy meliputi:

- Himpunan Kandidat, C: Berisi kandidat pemilihan setiap langkah
- Himpunan Solusi, S: Berisi kandidat yang sudah dipilih
- Fungsi Solusi: Menentukan apakah kandidat yang dipilih sudah memberikan solusi
- Fungsi Seleksi: Memilih kandidat berdasarkan strategi greedy yang ditentukan
- Fungsi Kelayakan: Memeriksa apakah kandidat yang dipilih layak atau tidak dimasukkan ke himpunan solusi
- Fungsi Obyektif: Memaksimalkan atau meminimalkan

Walaupun algoritma greedy mampu memberikan jawaban yang optimal untuk sejumlah masalah, tidak semua masalah bisa dipecahkan secara optimal dengan pendekatan ini. Dalam situasi tertentu, algoritma greedy hanya menghasilkan solusi yang hampir optimal tetapi tidak sepenuhnya optimal. Masalah Knapsack, yang mengoptimalkan pemilihan item berdasarkan nilai dan bobot, metode Dijkstra untuk menentukan jalur terpendek dalam grafik, dan Pengkodean Huffman dalam kompresi data adalah beberapa contoh bagaimana algoritma greedy digunakan dalam berbagai industri.

2.2 Cara Kerja Program

2.2.1 Cara Bot Melakukan Aksinya

Tujuan utama dari permainan Robocode Tank Royale adalah agar bot dapat bertahan lebih lama satu sama lain di arena hingga hanya tersisa satu bot. Untuk meningkatkan peluang bertahan hidup dan mendapatkan poin terbanyak, setiap bot dapat melakukan berbagai tugas selama setiap giliran. Berikut adalah beberapa tindakan yang mungkin dilakukan:

- Bergerak: Bot dapat bergerak maju atau mundur dengan kecepatan tertentu sesuai dengan strategi yang ditentukan
- Berputar: Bagian tubuh bot, meriam, dan radar dapat berputar secara independen satu sama lain untuk menyesuaikan arah atau mendeteksi musuh
- Menembak: Bot dapat menggunakan meriam untuk menyerang lawan, dengan pertimbangan bahwa setiap tembakan mengurangi energi

- Memindai: Radar bot dapat digunakan untuk memindai lokasi musuh dalam radius tertentu
- Bereaksi terhadap peristiwa: Bot dapat bereaksi terhadap tabrakan kepada dinding atau bot lain serta mengatur strategi jika tertembak oleh musuh

Setiap pilihan yang dibuat bot selama giliran ditentukan oleh informasi yang ada, termasuk posisi lawan, jumlah energi yang tersisa, dan kemungkinan serangan akan berhasil. Bot dapat memilih tindakan yang paling menguntungkan untuk meningkatkan peluang menangnya dengan memperhitungkan variabel-variabel ini.

2.2.2 Peran Algoritma Greedy dalam Bot

Pada setiap tahap pemilihan keputusan pada bot, algoritma greedy terlibat. Bot bisa dengan efisien menyesuaikan diri dengan perubahan keadaan dengan memilih tindakan yang menawarkan secara langsung solusi optimal. Dengan menggunakan algoritma greedy, bot bisa selalu mengambil keputusan jangka pendek terbaik tanpa memikirkan jangka panjang. Sebagai contoh, bot dapat menggunakan pendekatan greedy dalam memilih apakah akan menyerang, menghindar, atau berpindah tempat berdasarkan keuntungan yang paling besar pada giliran tersebut.

2.2.3 Menjalankan Bot

Untuk menjalankan bot dalam Robocode Tank Royale, langkah-langkah berikut harus dilakukan:

1. Mengimplementasikan Bot: Bot dikembangkan menggunakan bahasa pemrograman C# dengan mengacu pada mekanisme permainan yang telah ditentukan.
2. Mengkompilasi Program: Setelah kode selesai ditulis, program perlu dikompilasi menggunakan .NET agar dapat dijalankan.
3. Menjalankan Simulator: Game engine yang telah dimodifikasi oleh asisten digunakan untuk menguji performa bot dalam lingkungan simulasi.
4. Melakukan Pengujian: Bot diuji dalam berbagai skenario pertempuran untuk mengevaluasi efektivitas penerapan algoritma greedy.
5. Menganalisis Performa: Setelah pengujian dilakukan, hasil pertandingan dianalisis untuk menentukan apakah pendekatan yang digunakan sudah optimal atau masih perlu disesuaikan.

Dengan mengikuti prosedur ini, bot dapat berpartisipasi dalam kompetisi dan bersaing dengan bot lain secara optimal. Penggunaan algoritma greedy dalam implementasi bot memungkinkan keputusan cepat dan efisien dalam pertempuran, dengan tujuan utama memperoleh skor setinggi mungkin.

BAB III
APLIKASI STRATEGI GREEDY

3.1 Eksplorasi Alternatif Solusi Greedy

3.1.1 Greedy by Energy and Distance

Strategi Greedy by Energy dan Distance berfokus kepada efisiensi penggunaan energi dari bot dengan mengukur jarak dari bot kita ke bot lawan serta mengukur energi kita. Dalam pendekatan ini, bot akan menembak dan bergerak ke kanan, ke kiri, dan berputar dengan agresif terhadap setiap lawan jika memiliki energi yang cukup banyak yaitu lebih dari sepuluh poin. Bot akan menembak musuh yang ia scan dengan perkiraan jarak. Jika jarak jauh maka peluru yang ditembakkan akan berkekuatan kecil namun jika jarak sangat dekat maka peluru yang ditembakan akan berkekuatan maksimal. Jika bot memiliki energi kurang dari sepuluh poin maka bot akan bergerak berputar-putar dengan kecepatan tinggi agar sulit ditembak musuh dan untuk mengulur waktu.

