

LAPORAN TUGAS BESAR 1

Strategi Algoritma



Kelompok 28

SATE AMBAL PAK TINO

Anggota Kelompok:

Muhammad Ra'if Alkautsar (13523011)

Orvin Andika Ikhsan A (13523017)

Reza Ahmad Syarif (13523119)

Program Studi Teknik Informatika
Institut Teknologi Bandung

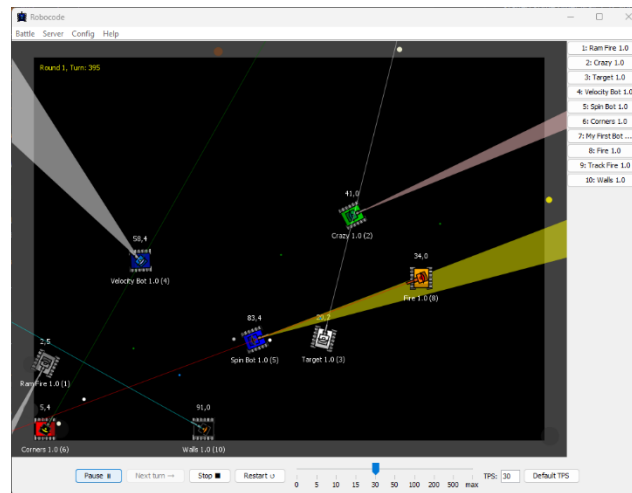
DAFTAR ISI

1	DESKRIPSI TUGAS	4
2	LANDASAN TEORI	8
2.1	Algoritma Greedy	8
2.2	Cara Kerja Program	9
3	APLIKASI STRATEGI GREEDY	11
3.1	Mapping Persoalan Robocode Tank Royale	11
3.2	Alternatif Solusi	11
3.3	Analisis Efisiensi dan Efektivitas Alternatif Solusi	14
3.4	Strategi Greedy Terpilih	15
4	IMPLEMENTASI DAN PENGUJIAN	16
4.1	Implementasi Algoritma Greedy	16
4.2	Penjelasan Struktur Data	23
4.3	Analisis Desain Solusi	24
5	KESIMPULAN	25
5.1	Kesimpulan	25
5.2	Saran	25
5.3	Komentar	25
5.4	Refleksi	25
6	LAMPIRAN	26
6.1	Referensi	26
6.2	Tautan Repository Github	26
6.3	Tautan Video	26

DAFTAR GAMBAR

Gambar 1.1 GUI Robocode Tank Royale.....	4
Gambar 2.1 Inheritansi Kelas Bot	10
Gambar 2.2 Proses Building dari Bot.....	10

1 DESKRIPSI TUGAS



Gambar 1.1
GUI 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.

Komponen-komponen dari permainan ini antara lain:

1. Rounds dan Turns

Pertempuran dapat terdiri dari beberapa rounds. Secara default, satu pertempuran berisi 10 rounds, di mana setiap rounds akan memiliki pemenang dan yang kalah. Setiap round dibagi menjadi beberapa turns, yang merupakan unit waktu terkecil. Satu turn adalah satu ketukan waktu dan satu putaran permainan. Jumlah turn dalam satu round tergantung pada berapa lama waktu yang dibutuhkan hingga hanya tersisa bot terakhir yang bertahan.

Pada setiap turn, sebuah bot dapat:

- Menggerakkan bot, memindai musuh, dan menembakkan senjata.
 - Bereaksi terhadap peristiwa seperti saat bot terkena peluru atau bertabrakan dengan bot lain atau dinding.
- Perintah untuk bergerak, berputar, memindai, menembak, dan sebagainya dikirim ke server untuk setiap turn.

Perlu diperhatikan bahwa API (Application Programming Interface) bot resmi secara otomatis mengirimkan niat bot ke server di balik layar, sehingga Anda tidak perlu mengkhawatirkannya, kecuali jika Anda membuat API Bot sendiri. Pada setiap turn, bot akan secara otomatis menerima informasi terbaru tentang posisinya dan orientasinya di medan perang. Bot juga akan mendapatkan informasi tentang bot musuh ketika mereka terdeteksi oleh pemindai. Perlu diketahui bahwa game engine yang akan digunakan pada tugas besar ini tidak mengikuti aturan default mengenai komponen Round & Turns.

