

IF2211 - Strategi Algoritma
Laporan Tugas Besar 1 Stima: Robocode Dengan
Algoritma Greedy



Azadi Azhrah 12823024

Samuel Gerrard 13523064
Hamonangan Girsang

Nadhif Al Rozin 13523076

Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung
2025

BAB I

DESKRIPSI MASALAH

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, akan dibuat sebuah bot yang nantinya akan dipertandingkan satu sama lain. Bot yang digunakan akan menggunakan strategi greedy dalam ide utamanya.

BAB II

LANDASAN TEORI

Algoritma greedy adalah teknik optimasi yang membuat keputusan secara lokal optimal pada setiap langkah dengan harapan bahwa keputusan ini akan menghasilkan solusi yang global optimal. Dalam setiap langkahnya, algoritma ini memilih opsi terbaik yang tersedia tanpa mempertimbangkan konsekuensi jangka panjang. Algoritma greedy merupakan metode yang paling populer dan sederhana untuk memecahkan persoalan optimasi. Hanya ada dua macam persoalan optimasi, yaitu maksimasi dan minimasi. Algoritma greedy membentuk solusi langkah per langkah. Pada setiap langkah, terdapat banyak pilihan yang perlu dievaluasi. Oleh karena itu, pada setiap langkah harus dibuat keputusan yang terbaik dalam menentukan pilihan. Algoritma ini tidak mempunyai kemungkinan untuk merubah keputusan yang sudah dibuat. Namun, tidak semua masalah dapat diselesaikan secara optimal dengan metode ini.

Terdapat beberapa komponen dalam algoritma greedy yaitu:

1. Himpunan Kandidat: Kumpulan kandidat yang dapat dipilih.
2. Himpunan Solusi: Kumpulan kandidat yang telah dipilih sebagai solusi.
3. Fungsi Solusi: Menentukan apakah kandidat yang telah dipilih dapat memberi solusi.
4. Fungsi Seleksi: Fungsi yang memilih kandidat berdasarkan strategi greedy tertentu. Strategi yang digunakan bersifat heuristik.
5. Fungsi Kelayakan: Menentukan apakah kandidat yang dipilih layak masuk ke dalam himpunan solusi.
6. Fungsi Objektif: Memaksimalkan atau meminimalkan solusi dari strategi greedy.

Robocode adalah permainan pemrograman yang bertujuan untuk membuat kode bot. Bot tersebut adalah tank yang harus bersaing dengan tank lain di arena pertempuran virtual. "Pemain" dalam permainan ini adalah programmer (coder) bot, yang tidak akan memiliki pengaruh langsung pada permainan saat permainan sedang berjalan. Sebaliknya, pemain ("Robocoder") harus menulis program yang menjadi otak tank. Program (kode) tersebut memberi tahu bagaimana tank harus berperilaku dan bereaksi terhadap peristiwa yang terjadi di medan pertempuran.

Kode tersebut akan memberi tahu tank cara bergerak di medan perang dan menghindari tembakan musuh saat bergerak dalam berbagai situasi. Dan pada saat

yang sama, tank harus memindai tank musuh dengan memutar radarnya, dan menembakkan senjatanya ke arah tank musuh.

Sebuah pertempuran antara bot di Robocode dapat terdiri dari beberapa ronde. Sebuah pertempuran terdiri dari 10 ronde individu, di mana setiap ronde akan memiliki pemenang dan yang kalah. Setiap ronde dibagi menjadi giliran, yang merupakan unit waktu terkecil. Satu giliran adalah satu ketukan jam dan satu siklus permainan di Robocode. Berapa banyak giliran yang dibutuhkan dalam satu ronde tergantung pada berapa lama waktu yang dibutuhkan oleh penyintas terakhir untuk bertahan dalam ronde tersebut.

Pada setiap giliran, setiap bot dapat menggerakkan bot, memindai musuh, dan berpotensi menembakkan senjata. Selain itu juga bereaksi terhadap peristiwa seperti ketika bot terkena peluru, bertabrakan dengan bot lain, atau menabrak dinding

Semua bot (tank) mendapat skor dan penempatan tergantung pada seberapa baik mereka tampil di medan perang. Jadi tujuannya adalah untuk mendapatkan skor dan penempatan setinggi mungkin.