Keputusan yang diambil bot pada setiap giliran mengikuti konsep greedy, yaitu memilih berdasarkan kondisi bot saat itu dimulai dari energi dan jarak bot ke musuh yang discan. Bot memilih tindakan yang dirasa paling menguntungkan dirinya saat itu tanpa memikirkan dampak jangka panjangnya seperti bagaimana jika pergerakan musuh sangat lincah dan sulit untuk ditembak namun bot tetap menembak tanpa henti yang mengakibatkan energinya habis sendiri.

A. Mapping persoalan dalam strategi ini sebagai berikut:
   - Himpunan Kandidat: Mencakup semua aksi yang dapat diambil oleh bot pada setiap round seperti bergerak, menembak, menghindar, berputar.
   - Himpunan Solusi: Mencakup semua aksi yang telah diambil selama pertandingan.
   - Fungsi Seleksi: Memilih aksi terbaik berdasarkan dua faktor utama yaitu energi bot dan jarak bot ke musuh.
   - Fungsi Kelayakan: Memilih aksi yang dipastikan dapat dilakukan dalam batasan permainan seperti energi yang cukup untuk menembak, kecepatan maksimum, dan tidak ada halangan yang menghambat pergerakan bot.
   - Fungsi Solusi: Bot akan menganggap solusi berhasil jika bisa bertahan lebih lama dengan energi rendah dari yang lain dan memberikan damage lebih tinggi dari lawan.
   - Fungsi Objektif: : Bot berusaha mencapai objektif dengan cara yang paling cepat dan sederhana dengan memaksimalkan damage, menjaga akurasi tembakan, dan meminimalkan kerugian.

B. Analisis efisiensi solusi

Strategi yang diterapkan pada bot ini sangat efisien karena diukur dari jarak antara bot dan lawan, bot akan memberikan damage yang sangat tinggi bila lawan sangat dekat. Selain itu, pergerakan bot bisa dibilang cepat dan sederhana namun sulit ditebak oleh lawan karena adanya pergerakan belok kanan dan belok kiri yang terjadi secara acak tergantung jarak. Namun, bot memiliki kekurangan dimana karena bot akan terus menembak saat melihat musuh baik itu lokasi dekat maupun jauh, maka akan banyak peluru dan energi yang terbuang sia-sia.

C. Analisis efektivitas solusi

Strategi bot ini tergolong cukup efektif bagi musuh yang memiliki pergerakan sederhana dan mudah di-track karena bot akan terus menembakkan peluru ke bot musuh dengan mudah. Namun ada beberapa kekurangan seperti bot akan kesulitan menembak musuh yang lincah dan bergerak secara acak. Selain itu juga karena pergerakan bot yang bisa dibilang cepat namun acak, bot akan sering menabrak tembok dan membuat energi dari bot akan terbuang sia-sia karena menabrak tembok itu.

3.1.2 Greedy by Damage and Accuracy

Strategi Greedy by Damage and Accuracy berfokus kepada pemaksimalan skor dengan penyerangan yang kuat dan tepat. Dalam pendekatan ini, bot akan bergerak maju dengan agresif terhadap lawan hingga pada jarak optimal lalu menembak dengan peluru kekuatan maksimal. Dengan mendekati musuh, probabilitas peluru tepat kena kepada musuh menjadi lebih tinggi, sehingga memaksimalkan jumlah damage yang dihasilkan.

Keputusan yang diambil bot pada setiap giliran mengikuti konsep greedy, yaitu memilih yang secara langsung memberikan dampak terbesar terhadap total damage yang diberikan ke lawan. Bot tidak akan mempertimbangkan dampak jangka panjang seperti penghematan energi atau serangan balik dari musuh, melainkan hanya fokus pada menghasilkan damage terbesar dalam waktu sesingkat mungkin.

A. Mapping persoalan dalam strategi ini sebagai berikut:
   - Himpunan Kandidat: Semua aksi yang dapat dilakukan bot, seperti bergerak maju, menembak, atau menghindar.
   - Himpunan Solusi: Kumpulan aksi yang sudah diambil
   - Fungsi Seleksi: Memilih aksi yang secara langsung memberikan damage tertinggi dan meningkatkan peluang serangan sukses.
   - Fungsi Kelayakan: Memastikan bahwa aksi yang dipilih memungkinkan dalam batasan permainan, seperti energi yang cukup untuk menembak dan tidak ada halangan yang menghambat pergerakan bot.

- Fungsi Solusi: Strategi dianggap berhasil jika bot mampu bertahan lebih lama dengan skor damage yang lebih tinggi dibandingkan musuh.
- Fungsi Objektif: Memaksimalkan damage yang diberikan kepada musuh dengan tetap menjaga akurasi tembakan agar peluru tidak terbuang sia-sia.