2. Batas Waktu Giliran

Penting untuk dicatat bahwa setiap bot memiliki batas waktu untuk setiap turn yang disebut turn timeout, biasanya antara 30-50 ms (dapat diatur sebagai aturan pertempuran). Ini berarti bahwa bot tidak bisa mengambil waktu sebanyak yang mereka inginkan untuk bergerak dan menyelesaikan turn saat ini. Setiap kali turn baru dimulai, penghitung waktu ulang diatur ulang dan mulai berjalan.

Jika batas waktu tercapai dan bot tidak mengirimkan pergerakannya untuk turn tersebut, maka tidak ada perintah yang dikirim ke server. Akibatnya, bot akan melewati turn tersebut. Jika bot melewati turn, ia tidak akan bisa menyesuaikan gerakannya atau menembakkan senjatanya karena server tidak menerima perintah tepat waktu sebelum turn berikutnya dimulai.

3. Energi

Semua bot memulai permainan dengan jumlah energi awal sebanyak 100 poin energi.

- Bot akan kehilangan energi jika ditembak atau ditabrak oleh bot musuh.
- Bot juga akan kehilangan energi jika menembakkan meriamnya.
- Bot akan mendapatkan energi jika peluru dari meriamnya mengenai musuh. Energi yang didapat akan lebih banyak 3 kali lipat dari energi yang digunakan untuk menembakkan peluru.
- Bot dengan energi nol akan dinonaktifkan dan tidak bisa bergerak. Jika bot terkena serangan dalam keadaan ini, bot akan hancur.

4. Peluru

Semakin banyak energi (daya tembak) yang digunakan untuk menembakkan peluru, semakin berat peluru tersebut dan semakin lambat gerakannya. Namun, peluru yang lebih berat juga menghasilkan lebih banyak kerusakan dan memungkinkan bot mendapatkan lebih banyak energi saat mengenai bot musuh. Seperti disebutkan sebelumnya, peluru yang lebih berat akan bergerak lebih lambat. Ini berarti akan membutuhkan waktu lebih lama untuk mencapai target, meningkatkan risiko peluru tidak mengenai sasaran. Sebaliknya, peluru yang lebih ringan bergerak lebih cepat, sehingga lebih mudah mengenai target, tetapi peluru ringan tidak memberikan banyak poin energi saat mengenai bot musuh.

5. Panas Meriam (Gun Heat)

Saat menembakkan peluru, meriam akan menjadi panas. Peluru yang lebih berat menghasilkan lebih banyak panas dibandingkan peluru yang lebih ringan. Ketika meriam terlalu panas, bot tidak dapat menembak hingga suhu meriam turun ke nol. Selain itu, meriam juga sudah dalam keadaan panas di awal round dan perlu waktu untuk mendingin sebelum bisa digunakan untuk pertama kalinya.

6. Tabrakan

Perlu diperhatikan bahwa bot akan menerima kerusakan jika menabrak dinding (batas arena), yang disebut wall damage. Hal yang sama juga terjadi jika bot bertabrakan dengan bot lain. Jika bot menabrak bot musuh dengan bergerak maju, ini disebut ramming (menabrak dengan sengaja), yang akan memberikan sedikit skor tambahan bagi bot yang menyerang.

7. Bagian Tubuh Tank

Tubuh tank terdiri dari 3 bagian:

- Body adalah bagian utama dari tank yang digunakan untuk menggerakkan tank.
- Gun digunakan untuk menembakkan peluru dan dapat berputar bersama body atau independen dari body.
- Radar digunakan untuk memindai posisi musuh dan dapat berputar bersama body atau independen dari body.

8. Pergerakan

Bot dapat bergerak maju dan mundur hingga kecepatan maksimum. Dibutuhkan beberapa giliran untuk mencapai kecepatan maksimum. Bot dapat mengalami percepatan maksimum sebesar 1 unit per giliran dan pengereman dengan perlambatan maksimum 2 unit per giliran. Percepatan dan perlambatan maksimum tidak bergantung pada kecepatan bot saat itu.