Pada tugas besar ini, akan diaplikasikan algoritma greedy untuk mendasari pembuatan sebuah bot. Nantinya, bot ini akan dipertandingkan untuk memperoleh algoritma apa yang paling optimal digunakan untuk sebuah bot.

BAB III

APLIKASI STRATEGI GREEDY

A. Mapping Kondisi

1. Himpunan Kandidat

Semua function yang dapat dilakukan di robocode tank royale, dapat berupa gerakan movement (Forward, Back, TurnGun, TurnRadar), Prosedur Fire, dan event event lain seperti Run, ScannedBot, Hit, BulletHit dll.

2. Himpunan Solusi

Function dan Prosedur yang dipilih untuk digunakan

3. Fungsi Solusi

Score pada aspek greedy yang dipilih punya nilai yang sesuai

4. Fungsi Seleksi

Gunakan metode lock target, Energy Management, Dodging dan Pencarian posisi strategis

5. Fungsi Kelayakan

Memeriksa apakah function dan prosedur yang digunakan tidak bertabrakan

6. Fungsi Objektif

Jumlah Score yang didapat untuk 10 ronde maksimal

B. Alternatif Solusi

1. Solusi 1: Claude

Claude merupakan salah satu alternatif solusi yang kami buat. Claude memiliki fokus untuk mengambil keuntungan sebanyak mungkin dari tembakan yang mengenai lawan. Claude melompat ke tengah arena untuk mendapatkan asumsi jarak yang dekat dengan semua bot. Setelah mencapai pusat, Claude akan terus berputar dengan kecepatan sedang dan menggerakkan laras meriamnya tanpa henti untuk melakukan pemindaian. Setiap kali mendeteksi bot lawan, ia langsung menembakkan peluru dengan kekuatan 1, tanpa mempertimbangkan jarak atau status musuh. Strategi ini membuatnya lebih reaktif daripada strategis, mengandalkannya di pusat arena untuk mendeteksi dan menyerang lawan yang mendekat. Namun, bot ini tidak memiliki mekanisme penghindaran atau pengejaran musuh, sehingga rentan terhadap serangan lawan yang lebih agresif atau taktik yang lebih kompleks.

2. Solusi 2: Dodge

Dodge adalah alternatif solusi lainnya yang memiliki fokus di aspek bertahan hidup (*survival*), yaitu mobilitas dan penghindaran. Dodge memanfaatkan strategi berbasis kecepatan dan pergerakan acak untuk menghindari serangan lawan sambil tetap mampu memberikan tekanan melalui tembakan terarah.. Dodge lebih *fluid* jika dibandingkan dengan *Spinbot*. Saat dijalankan, bot ini mulai bergerak maju dengan pola zig-zag yang tidak terduga untuk menghindari tembakan lawan. Sementara itu, laras meriamnya terus berputar tanpa henti, memungkinkan pemindaian musuh secara konstan. Saat mendeteksi lawan, bot ini menghitung jarak ke target dan menyesuaikan kekuatan tembakannya—menggunakan peluru berkekuatan 1 jika musuh jauh dan 3 jika musuh dekat, meningkatkan peluang serangan efektif. Jika ia menabrak dinding atau bertabrakan dengan bot lain, ia segera membalik arah, yang secara bergantian membuatnya bergerak maju atau mundur untuk menghindari terjebak.

3. Solusi 3: PersonalBot

PersonalBot adalah salah satu alternatif solusi yang kami pilih. PersonalBot sesuai namanya 'Personal' akan mengunci satu target hingga target tersebut hancur. Bot ini memanfaatkan informasi yang ada yaitu bot musuh yang pertama kali discan. PersonalBot akan menyimpan data id dari bot yang pertama kali discan dan hanya akan menyerang bot dengan id yang sesuai. PersonalBot juga menghitung faktor jarak. Jika PersonalBot berada di jarak yang cukup jauh dengan bot target, maka PersonalBot akan berusaha mendekati target. Jika jarak dianggap cukup, PersonalBot akan menembak target tersebut.

4. Solusi 4: Oportunis

Oportunis adalah bot yang menggunakan strategi berbasis energi dan perilaku musuh untuk menentukan tindakannya. Bot ini beroperasi dengan cara memindai area dan menyesuaikan strateginya berdasarkan status energinya.