B. Analisis Efisiensi Solusi

Strategi ini sangat efisien dalam memberikan damage tinggi dalam waktu singkat karena bot langsung mendekati lawan dan menembakkan peluru dengan kekuatan maksimum. Lalu, dengan tidak mempertimbangkan pertahanan, bot dapat lebih cepat membuat keputusan. Sehingga setiap giliran dapat dimanfaatkan dengan maksimal. Namun, strategi ini bisa menjadi tidak efisien jika musuh memiliki strategi penghindaran atau pertahanan yang baik, sehingga tembakan tidak mengenai target.

C. Analisis Efektivitas Solusi

Strategi ini efektif dalam melawan bot yang diam atau pergerakannya relatif sederhana karena bot dapat menyerang lawan dengan akurasi tinggi. Ketika lawan memiliki serangan balik yang kuat, efektivitas ini dapat menurun karena bot tidak memiliki strategi pertahanan yang baik. Jika terdapat beberapa bot yang musuh yang berdekatan juga bisa menurunkan efektivitas dari bot ini karena bot tidak memiliki algoritma dalam pemilihan target terefektif sehingga bisa jadi salah memilih target.

3.1.3 Greedy by Distance and Avoidance

Strategi ini akan berusaha mencari bot dengan jarak paling dekat di sekitarnya. Hal ini dimulai dengan melakukan pemindaian ke seluruh sudut bot. Jika jarak bot yang dipindai merupakan jarak minimum saat itu, bot akan menyerang dengan kekuatan tergantung jarak tersebut. Harapannya, dengan jarak yang dekat akan dibutuhkan waktu perjalanan peluru yang minimum. Selain itu,

Strategi ini merupakan strategi greedy dengan mencari musuh paling dekat untuk ditembak dan menghindar. Bot tidak memperhatikan energi, serangan, serta kondisi lain pada bot musuh. Bot akan menargetkan lawan yang paling mudah dijangkau dan memaksimalkan kesempatan penyerangan sukses.

A. Mapping persoalan dalam strategi ini sebagai berikut:
   - Himpunan Kandidat: Semua aksi yang dapat dilakukan bot, seperti bergerak maju, menembak, atau menghindar.
   - Himpunan Solusi: Kumpulan aksi yang sudah diambil
   - Fungsi Seleksi: Memilih bot dengan jarak paling dekat
   - Fungsi Kelayakan: Bot menyerang dan menghindari bot musuh paling dekat.

- Fungsi Solusi: Bot terus bergerak dalam pola yang menghindari rintangan, menargetkan musuh terdekat, dan menyesuaikan posisi untuk menyerang atau menghindar.
- Fungsi Objektif: Memaksimalkan serangan ke musuh sekaligus meminimalkan terkena serangan

B. Analisis Efisiensi Solusi

Strategi ini efisien dalam menghadapi banyak bot, dengan mencari bot yang paling mudah dijangkau. Dengan memperpendek jarak antar bot dan lawan, waktu yang dibutuhkan peluru untuk sampai ke tujuan akan semakin pendek juga, mengurangi kemungkinan meleset. Namun, efisiensi strategi ini dapat berkurang jika musuh melakukan pergerakan acak atau menghindar. Bot musuh dapat dengan mudah bergerak dan menghindari serangan tanpa adanya respons dari strategi ini.

C. Analisis Efektivitas Solusi

Strategi ini akan efektif melawan bot tanpa atau minim pergerakan. Namun, efektivitas strategi ini akan menurun jika seluruh bot musuh berada dalam jarak yang jauh atau berkumpul di suatu area. Jika bot musuh berkumpul di suatu area, strategi ini akan memilih bot musuh yang berbeda - beda setiap saat (jika bot juga bergerak). Akibatnya, bot tidak fokus dalam menghancurkan satu bot musuh, yang mungkin dapat mengurangi efektivitas penyerangan.

3.1.4 Greedy by Evasion and Opportunistic Attack

Strategi Greedy by Evasion and Opportunistic Attack fokus kepada gerakan terus-menerus untuk menghindari serangan dari musuh dengan tetap memanfaatkan kesempatan menyerang musuh jika dalam jangkauan tembak. Dalam strategi ini, bot bergerak dalam pola melingkar atau acak untuk menghindari serangan musuh lalu menyerang kembali ketika ada peluang yang menguntungkan.

Bot akan menembak dengan kekuatan maksimum jika musuh berada dalam jarak dekat dan mengurangi kekuatan jika semakin musuh menjauh. Selain itu, Bot tidak melakukan penggerakan radar secara aktif, melainkan hanya memindai musuh ketika masuk ke dalam jangkauan deteksinya.

A. Mapping persoalan dalam strategi ini sebagai berikut:
   - Himpunan Kandidat: Semua aksi yang dapat dilakukan bot, seperti bergerak maju, menembak, atau menghindar.
   - Himpunan Solusi: Kumpulan aksi yang sudah diambil
   - Fungsi Seleksi: Memilih aksi yang secara langsung meningkatkan peluang bertahan hidup sambil memanfaatkan peluang menyerang.

- Fungsi Kelayakan: Memastikan bot tidak menabrak dinding, tetap berada dalam arena, dan hanya menembak saat musuh dalam jangkauan optimal.
- Fungsi Solusi: Strategi dianggap berhasil jika bot mampu bertahan lebih lama dibandingkan musuh dan memanfaatkan kesempatan serangan secara optimal.
- Fungsi Objektif: Memaksimalkan kelangsungan hidup bot dengan terus bergerak dan menghindari serangan musuh, sambil tetap memberikan damage saat ada kesempatan.