9. Berbelok

Seperti yang disebutkan sebelumnya, bagian tubuh, turret (meriam), dan radar dapat berputar secara independen satu sama lain. Jika turret atau radar tidak diputar, maka keduanya akan mengarah ke arah yang sama dengan tubuh bot. Setiap bagian tubuh memiliki kecepatan putar yang berbeda. Radar adalah bagian tercepat dan dapat berputar hingga 45 derajat per giliran, yang berarti dapat berputar 360 derajat dalam 8 giliran. Turret dan meriam dapat berputar hingga 20 derajat per giliran. Bagian paling lambat adalah tubuh tank, yang dalam kondisi terbaik dapat berputar hingga 10 derajat per giliran. Namun, ini bergantung pada kecepatan bot saat ini. Semakin cepat bot bergerak, semakin lambat kemampuannya untuk berbelok. Perlu diperhatikan bahwa tidak ada energi yang dikonsumsi saat bot bergerak atau berbelok.

10. Pemindaian

Aspek penting dalam Robocode adalah memindai bot musuh menggunakan radar. Radar dapat mendeteksi bot dalam jangkauan hingga 1200 piksel. Musuh yang berada lebih dari 1200 piksel dari bot tidak dapat terdeteksi atau dipindai oleh

radar. Penting untuk diperhatikan bahwa sebuah bot hanya dapat memindai bot musuh yang berada dalam jangkauan sudut pemindaian (scan arc)-nya. Sudut pemindaian ini merupakan "sapuan radar" dari arah radar sebelumnya ke arah radar saat ini dalam satu giliran. Jika radar tidak bergerak dalam suatu giliran, artinya radar tetap mengarah ke arah yang sama seperti pada giliran sebelumnya, maka sudut pemindaian akan menjadi nol derajat, dan bot tidak akan dapat mendeteksi musuh. Oleh karena itu, sangat disarankan untuk selalu mengubah arah radar agar tetap dapat memindai musuh.

11. Skor

Pada akhir pertempuran, setiap bot akan diranking berdasarkan total skor yang diperoleh masing-masing bot selama keseluruhan pertempuran. Tentunya, tujuan utama pada tugas besar ini adalah membuat bot yang memberikan skor setinggi mungkin. Berikut adalah rincian komponen skor pada pertempuran:

- Bullet Damage: Bot mendapatkan poin sebesar damage yang dibuat kepada bot musuh menggunakan peluru.
- Bullet Damage Bonus: Apabila peluru berhasil membunuh bot musuh, bot mendapatkan poin sebesar 20% dari damage yang dibuat kepada musuh yang terbunuh.
- Survival Score: Setiap ada bot yang mati, bot lainnya yang masih bertahan pada ronde tersebut mendapatkan 50 poin.
- Last Survival Bonus: Bot terakhir yang bertahan pada suatu ronde akan mendapatkan 10 poin dikali dengan banyaknya musuh.
- Ram Damage: Bot mendapatkan poin sebesar 2 kalinya damage yang dibuat kepada bot musuh dengan cara menabrak.
- Ram Damage Bonus: Apabila musuh terbunuh dengan cara ditabrak, bot mendapatkan poin sebesar 30% dari damage yang dibuat kepada musuh yang terbunuh.

Skor akhir bot adalah akumulasi dari 6 komponen diatas. Perlu diperhatikan bahwa game akan menampilkan berapa kali suatu bot meraih peringkat 1, 2, atau 3 pada setiap ronde. Namun, hal ini tidak dihitung sebagai komponen skor maupun untuk perankingan akhir. Bot yang dianggap menang pertempuran adalah bot dengan akumulasi skor tertinggi.

2 LANDASAN TEORI

2.1 Algoritma Greedy

Algoritma greedy adalah algoritma yang memecahkan persoalan secara langkah per langkah sedemikian sehingga pada setiap langkah:

1. Mengambil pilihan yang terbaik yang dapat diperoleh pada saat itu tanpa memperhatikan konsekuensi ke depan (prinsipio “*take what you can get now!*”)
2. Berharap bahwa dengan memilih optimum local pada setiap langkah akan berakhir dengan optimum global.