Jika energinya rendah, bot ini akan mencari posisi aman berdasarkan distribusi musuh dan bergerak menjauh. Jika tidak dalam mode bertahan, bot dapat mengejar lawan yang memiliki energi rendah untuk memburu dan menghabisi mereka. Bot ini akan menghindari serangan dengan cara berputar secara acak dan bergerak dengan kecepatan tinggi. Saat memindai musuh, bot menyimpan informasi mereka dan menyesuaikan arah radar serta meriam untuk menembak dengan kekuatan yang dihitung berdasarkan jarak. Jika musuh memiliki kecepatan rendah, bot akan menembakkan tembakan kuat untuk meningkatkan kemungkinan mengenai sasaran. Selain itu, bot juga bereaksi terhadap kejadian seperti menabrak dinding atau menabrak bot lain.

C. Analisis

1. Perkiraan efisiensi dan Efektifitas solusi 1

Claude menitik beratkan pada pengoptimalan setiap bullet yang ditembakkan, hal ini dilakukan dengan mengambil posisi yang paling dekat dengan semua bot, untuk mempercepat proses berpikir, ini selalu ditempatkan di tengah arena. Tentu agar tidak menjadi target yang empuk, bot akan berputar sambil menembakkan peluru ke bot yang ada dalam radarnya. Bot ini akan lebih efektif di kondisi dengan banyak bot lain, dan akan semakin buruk semakin sedikit bot musuh, dikarenakan target yang ditembak adalah random sehingga jika hanya ada sedikit musuh akan membuang banyak waktu untuk memutar tembakan.

2. Perkiraan efisiensi dan Efektifitas solusi 2

Dodge berfokus pada menghindari peluru musuh, bot ini memiliki beberapa random turn yang ada pada movement nya sehingga tidak monoton, bot ini bagus melawan bot lain yang mencoba memprediksi gerakan dan berfokus pada menembak, namun jika melawan bot yang mengejar dan membunuh hanya 1 target bot ini bisa kalah jika tidak ada ruang untuk lari (ada diantara tembok dan musuh), atau musuh mengejar lebih cepat dari bot ini bisa berlari.

3. Perkiraan efisiensi dan Efektifitas solusi 3

PersonalBot memberikan kesempatan dalam efisiensi energi. Hal ini karena PersonalBot memfokuskan penggunaan energinya hanya pada 1 bot. PersonalBot ini sangat efektif terutama melawan bot dengan pergerakan yang *predictable* dan linear. Tetapi, dikarenakan PersonalBot hanya mengincar satu orang, maka PersonalBot sangat rentan terhadap serangan bot lain yang bukan merupakan target dari PersonalBot. PersonalBot juga memiliki kelemahan ketika melawan bot yang memiliki pergerakan lincah dan sulit diikuti.

4. Perkiraan efisiensi dan Efektifitas solusi 4

Oportunis adalah bot yang berfokus di energy management, bot ini adalah bot yang sangat baik untuk bertarung 1 vs 1, hal ini dikarenakan bot ini perlu lebih banyak waktu semakin banyak musuh yang ada untuk mempertimbangkan hal terbaik yang bisa dilakukan. Namun meski begitu hal ini tidak menjadi masalah meskipun dalam battle royale, karena hal yang cukup lama ini dilakukan saat ia low energy saja dan ia tetap melakukan gerakan lain sambil menghitung gerakan selanjutnya.

5. Solusi yang dipilih

Solusi yang kami pilih adalah Oportunis, hal ini dikarenakan hasil yang didapat lebih baik dari pada solusi lain secara mayoritas, meskipun dalam kondisi terburuk hasilnya sangat buruk jika bertemu musuh yang tidak tepat