B. Analisis Efisiensi Solusi

Strategi ini cukup efisien dalam mempertahankan bot dari serangan musuh dengan terus bergerak, mengurangi kemungkinan terkena tembakan. Tidak adanya pemindaian aktif mengurangi penggunaan sumber daya untuk pengolahan data musuh, tetapi bisa menyebabkan bot melewatkan musuh yang tidak masuk ke jalur deteksinya. Efisiensi dalam memberikan damage bisa berkurang karena bot hanya menyerang saat musuh terdeteksi, sehingga tidak selalu aktif dalam pertempuran.

C. Analisis Efektivitas Solusi

Strategi ini efektif dalam melawan musuh yang mengandalkan serangan diam atau memiliki pola pergerakan tetap, karena bot bisa menghindari banyak tembakan. Dalam pertarungan dengan bot yang lebih agresif dan memiliki sistem pemindaian aktif, strategi ini bisa kalah efektif karena bot hanya menyerang berdasarkan kesempatan. Jika bot bisa bertahan lebih lama daripada lawan, strategi ini bisa memberikan keuntungan dalam skor bertahan hidup.

3.2 Strategi Greedy yang Dipilih

Berdasarkan eksplorasi strategi greedy yang sudah dilakukan, strategi yang dipilih untuk menjadi strategi utama adalah Greedy by Energy and Distance. Strategi ini dipilih karena memberikan keseimbangan ideal antara penyerangan agresif dan penghematan energi, dua faktor yang memberikan dampak besar pada ketahanan bot dalam permainan. Bot dapat menyesuaikan diri secara dinamis dengan kondisi dan mempertimbangkan energi dan jarak musuh, sehingga memberikan fleksibilitas dalam menyerang dan bertahan.

Jika dibandingkan dengan Greedy by Damage and Accuracy, strategi ini lebih efisien dalam mempertahankan energi dan tidak hanya berfokus melakukan penyerangan agresif. Greedy by Damage and Accuracy tidak mempertimbangkan pertahanan dan energi sehingga lebih susah untuk bertahan hidup dan sangat rentan terhadap energi habis.

Jika dibandingkan dengan Greedy by Distance and Avoidance, strategi yang dipilih lebih fleksibel. Greedy by Distance and Avoidance lebih fokus kepada menyerang musuh

terdekat, tanpa memperhitungkan efisiensi dan pergerakan lawan. Sedangkan strategi yang dipilih bisa menyesuaikan kekuatan serangan dan pergerakan berdasarkan kondisi energi dari bot dan jarak lawan.

Jika dibandingkan dengan Greedy by Evasion and Opportunistic Attack, strategi yang dipilih lebih seimbang antara penyerangan dan pertahanan. Greedy by Evasion and Opportunistic Attack lebih fokus kepada penghindaran tanpa melakukan pemindaian aktif sehingga bisa melewatkan kesempatan menyerang yang efektif.

Efisiensi pendekatan ini dalam mengendalikan penggunaan energi adalah salah satu alasan utama untuk memilihnya. Ketika bot memiliki cukup energi, bot akan bertindak agresif, menembak dan bergerak secara berkala untuk menghindari serangan. Di sisi lain, bot akan berkonsentrasi pada penghindaran ketika energi mereka mulai menipis, bergerak cepat dan tidak menentu untuk menghindari tembakan lawan dan memperpanjang durasi pertandingan mereka. Ini menjamin bahwa bot bisa bertahan hidup dengan baik dan sulit untuk dikalahkan dengan cepat.

Dengan mempertimbangkan semua aspek tersebut, Greedy by Energy and Distance dipilih sebagai strategi terbaik untuk diimplementasikan dalam bot karena kemampuannya untuk beradaptasi dengan berbagai kondisi pertempuran. Kombinasi antara efisiensi energi dan fleksibilitas dalam pergerakan serta penyerangan menjadikannya pilihan yang optimal untuk memperoleh skor tinggi dan bertahan lebih lama dalam kompetisi Robocode Tank Royale.