Dalam implementasinya, algoritma greedy memiliki beberapa elemen yang dapat digunakan untuk mempermudah perancangan algoritma. Elemen-elemen tersebut di antaranya:

1. Himpunan Kandidat (C)
Berisi kandidat yang akan dipilih pada setiap langkah, misalnya simpul/sisi dalam graf, job, task, koin, benda, karakter, dan sebagainya.
2. Himpunan Solusi (S)
Berisi kandidat yang sudah dipilih oleh algoritma.
3. Fungsi Solusi
Fungsi yang digunakan untuk menentukan apakah himpunan kandidat yang dipilih sudah memberikan solusi.
4. Fungsi Seleksi (Selection Function)
Fungsi untuk memilih kandidat berdasarkan strategi greedy tertentu. Strategi greedy ini bersifat heuristik yang tentunya berbeda untuk setiap masalah yang dihadapi.
5. Fungsi Kelayakan (Feasible)
Fungsi untuk memeriksa apakah kandidat yang dipilih dapat dimasukkan ke dalam himpunan solusi atau tidak.
6. Fungsi Objektif
Fungsi untuk mengoptimalkan solusi, bisa dengan memaksimumkan atau meminimumkan target.

Berdasarkan elemen-elemen tersebut dapat disimpulkan bahwa algoritma greedy melibatkan pencarian sebuah himpunan bagian S dari himpunan kandidat C, dan S harus memenuhi beberapa kriteria yang ditentukan, yaitu S menyatakan suatu solusi dan S dioptimalisasi oleh fungsi objektif.

Algoritma greedy bukanlah algoritma yang sempurna. Salah satu kekurangannya adalah optimum global yang dibentuk dari pemilihan optimum lokal belum tentu merupakan solusi optimum atau terbaik. Bisa jadi optimum global hanyalah solusi sub-optimum atau pseudo-optimum. Hal ini bisa terjadi karena dua alasan, yaitu:

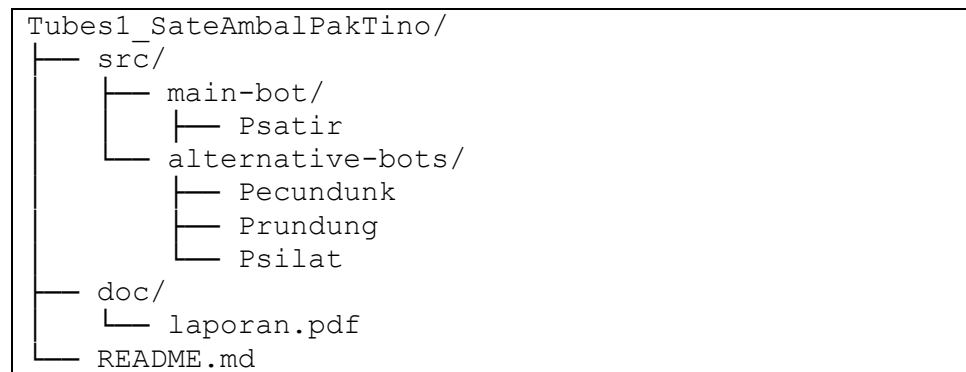
1. Algoritma greedy tidak beroperasi secara menyeluruh terhadap semua kemungkinan solusi yang ada, berbeda dengan metode *exhaustive search*.
2. Terdapat berbeda fungsi seleksi yang berbeda sehingga kita harus memilih fungsi yang tepat jika kita ingin algoritma menghasilkan solusi optimal.

Keoptimalan solusi algoritma greedy harus bisa dibuktikan secara matematis. Namun, membuktikan optimalitas algoritma greedy secara matematis merupakan hal yang cukup rumit. Lebih mudah memperlihatkan algoritma greedy tidak selalu optimal dengan *counterexample* atau contoh kasus yang menunjukkan solusi yang diperoleh tidak optimal.

2.2 Cara Kerja Program

1. Struktur Program

Program Bot yang dibuat pada tugas besar 1 strategi algoritma ini memiliki struktur program pada repository sebagai berikut.