BAB IV

IMPLEMENTASI DAN PENGUJIAN

A. Implementasi

1. Claude

```
DECLARE RandomNumberGenerator
DECLARE LowEnergy AS BOOLEAN = FALSE

FUNCTION Main()
    CREATE INSTANCE OF ClaudeBot
    START ClaudeBot

CLASS ClaudeBot EXTENDS Bot
    FUNCTION Constructor()
        CALL BASE CONSTRUCTOR WITH CONFIGURATION FROM
        "Claude.json"

    FUNCTION Run()
        SET BodyColor = LightBlue
        SET TurretColor = Blue
        SET RadarColor = DarkBlue
        SET ScanColor = Aqua
        SET BulletColor = DeepSkyBlue

        INITIALIZE RandomNumberGenerator

        CALL GoToCenter()

        WHILE Bot Is Running
            SET TargetSpeed = 3
            SET TurnRate = 20
            TURN Gun Left Infinitely
            WAIT UNTIL Turn Completes

        FUNCTION OnScannedBot(event)
            FIRE Bullet With Power 1

        FUNCTION GoToCenter()
            PRINT "Going to center"
            SET CenterX = Arena Width / 2
            SET CenterY = Arena Height / 2

            CALL TurnToFaceTarget(CenterX, CenterY)
            MOVE Forward DistanceTo(CenterX, CenterY)
            WAIT UNTIL Turn Completes

        FUNCTION TurnToFaceTarget(x, y)
```

```

    SET Bearing = BearingTo(x, y)
    TURN Left Bearing

CLASS TurnCompleteCondition EXTENDS Condition
    DECLARE BotInstance

    FUNCTION Constructor(bot)
        SET BotInstance = bot

    FUNCTION Test()
        RETURN BotInstance.TurnRemaining == 0

```

2. Dodge

```

DECLARE MovingForward AS BOOLEAN
DECLARE RandomNumberGenerator

FUNCTION Main()
    CREATE INSTANCE OF DodgeBot
    START DodgeBot

CLASS DodgeBot EXTENDS Bot
    FUNCTION Constructor()
        CALL BASE CONSTRUCTOR WITH CONFIGURATION FROM
        "Dodge.json"

    FUNCTION Run()
        SET BodyColor = Blue
        SET TurretColor = Blue
        SET RadarColor = Black
        SET ScanColor = Yellow

        SET MovingForward = TRUE
        ROTATE Gun Right Infinitely

        WHILE Bot Is Running
            MOVE Forward 40000
            TURN Right 60
            WAIT UNTIL Turn Completes

            TURN Left (90 + Random Value Between 0-30)
            WAIT UNTIL Turn Completes

            TURN Right (60 + Random Value Between 0-30)
            WAIT UNTIL Turn Completes

```

```

FUNCTION OnScannedBot(event)
    SET dx = event.X - CurrentX
    SET dy = event.Y - CurrentY
    SET Distance = SquareRoot(dx^2 + dy^2)

    IF Distance > 100 THEN
        SET FirePower = 1
    ELSE
        SET FirePower = 3

    FIRE Bullet With FirePower

FUNCTION OnHitWall(event)
    CALL ReverseDirection()

FUNCTION OnHitBot(event)
    IF event.IsRammed THEN
        CALL ReverseDirection()

FUNCTION ReverseDirection()
    IF MovingForward THEN
        MOVE Back 150
        SET MovingForward = FALSE
    ELSE
        MOVE Forward 150
        SET MovingForward = TRUE

CLASS TurnCompleteCondition EXTENDS Condition
    DECLARE BotInstance

    FUNCTION Constructor(bot)
        SET BotInstance = bot

    FUNCTION Test()
        RETURN BotInstance.TurnRemaining == 0

```

3. PersonalBot

```

DECLARE targetBotId = -1
DECLARE movingForward AS BOOLEAN = TRUE
DECLARE lastScannedTick = 0
DECLARE currentTick = 0

FUNCTION Main(args)
    CREATE INSTANCE OF PersonalBot
    START PersonalBot