BAB IV
IMPLEMENTASI DAN PENGUJIAN

4.1 Implementasi Solusi

4.1.1 Greedy by Energy and Distance (Nama bot: KuncirRamBOT)

Berikut adalah pseudocode untuk strategi Greedy by Energy and Distance, dimana bot akan menghitung jarak musuh dengan dirinya serta akan menembak jika energinya masih diatas sepuluh poin.

```
// Inisialisasi variabel global
arahSenjata : int <- 1 // Menentukan arah rotasi senjata
(1 searah jarum jam, -1 berlawanan)
scanning : boolean <- false // Status apakah bot sedang
melakukan scanning
maju : boolean <- true // Status apakah bot sedang
bergerak maju
kenceng : boolean <- false // Status apakah bot bergerak
dengan kecepatan tinggi
```

```
procedure Run()
  while IsRunning do
    AdjustGunForBodyTurn ← false
    maju ← true
    kenceng ← false
    GunTurnRate ← 15
   // Jika energi bot lebih dari 10, lakukan pergerakan
agresif
    if Energy > 10 then
      MaxSpeed ← 6
      MoveForward(4000)
      TurnRight(45)
      TurnLeft(45)
    else  // Jika energi kurang dari 10, bot fokus
menghindar dengan kecepatan tinggi
      MaxSpeed ← 8
      TurnLeft(10_000)
      MoveForward(10_000)
      kenceng ← true
    endif
    RotateGun(360) // Senjata terus berputar untuk
scanning
    Go()
```

```
  endwhile
endprocedure
```

```
procedure OnScannedBot(event)
// Bila ada bot yang terkena scan
  scanning ← true
  arahSenjata ← -arahSenjata // Arah senjata berputar
terus menerus
  AdjustGunForBodyTurn ← true
  distance ← DistanceTo(event.X, event.Y) // Variabel
untuk mengetahui jarak musuh

  // Mengarahkan senjata ke musuh
  TurnGunTowards(event.X, event.Y)
  // Jika energi lebih dari 10, bot menembak musuh
sesuai jarak dan sambil jiggle jiggle
  if Energy > 10 then
    MoveForward(4000)

    if distance < 75 then
      Fire(3)
    else if distance < 100 then
      Fire(2.8)
    else if distance < 150 then
      TurnRight(45)
      Fire(2.5)
    else if distance < 250 then
      TurnLeft(45)
      Fire(2.3)
    else if distance < 300 then
      TurnRight(45)
      Fire(2)
    else
      MoveBasedOnDistance(distance)
      Fire(1.5)
    endif
  // Jika energi kurang dari 10, bot tidak menembak
musuh namun bergerak berputar dengan kecepatan tinggi
  else
    MaxSpeed ← 8
    TurnLeft(10_000)
    MoveForward(10_000)
    kenceng ← true
  endif
```

```
  RotateGun(360 * arahSenjata)
  Go()
endprocedure
```

```
procedure OnHitBot(event)
// Jika bot menabrak atau ditabrak bot lain
// Bot akan memutar senjata ke posisi bot yang ditabrak
atau menabrak
  TurnGunTowards(event.X, event.Y)
  arahSenjata ← -arahSenjata
  RotateGun(360 * arahSenjata)
  // Jika bot kita yang menabrak bot lain maka akan maju
dan menembak musuh dengan kekuatan maksimal
  if event.IsRammed then
    MaxSpeed ← 8
    MoveForward(300)
    TurnLeft(115)
    Fire(3)
    kenceng ← true
  // Jika bot kita yang ditabrak bot lain maka akan
mundur dan berbelok untuk lari dengan kecepatan tinggi
  else
    MaxSpeed ← 8
    MoveBackward(100)
    TurnRight(50)
    kenceng ← true
  endif

  Go()
endprocedure
```

```
procedure OnHitWall(event)
// Jika bot menabrak dinding
  // Jika bot dalam posisi maju maka bot akan mundur
untuk efisiensi pergerakan tanpa berputar
  if maju then
    MoveBackward(4000)
    maju ← false
  // Jika bot dalam posisi mundur, maka bot akan maju
kembali
  else
    MoveForward(4000)
    maju ← true
```

```
  endif
  TurnLeft(115)

  // Jika bot sedang tidak sedang melakukan scanning,
maka bot melakukan scan 360 derajat untuk mencari bot
musuh
  if not scanning then
    RotateGun(360)
  endif

  Go()
endprocedure
```

```
procedure MoveBasedOnDistance(distance)
// Fungsi untuk bergerak berdasarkan jarak ke musuh >
300
  jarakbaru ← distance mod 3

  if jarakbaru = 0 then
    TurnRight(45)
    MoveForward(4000)
  else if jarakbaru = 1 then
    TurnLeft(45)
    MoveForward(4000)
  else
    MoveBackward(4000)
  endif
endprocedure
```

```
procedure TurnGunTowards(targetX, targetY)
// Fungsi untuk memutar senjata ke arah musuh
  bearing ← BearingTo(targetX, targetY)
  absoluteBearing ← Direction + bearing
  gunTurn ← absoluteBearing - GunDirection
  RotateGun(NormalizeBearing(gunTurn))
endprocedure
```

```
procedure NormalizeBearing(angle)
// Normalisasi sudut saat memutar senjata agar lebih
efisien
  while angle > 180 do
    angle ← angle - 360
  endwhile
  while angle < -180 do
```

```
    angle ← angle + 360
  endwhile
  return angle
endprocedure
```