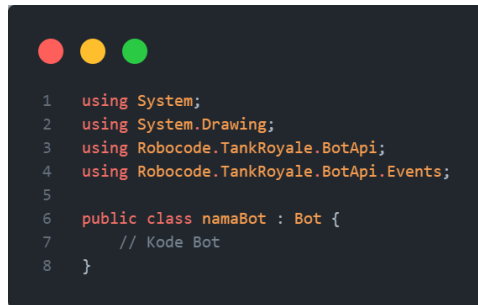
2. Proses Pelaksanaan Aksi oleh Bot

Bot dalam Robocode Tank Royale berjalan dalam event-driven loop berbasis turns. Setiap bot akan menerima informasi dari server, merespons informasi, lalu mengirim kembali aksi atas informasi yang didapatkan ke server. Berikut adalah siklus yang terjadi pada setiap turn.

- Server Game akan mengirimkan informasi status terbaru kepada bot. Informasi ini bisa berupa posisi musuh, status energi dan arah, serta event yang terjadi, seperti musuh terlihat, terkena tembakan, menabrak dinding, dan lain sebagainya.
- Bot menjalankan kode logika untuk memproses informasi dan event. Kemudian, menetapkan aksi yang akan dilakukan sebagai respons dari informasi atau event yang didapatkan. Lalu, mengirimkan kembali aksi balasan tersebut ke server.
- Server mengeksekusi aksi dan mengupdate arena. Hasil dari eksekusi aksi akan membuat bot bergerak, menembak, menabrak, dan sebagainya. Server akan mengupdate keadaan dan mengirim kembali ke bot. Begitu seterusnya untuk turn berikutnya.

3. Implementasi Algoritma Greedy ke Dalam Bot

Untuk mengimplementasikan algoritma greedy ke dalam sebuah bot, perlu dibuat sebuah file .cs, misalnya Psatir.cs, yang didalamnya memuat logika bot dan algoritma greedy yang diterapkan. Dalam file ini, bot harus meng-inherit Class Bot dari Robocode.TankRoyale.BotApi.



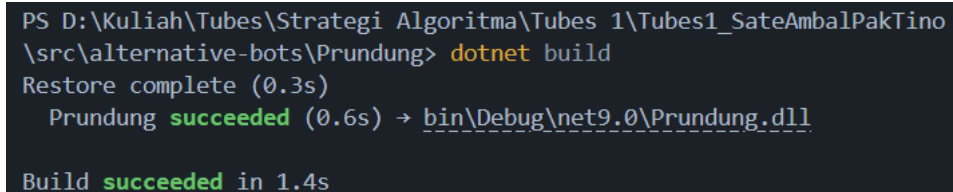
```

1  using System;
2  using System.Drawing;
3  using Robocode.TankRoyale.BotApi;
4  using Robocode.TankRoyale.BotApi.Events;
5
6  public class namaBot : Bot {
7      // Kode Bot
8  }

```

Gambar 2.1
Inheritansi Kelas Bot

Di dalam file ini, terdapat berbagai logika dan event handler untuk melakukan respons terhadap informasi yang didapatkan dari server. Setiap bot kemudian dibangun secara independen menjalankan perintah pada direktori bot berikut sebagai contoh.



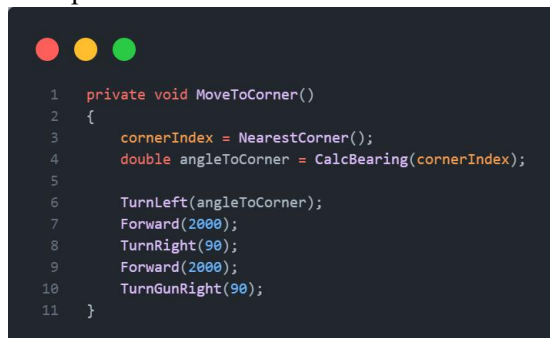
```

PS D:\Kuliah\Tubes\Strategi Algoritma\Tubes 1\Tubes1_SateAmbalPakTino
\src\alternative-bots\Prundung> dotnet build
Restore complete (0.3s)
Prundung succeeded (0.6s) → bin\Debug\net9.0\Prundung.dll
Build succeeded in 1.4s