```

```

CLASS PersonalBot EXTENDS Bot
  FUNCTION Constructor()
    CALL BASE CONSTRUCTOR WITH CONFIGURATION FROM
    "PersonalBot.json"

  FUNCTION Run()
    SET BodyColor = Red
    SET movingForward = TRUE

    WHILE Bot Is Running
      INCREMENT currentTick
      PRINT "Current tick: " + currentTick

      ROTATE Gun Right Infinitely
      MOVE Forward 40000
      TURN Right 60
      WAIT UNTIL Turn Completes

      TURN Left 60
      WAIT UNTIL Turn Completes

      IF targetBotId != -1 AND currentTick - lastScannedTick > 3 THEN
        PRINT "Lost target: " + targetBotId
        SET targetBotId = -1

  FUNCTION OnScannedBot(event)
    IF targetBotId == -1 THEN
      SET targetBotId = event.ScannedBotId
      PRINT "Locked on target: " + targetBotId

    IF event.ScannedBotId == targetBotId THEN
      SET lastScannedTick = currentTick
      PRINT "Scanned Tick: " + lastScannedTick

      CALL TurnToFaceTarget(event.X, event.Y)

      SET distance = DistanceTo(event.X, event.Y)
      IF distance > 100 THEN
        MOVE Forward 50
      ELSE
        MOVE Back 30

      SET bearingFromGun = GunBearingTo(event.X, event.Y)
      TURN Gun Left bearingFromGun

      IF ABS(bearingFromGun) <= 2 THEN
        FIRE(2)

      IF bearingFromGun == 0 THEN

```

```

        RESCAN()

    FUNCTION OnHitWall(event)
        CALL ReverseDirection()
        TURN Left 45

    FUNCTION OnHitBot(event)
        IF event.IsRammed OR NOT event.IsRammed THEN
            CALL ReverseDirection()
            TURN Left 45

    FUNCTION ReverseDirection()
        IF movingForward THEN
            MOVE Back 100
            SET movingForward = FALSE
        ELSE
            MOVE Forward 100
            SET movingForward = TRUE

    FUNCTION TurnToFaceTarget(x, y)
        SET bearing = BearingTo(x, y)
        TURN Left bearing

    CLASS TurnCompleteCondition EXTENDS Condition
        DECLARE botInstance

    FUNCTION Constructor(bot)
        SET botInstance = bot

    FUNCTION Test()
        RETURN botInstance.TurnRemaining == 0

```

4. Oportunis

```

    DECLARE lowEnergy AS BOOLEAN = false
    DECLARE chasing AS BOOLEAN = false
    DECLARE random = new Random()
    DECLARE TargetX, TargetY, TargetId
    DECLARE enemyStates = Dictionary of (int, ScannedBotEvent)
    DEFINE CLASS Oportunis EXTENDS Bot:

        FUNCTION Main():
            CREATE Oportunis instance
            CALL Start()

        CONSTRUCTOR Oportunis():

```

```
CALL BASE CONSTRUCTOR WITH BotInfo.FromFile("Oportunis.json")
```

```
PROCEDURE Run():
```

```
  SET BodyColor TO Gray
```

```
  SET TargetSpeed TO 7
```

```
  WHILE Bot is Running:
```

```
    Turn Radar Left 360 degrees
```

```
    lowEnergy <- Energy < 40
```

```
    IF lowEnergy:
```

```
      CALL Escape()
```

```
    ELSE:
```

```
      IF chasing:
```

```
        CALL Chase()
```

```
      ELSE:
```

```
        CALL Dodge()
```

```
FUNCTION Escape():
```

```
  SET TargetSpeed TO 8
```

```
  PRINT "Not Today, Buddy Boy!"
```

```
  (safeX, safeY) = Safe()
```

```
  PRINT "I am safe here", safeX, safeY
```

```
  GoTo(safeX, safeY)
```

```
  Dodge()
```

```
FUNCTION Chase():
```

```
  PRINT "Thou shall die", TargetId
```

```
  CALL TurnBodyToTarget(TargetX, TargetY)
```

```
  SET TargetSpeed TO 5
```

```
FUNCTION Dodge():
```

```
  PRINT "Ha! I Raised My Dex Stat to lvl 9999"
```

```
  SET TargetSpeed TO 8
```

```
  SET Turn Right (90 + Random Angle)
```

```
  WAIT UNTIL Turn Complete
```

```
  SET Turn Right (90 + Random Angle)
```

```
  WAIT UNTIL Turn Complete
```

```
FUNCTION Safe() RETURNS (double, double):
```

```
  SET safeX, safeY TO Current X, Y
```

```
  IF enemyStates IS EMPTY:
```

```
    RETURN (safeX, safeY)
```

```
  SET avgX, avgY TO 0
```

```
  FOR EACH enemy IN enemyStates:
```

```
    ADD enemy.X TO avgX
```

```
    ADD enemy.Y TO avgY
```