4.1.2 Greedy by Damage and Accuracy (Nama bot: Sakbot)

Berikut adalah pseudocode untuk strategi Greedy by Damage and Accuracy, di mana bot akan mendekati musuh lalu menembak dengan kekuatan maksimal:

```
// Inisialisasi variabel global
enemyX, enemyY : double // Posisi musuh terakhir yang
terdeteksi
enemyDetected : boolean <- false // Status apakah musuh
terlihat
lastSeenTurn : int <- 0 // Turn terakhir musuh terlihat

SAFE_DISTANCE <- 50 // Jarak aman minimum dari musuh
FIRE_DISTANCE <- 150 // Jarak maksimum untuk menembak
RADAR_SWEEP_ANGLE <- 20 // Sudut pemindaian radar
```

```
procedure run()
  while isRunning do
    // Jika musuh tidak terdeteksi atau sudah lama tidak
    terlihat, scan area
    If (not enemyDetected or (TurnNumber - lastSeenTurn
    > 5)) then
      // Lakukan scanning
      TurnRadarRight(RADAR_SWEEP_ANGLE)
    else
      MoveTowardEnemy() // Gerak ke arah musuh
      FireIfInRange() // Tembak jika dalam jarak tembak
    endif
  endwhile
```

```
// Ketika bot mendeteksi musuh
procedure onScannedBot(event)
  // Simpan posisi musuh
  enemyX <- event.X
  enemyY <- event.Y
  enemyDetected <- true
  lastSeenTurn <- TurnNumber
```

```
  // Hitung sudut ke musuh, lalu sesuaikan radar
  angleToEnemy <- BearingTo(enemyX, enemyY)
  radarAdjust <- NormalizeAngel(angleToEnemy -
  RadarDirection)
  TurnRadarRight(radarAdjust)
```

```
// Fungsi untuk mendekati musuh
procedure MoveTowardEnemy()
  // Hitung jarak dan arah ke musuh
  distance <- CalculateDistance(enemyX, enemyY)
  angleToEnemy <- BearingTo(enemyX, enemyY)

  TurnLeft(angleToEnemy)

  // Jika jarak lebih dari batas aman, dekati musuh
  if (distance > SAFE_DISTANCE + 20) then
    Forward(distance - SAFE_DISTANCE)
  endif
```

```
// Fungsi untuk menembak jika dalam jangkauan
procedure FireIfInRange()
  distance <- CalculateDistance(enemyX, enemyY)

  // Tembak dengan kekuatan maksimum jika musuh dalam
  jarak tembak
  if (distance <= FIRE_DISTANCE and GunHeat = 0) then
    Fire(3)
  endif

  // Sesuaikan radar ke musuh setelah menembak
  angleToEnemy <- BearingTo(enemyX, enemyY)
  TurnRadarRight(angleToEnemy - RadarDirection)

  // Jika musuh tidak terlihat selama 5 turn, reset
  scanning radar
  if (TurnNumber - lastSeenTurn > 5) then
    enemyDetected <- false
    TurnRadarRight(RADAR_SWEEP_ANGLE)
  endif
```

```
// Fungsi menghitung jarak ke musuh
function CalculateDistance(x: Integer, y: Integer) ->
Integer
```

```
<- SquareRoot((x - BotX)^2 + (y - BotY)^2)
```

4.1.3 Greedy by Distance and Avoidance (Nama bot: Tang)

Berikut adalah pseudocode untuk strategi Greedy by Distance and Avoidance.

```
// Inisialisasi variabel global
double distance
bool closest
double t_x, t_y
```

```
  // Inisialisasi bot
procedure Run()
  while isRunning do
      AdjustRadarForGunTurn <- true
      SetColor(Body, Gray)
      close <- false
      closest <- Infinity

      // Pola pergerakan utama
      Forward(200)
      TurnRight(90)
      SetTurnRadarRight(90)
      TurnLeft(90)
  endwhile
```

```
// Ketika bot mendeteksi musuh
procedure onScannedBot(event)
  distance <- CalculateDistanceToEnemy(event.X, event.Y)

  // Jika musuh lebih dekat dari sebelumnya, jadikan
  target utama
  if (distance < closest) then
      close <- true
      closest <- distance
      t_x <- event.X
      t_y <- event.Y
  else
      close <- false
  endif
  if (close) then
      firepower <- (50 / distance) * 10
      TurnGunLeft(GunBearingTo(t_x, t_y))
```

```
      Fire(firepower)
  endif

  if (distance < 50) then
      SetTurnRight(90)
      SetForward(200)
  else
      SetForward(200)
  endif
```

```
  // Ketika bot bertabrakan dengan bot lain
procedure onHitBot(event)
  if (event.IsRammed) then
      SetTurnRight(90)
      SetBack(100)
  else
      SetTurnRight(90 + BearingTo(event.X, event.Y))
      SetBack(200)
      SetTurnRight(45)
      SetForward(200)
  endif
```

```
  // Ketika bot terkena tembakan
procedure onHitByBullet(event)
  bearing <- CalcBearing(event.Bullet.Direction)
  TurnLeft(90 - bearing)
```

4.1.4 Greedy by Evasion and Opportunistic Attack (Nama bot: Hytam)

Berikut adalah pseudocode untuk strategi Greedy by Evasion and Opportunistic Attack, di mana bot akan selalu bergerak menghindar tembakan musuh lalu menembak jika terdapat kesempatan.

```
// Inisialisasi bot
procedure Run()
  while isRunning do
    SetTurnLeft(10_000) // Putar terus ke kiri
    MaxSpeed <- 5
    Forward(10_000)// Bergerak maju tanpa henti
  endwhile
```

```
// Ketika bot mendeteksi musuh
procedure onScannedBot(event)
  distance <- CalculateDistanceToEnemy(event.X, event.Y)

  // Tembak dengan kekuatan berbeda berdasarkan jarak
  if (distance < 200) then
    Fire(3)
  else
    Fire(2)
  endif

  // Arahkan meriam ke musuh
  SetTurnGunTowards(event.X, event.Y)
```

```
// Ketika bot bertabrakan dengan bot lain
procedure onHitBot(event)
  SetTurnGunTowards(event.X, event.Y)
  Forward(50) // Bergerak maju untuk menghindari
  tabrakan
```

```
// Ketika bot menabrak dinding
procedure onHitWall(event)
  TurnLeft(115)
```

```
// Menghitung jarak ke musuh
function CalculateDistanceToEnemy(x: double, y: double)
-> double
  deltaX <- x - BotX
  deltaY <- y - BotY
  <- SquareRoot(deltaX * deltaX + deltaY * deltaY)
```

```
// Mengarahkan meriam ke musuh
procedure SetTurnGunTowards(targetX, targetY)
  angleToTarget = Arctan2(targetY - BotY, targetX -
  BotX) * (180 / PI)
  gunTurnAngle = angleToTarget - GunDirection
  SetTurnGunRight(NormalizeBearing(gunTurnAngle))
```

4.2 Pengujian dan Analisis

Kami melakukan pengujian terhadap empat bot yang telah kami buat dengan metode setiap bot saling melawan satu sama lain sebanyak tiga kali. Saat melakukan pengujian,

kami memperhatikan perilaku tiap bot dan menganalisis berdasarkan hasil yang didapat dari tiga hasil pengujian.

4.2.1 Pengujian

Kami melakukan tiga kali pengujian menggunakan empat bot yang telah kami buat dan hasilnya adalah sebagai berikut.

Pengujian 1

Results for 10 rounds

Rank	Name	Total Score	Survival	Surv. Bonus	Bullet Dmg.	Bullet Bonus	Ram Dmg.	Ram Bonus	1sts	2nds	3rds
1	KuncirRamBOT 1.0	1594	600	30	815	65	84	0	2	4	0
2	Hytam 1.0	1530	700	120	554	60	61	34	4	1	0
3	Tang 1.0	474	350	0	113	1	10	0	0	1	4
4	Sakbot 1.0	244	100	0	128	3	13	0	0	0	2

Pengujian 2

Results for 10 rounds

Rank	Name	Total Score	Survival	Surv. Bonus	Bullet Dmg.	Bullet Bonus	Ram Dmg.	Ram Bonus	1sts	2nds	3rds
1	KuncirRamBOT 1.0	1457	500	60	728	87	82	0	2	3	0
2	Hytam 1.0	1087	500	60	446	35	46	0	3	1	1
3	Tang 1.0	392	300	0	83	2	6	0	0	1	3
4	Sakbot 1.0	158	50	0	96	0	12	0	0	0	1

Pengujian 3

Results for 10 rounds

Rank	Name	Total Score	Survival	Surv. Bonus	Bullet Dmg.	Bullet Bonus	Ram Dmg.	Ram Bonus	1sts	2nds	3rds
1	KuncirRamBOT 1.0	1307	500	60	578	59	97	12	3	1	1
2	Hytam 1.0	1178	550	30	488	27	55	28	2	3	0
3	Tang 1.0	418	250	30	98	0	40	0	0	1	2
4	Sakbot 1.0	321	150	0	160	0	11	0	0	0	2

4.2.2 Analisis

Berdasarkan tiga pengujian yang sudah dilakukan, klasemen tiap bot untuk pengujian-nya selalu sama, yaitu bot KuncirRamBOT pertama, bot Hytam kedua, bot Tang ketiga, dan bot Sakbot keempat. Analisis ini akan dibagi menjadi empat bagian, yaitu analisis berdasarkan waktu bertahan, damage peluru, damage *ramming,* dan skor akhir.

- Analisis berdasarkan waktu bertahan

  Berdasarkan hasil dari tiga kali pengujian, terbukti bahwa bot Hytam memiliki kemampuan bertahan paling baik. Hal itu bisa dilihat pada kolom *survival*, *survival bonus,* dan *ranking* tiap pengujiannya. Hal itu disebabkan gerakan bot Hytam yang selalu berputar dalam melingkar. Gerakan berputar tersebut bisa bermanfaat untuk menghindari peluru-peluru yang menuju ke arah bot Hytam. Juara bertahan kedua adalah bot KuncirRamBOT yang disebabkan pergerakan *random* yang ia miliki. Memang hal ini membuat penembakan peluru menjadi kurang efektif dan mudah menabrak tembok, namun terdapat algoritma *greedy* ketika energi dibawah sepuluh maka bot ini akan berputar-putar saja tanpa menembak untuk mengulur waktu. Peringkat ketiga dan keempat adalah bot Tang dan Sakbot dimana pergerakan dari kedua tank ini tergolong tidak cukup lincah karena kedua

bot ini harus melakukan *scanning* terlebih dahulu baru bergerak ke musuh. Selain itu, Sakbot tidak memiliki strategi pertahanan atau penghindaran dari tembakan musuh sehingga lebih mudah di target dan terkena serangan musuh. Berdasarkan analisis ini, strategi *greedy* untuk pergerakan paling optimal adalah pergerakan terus memutar tanpa henti.