DIVIDE avgX, avgY BY enemyStates.Count

SET angle TO Angle Between (Current Position) AND (avgX, avgY)
SET safeX, safeY TO New Position AWAY FROM avgX, avgY

ENSURE safeX, safeY ARE WITHIN Arena Bounds
RETURN (safeX, safeY)

FUNCTION OnScannedBot(event e):

PRINT "Found Them!", e.ScannedBotId, "at", e.X, e.Y

STORE e IN enemyStates[e.ScannedBotId]

COMPUTE BearingRadar TO e.X, e.Y

IF lowEnergy:

CALL TurnGunToTarget(e.X, e.Y)

FIRE 1

IF BearingRadar < 3:

CALL firePower(Distance to e.X, e.Y)

CALL Rescan()

ELSE:

IF chasing AND TargetId != e.ScannedBotId:

RETURN

SET chasing TO true IF e.Energy < 40

SET TargetX, TargetY, TargetId TO e.X, e.Y, e.ScannedBotId

IF BearingRadar < 3:

CALL Rescan()

CALL TurnGunToTarget(e.X, e.Y)

IF e.Speed < 3:

CALL firePower(Distance to e.X, e.Y)

ELSE:

FIRE 1

FUNCTION OnBotDeath(event e):

PRINT "Bot", e.VictimId, "has been returned to the void"

REMOVE e.VictimId FROM enemyStates

IF e.VictimId == TargetId:

SET chasing TO false

PRINT "Gotcha! gotta kill them all"

FUNCTION firePower(distance):

COMPUTE power BASED ON distance

FIRE power

```
FUNCTION distance(toX, toY) RETURNS double:  
    RETURN Euclidean Distance BETWEEN (X, Y) AND (toX, toY)
```

```
FUNCTION OnHitBot(event e):  
    CALL TurnGunToTarget(e.X, e.Y)  
    FIRE 3  
    PRINT "Get out of my way", e.VictimId
```

```
FUNCTION OnHitWall(event e):  
    REVERSE TargetSpeed  
    PRINT "Breaking wall? Are we titan now?"
```

```
FUNCTION GoTo(x, y):  
    COMPUTE angle TO (x, y)  
    TURN Left TO Corrected Angle  
    MOVE Forward TO Distance(x, y)
```

```
FUNCTION NormalizeBearing(angle) RETURNS double:  
    WHILE angle > 180: SUBTRACT 360  
    WHILE angle < -180: ADD 360  
    RETURN angle
```

```
FUNCTION TurnGunToTarget(x, y):  
    COMPUTE bearing TO x, y  
    TURN Gun Left TO bearing  
    CALL AlignRadar()
```

```
FUNCTION AlignRadar():  
    COMPUTE radarTurn TO ALIGN Radar WITH Gun  
    TURN Radar Left BY radarTurn
```

```
FUNCTION TurnBodyToTarget(x, y):  
    COMPUTE bearing TO x, y  
    TURN Left TO bearing
```

```
DEFINE CLASS TurnCompleteCondition EXTENDS Condition:  
    VARIABLE bot
```

```
    CONSTRUCTOR TurnCompleteCondition(botInstance):  
        SET bot TO botInstance
```

```
    FUNCTION Test() RETURNS bool:  
        RETURN bot.TurnRemaining == 0
```


B. Struktur data, fungsi dan prosedur yang digunakan

1. Struktur Data

Data yang disimpan dalam bot adalah atribut boolean seperti chasing, lowEnergy, atribut musuh berupa posisi(TargetX dan TargetY) identifikasi musuh (TargetId). Dan Map keyvalue untuk id dan bot musuh yang terdeteksi.

2. Kelas

Ada 2 kelas yang digunakan, 1 sebagai kelas bot dan merupakan kelas utama, yang lainnya adalah kelas helper TurnCompleteCondition, untuk membantu mengatur jalannya perintah. Kelas Oportunis memiliki Konstruktor yang mengambil info dasar dari file JSON, yang di inherit dari Bot. sehingga memiliki member dasar yang sudah disiapkan dari kelas abstrak Bot berupa Run, OnScannedBot, OnHitBot dll.

3. Fungsi

Fungsi yang di implementasikan adalah :

1. Safe

Function ini mengembalikan pasangan value double yang menghitung koordinat X dan Y yang mencari rata rata koordinat semua bot musuh dan mendapat koordinat posisi yang aman.