- Analisis berdasarkan damage peluru

  Berdasarkan hasil dari tiga kali pengujian, terbukti bahwa bot KuncirRamBOT memimpin untuk kategori ini, diikuti bot Hytam, bot Sakbot, dan bot Tang. Hal ini dikarenakan bot KuncirRamBOT terus menembak tanpa henti jika ia melihat musuh dalam area *scan*. Hal itu membuat banyak peluru yang terkena ke musuh yang tidak ditargetkan juga. Analisis kedua untuk bot Hytam, bot ini mendapat cukup banyak damage peluru karena gerakannya yang membuat proses *scan* musuh menjadi lebih mudah. Analisis ketiga untuk bot Sakbot, penerapan algoritma *greedy* untuk bot ini tergolong cukup efektif dimana bot mendekat dan memberikan *burst* damage kepada lawannya. Namun, kelemahan dari Sakbot terlihat ketika dilawankan dengan bot-bot lain yang memiliki strategi penghindaran dan pergerakan yang kompleks. Sakbot dirancang untuk melawan bot yang diam atau pergerakannya sederhana. Karena bot lain pergerakannya kompleks, sebagian besar serangan Sakbot tidak kena target. Untuk bot Tang sedikit sulit mendapatkan poin dari peluru karena frekuensi penembakan bot Tang tergolong cukup jarang. Berdasarkan analisis ini, dapat ditentukan bahwa algoritma *greedy* paling optimal dilakukan dengan menembak secara terus menerus.

- Analisis berdasarkan damage *ramming*

  Berdasarkan hasil dari tiga kali pengujian, terbukti bahwa bot KuncirRamBOT memimpin untuk kategori ini, diikuti dengan bot Hytam, bot Tang, dan bot Sakbot. Hal ini dikarenakan bot KuncirRamBot bergerak secara random dengan kecepatan cukup tinggi yang membuat kemungkinan menabrak bot lain lebih besar. Selain itu terdapat algoritma *greedy* ketika menabrak bot lain juga menjadi salah satu faktor besar. Analisis kedua untuk bot Hytam, damage *ramming* dari bot ini cukup besar dikarenakan gerakan berputarnya yang ada kemungkinan menabrak bot lain yang sedang melewatinya. Analisis ketiga untuk bot Sakbot, damage *ramming* dari bot ini tidak terlalu besar dikarenakan memang fokus bot ini untuk menembak, bukan *ramming*. Untuk bot Tang, bot ini bergerak cukup unik yang mungkin membuat beberapa robot di dekatnya terkena ramming. Berdasarkan analisis diatas, implementasi *greedy* dengan gerakan acak yang lincah paling baik untuk mendapatkan damage *ramming* terbesar.

- Analisis berdasarkan skor akhir

Berdasarkan hasil tiga kali pengujian, terbukti bahwa KuncirRamBOT memimpin untuk kategoti skor akhir, diikuti dengan bot Hytam, bot Tang, dan bot Sakbot sesuai klasemen awal. Namun, perbedaan skor antara Hytam dan KuncirRamBOT tiap pengujianya tidak terlalu jauh. Hal ini membuktikan bahwa metode *greedy* dari kedua bot ini cukup efektif untuk mendapatkan skor yang besar. Penggabungan dua metode *greedy* dari kedua bot ini mungkin saja menghasilkan bot yang sangat kuat dibanding bot-bot lain karena kedua bot ini saling melengkapi satu sama lain.

BAB V
KESIMPULAN

5.1 Kesimpulan

Berdasarkan laporan tugas besar ini, kami berhasil menerapkan algoritma *greedy* pada semua bot yang telah kami buat. Setiap bot yang kami buat memiliki keunikan masing-masing dan memiliki pendekatan terhadap metode *greedy*-nya masing–masing. Hal itu yang menjadi ciri khas dari setiap bot yang kami buat. Tentunya, setiap bot yang kami buat memiliki kelebihan dan juga kekurangan. Kelebihan dan kekurangan dari bot yang kami buat akan saling melengkapi dan akan kami jadikan pelajaran pada proyek-proyek sejenis selanjutnya. Pada akhirnya, kami menentukan satu bot terbaik dari empat bot yang kami buat untuk kompetisi antar kelompok. Kami memilih bot tersebut dengan pertimbangan paling cukup untuk melawan bot-bot hebat lainnya.

5.2 Saran

Sangat banyak ruang pengembangan terhadap tugas besar ini yang tidak terfokus hanya pada metode *greedy* saja tapi bisa dikembangakan lebih jauh lagi dengan metode yang lebih rumit lagi. Hal tersebut dapat mendorong performa dari bot yang dibuat namun akan melewati batasan dari tugas besar ini dalam implementasi algoritma *greedy* dan pemahaman algoritma *greedy*.

5.3 Refleksi

Kami merasa bisa lebih baik dalam mengerjakan tugas besar ini, tapi kesibukan masing-masing dari kami yang membuat pengerjaan kami tidak sebaik biasanya. Oleh karena itu, kami mengambil banyak pelajaran dari tugas besar ini terkait manajemen waktu yang baik serta tidak menyepelekan segala sesuatunya.

LAMPIRAN

Tautan Repository Github

https://github.com/rakdaf08/Tubes1_tankyou

Tautan Link Video

https://www.youtube.com/watch?v=yLPphQCgPh8

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

DAFTAR PUSTAKA

https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2024-2025/04-Algoritma-Greedy-(2025)-Bag1.pdf

https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2024-2025/05-Algoritma-Greedy-(2025)-Bag2.pdf

https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2024-2025/06-Algoritma-Greedy-(2025)-Bag3.pdf

https://robocode-dev.github.io/tank-royale/articles/gui.html