2. Distance

Function ini mengembalikan jarak dari posisi bot saat ini ke posisi (X, Y) yang menjadi parameter fungsi.

3. NormalizeBearing

Fungsi ini digunakan untuk mengembalikan sudut yang didapat ada pada range -180 dan 180.

4. Prosedur

Prosedur yang di implementasikan adalah :

1. Escape

Prosedur ini dilakukan sebagai movement bot jika dalam kondisi low energy, menggunakan function safe untuk berpindah ke tempat yang aman

2. Dodge

Prosedur ini dilakukan sebagai movement bot dalam keadaan normal, ditujukan untuk melakukan gerakan memutar namun dengan random di sudut putarannya agar tidak monoton

3. Chase

Prosedur ini adalah movement bot yang dipicu jika ada bot dalam radius yang memiliki energy dibawah 40. Bot akan langsung menghadap bot target dan bergerak ke arah koordinat target

4. FirePower

Prosedur ini digunakan untuk menembak jika ingin memperhitungkan variabel jarak didalamnya dengan range 1-3

5. Goto

Memutar bot ke arah tujuan dan maju sebanyak jarak dari posisi bot saat ini ke koordinat tujuan.

6. TurnGunToTarget

Mengkalkulasikan sudut ke posisi musuh dari arah tembak senjata saat ini dan memposisikannya menuju target, dilanjutkan dengan align radar.

7. TurnBodyToTarget

Memutarkan body bot ke arah tujuan dengan sudut dicari menggunakan fungsi bawaan BearingTo

8. AlignRadar

Mencari selisih sudut arah radar dan arah tembak, selanjutnya di normalize lalu dibelokkan

C. Pengujian

1. Pertandingan 1

Results for 10 rounds											
Rank	Name	Total Score	Survival	Surv. Bonus	Bullet Dmg.	Bullet Bonus	Ram Dmg.	Ram Bonus	1sts	2nds	3rds
1	Claude 1.0	1063	550	90	308	40	74	0	3	1	1
2	Dodge 1.0	698	350	30	288	21	8	0	1	3	0
3	Oportunis 1.0	538	200	0	250	10	78	0	1	1	2
4	PersonalBot 1.0	203	100	0	90	0	13	0	0	0	2

2. Pertandingan 2

Results for 10 rounds											
Rank	Name	Total Score	Survival	Surv. Bonus	Bullet Dmg.	Bullet Bonus	Ram Dmg.	Ram Bonus	1sts	2nds	3rds
1	Dodge 1.0	955	350	30	536	17	22	0	2	2	1
2	Claude 1.0	775	400	60	256	15	43	0	2	0	1
3	PersonalBot 1.0	555	250	0	260	22	23	0	0	3	1
4	Oportunis 1.0	460	200	30	172	0	50	8	1	0	2

3. Pertandingan 3

Results for 10 rounds											
Rank	Name	Total Score	Survival	Surv. Bonus	Bullet Dmg.	Bullet Bonus	Ram Dmg.	Ram Bonus	1sts	2nds	3rds
1	Oportunis 1.0	1344	550	60	451	56	199	28	3	1	1
2	Dodge 1.0	912	400	30	452	16	13	0	1	2	1
3	Claude 1.0	832	350	30	316	9	109	17	1	1	3
4	PersonalBot 1.0	294	150	0	90	18	36	0	0	1	0

4. Pertandingan 4

Results for 10 rounds											
Rank	Name	Total Score	Survival	Surv. Bonus	Bullet Dmg.	Bullet Bonus	Ram Dmg.	Ram Bonus	1sts	2nds	3rds
1	Oportunis 1.0	1050	400	30	440	60	119	0	2	0	3
2	Dodge 1.0	981	400	30	480	30	30	10	1	3	0
3	PersonalBot 1.0	800	300	30	410	44	16	0	1	1	1
4	Claude 1.0	728	350	30	280	5	62	0	1	1	1

5. Pertandingan 5

Results for 10 rounds											
Rank	Name	Total Score	Survival	Surv. Bonus	Bullet Dmg.	Bullet Bonus	Ram Dmg.	Ram Bonus	1sts	2nds	3rds
1	Dodge 1.0	1425	550	60	672	80	62	0	2	1	3
2	PersonalBot 1.0	1037	400	30	500	50	56	0	1	2	1
3	Claude 1.0	1032	450	30	380	9	131	32	2	2	1
4	Oportunis 1.0	961	250	30	360	15	306	0	1	1	1

6. Pertandingan 6

Results for 10 rounds											
Rank	Name	Total Score	Survival	Surv. Bonus	Bullet Dmg.	Bullet Bonus	Ram Dmg.	Ram Bonus	1sts	2nds	3rds
1	Oportunis 1.0	682	350	30	212	25	61	3	1	3	0
2	Dodge 1.0	566	150	0	388	20	8	0	1	0	2
3	Claude 1.0	429	200	30	156	4	38	0	1	1	1
4	PersonalBot 1.0	354	200	30	90	10	24	0	1	0	1

D. Analisis

1. Optimal

Hasil yang optimal bisa dilihat pada pertandingan 3 ,4 dan 6, hal ini utamanya karena bot memiliki kemampuan yang cukup merata di semua kategori penilaian, dikarenakan kemampuannya untuk beradaptasi sesuai kondisi energinya saat ini

2. Sub Optimal

Hasil yang sub optimal dapat dilihat pada pertandingan 1, ini disebabkan bot kesulitan merespons jika di target terus menerus oleh 1 bot, dalam kasus ini ada personalBot, meskipun ini tidak semata merta menguntungkan personalBot, melainkan saling menjatuhkan. Selain itu, nilai bullet damage yang dimiliki juga secara umum lebih rendah daripada Dodge, hal ini dimungkinkan karena turn yang ada digunakan juga untuk strategi seperti mengejar atau lari dari musuh. Dapat dilihat pada poin bullet bonus yang umumnya lebih tinggi.

3. Tidak Optimal

Hasil ini utamanya dikarenakan personalBot, kondisi ini terjadi adalah jika Oportunis adalah target pertama dari personalBot dan berhasil melukai oportunis cukup parah bahkan dibunuh. Hal ini juga dapat dilihat jika Oportunis berpoin tinggi maka personal bot akan rendah, hal ini utamanya karena bot lain cukup sulit untuk ditarget oleh personal bot. Oportunis menjadi target yang mudah dikarenakan ada banyak kalkulasi untuk melakukan gerakan sehingga lebih bergerak pasif dan lebih lambat daripada bot lain.

BAB V

KESIMPULAN DAN SARAN

A. Kesimpulan

Dalam tugas besar kali ini, kami sudah mengimplementasikan strategi greedy untuk pengembangan bot untuk Robocode Tank Royale. Algoritma greedy yang dibuat memungkinkan bot untuk membuat keputusan secara lokal optimal dengan harapan dapat mencapai hasil global yang baik.

Beberapa bot yang dirancang memiliki pendekatan yang berbeda dalam menerapkan strategi greedy, seperti Claude yang fokus pada serangan bullet, Dodge yang mengutamakan mobilitas dan penghindaran, PersonalBot yang menargetkan satu musuh secara spesifik, serta Oportunis yang menyesuaikan strategi berdasarkan kondisi pertempuran.

Dari hasil analisis dan pengujian, dapat disimpulkan bahwa masing-masing bot memiliki keunggulan dan kelemahan tergantung pada situasi pertempuran dan karakteristik lawan yang dihadapi. Algoritma greedy terbukti efektif dalam situasi tertentu, tetapi juga memiliki keterbatasan karena tidak mempertimbangkan konsekuensi jangka panjang.

B. Saran

Untuk meningkatkan efektivitas bot, dapat diterapkan strategi yang mengombinasikan algoritma greedy dengan pendekatan lain, seperti algoritma berbasis probabilitas atau machine learning. Hal ini bisa berupa adanya guess factor yang dapat mengumpulkan informasi mengenai perkiraan gerakan musuh. Hal ini memungkinkan bot untuk lebih adaptif terhadap perubahan kondisi pertempuran dan mendapatkan poin dengan lebih efisien.

LAMPIRAN

Link Repositori: https://github.com/Narr21/Tubes1_Tengki-Pertamax.git

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.		✓

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/2024-2025/04-Algoritma-Greedy-(2025)-Bag1.pdf)

<https://robocode-dev.github.io/tank-royale/>