```

Gambar 2.2
Proses Building dari Bot

Algoritma greedy akan menentukan keputusan terbaik dalam setiap langkahnya, tanpa mempertimbangkan konsekuensi jangka panjang. Dalam robocode tank royale, fungsi objektif yaitu memperoleh skor setinggi mungkin pada akhir pertempuran. Setiap strategi greedy yang diterapkan menggunakan heuristik yang berbeda-beda. Misalnya, bot bergerak ke corner terdekat atau menembak musuh yang terlihat langsung. Selain itu, terdapat juga fungsi seleksi dan fungsi kelayakan yang akan menentukan apakah aksi tersebut layak dan sesuai dengan greedy yang diterapkan. Contohnya, dengan fungsi NearestCorner() akan diseleksi corner yang terdekat dari arah hadap bot.



```

1  private void MoveToCorner()
2  {
3      cornerIndex = NearestCorner();
4      double angleToCorner = CalcBearing(cornerIndex);
5
6      TurnLeft(angleToCorner);
7      Forward(2000);
8      TurnRight(90);
9      Forward(2000);
10     TurnGunRight(90);
11 }

```

3 APLIKASI STRATEGI GREEDY

3.1 Mapping Persoalan Robocode Tank Royale

Untuk mempermudah perancangan algoritma greedy diperlukan mapping elemen algoritma greedy ke permasalahan, yang dalam hal ini adalah permainan Robocode Tank Royale. Berikut adalah pemetaannya:

1. Himpunan Kandidat (C)
Himpunan kandidat pada permainan ini akan berisi strategi yang akan diterapkan dalam permainan. Strategi tersebut mencakup lawan mana yang akan diserang, arah pergerakan yang akan dipilih, strategi menghindari serangan, dan keputusan menyesuaikan sudut rotasi.
2. Himpunan Solusi (S)
Himpunan solusi pada permainan ini akan berisi keputusan-keputusan yang sudah diambil, misalnya arah gerak yang dipilih, serangan yang dilakukan, dan strategi pertahanan yang diterapkan.
3. Fungsi Solusi
Fungsi solusi digunakan untuk mengevaluasi apakah keputusan yang diambil sudah cukup untuk bertahan paling lama atau meraih poin paling banyak.
4. Fungsi Seleksi
Fungsi seleksi digunakan untuk memilih tindakan berdasarkan strategi greedy yang diterapkan pada tiap bot.
5. Fungsi Kelayakan
Fungsi kelayakan digunakan untuk mengecek apakah keputusan yang diambil masih memungkinkan sesuai aturan dari permainan. Contohnya, bot tidak akan bisa melakukan apa pun jika energi habis, maka bot tidak boleh menembak jika energi terlalu sedikit.
6. Fungsi Objektif
Fungsi objektif dalam permainan ini adalah memperoleh skor setinggi mungkin di akhir pertempuran dengan mengoptimalkan komponen penilaian yang ada.

3.2 Alternatif Solusi

1. Algoritma Greedy Menyerang Lawan Jarak Dekat
Ide dari strategi ini adalah memaksimalkan poin bullet damage dengan memperbesar peluang tiap peluru mengenai target. Hal ini dilakukan dengan menyerang lawan yang berjarak dekat saja. Apabila lawan yang terdeteksi terlalu jauh maka bot akan mendekat terlebih dahulu.
 - a. Himpunan Kandidat (C)
Himpunan kandidat dalam strategi ini adalah semua lawan yang terdeteksi. Setiap kali musuh terdeteksi, bot akan melakukan evaluasi terhadap posisi mereka dan menentukan tindakan yang harus dilakukan.
 - b. Himpunan Solusi (S)
Himpunan solusi dalam algoritma ini adalah bot-bot musuh yang berada pada jarak yang dekat. Bot hanya akan menyerang musuh yang berjarak dekat

karena strategi utamanya adalah mendekati dan menyerang musuh dengan kekuatan penuh.

c. Fungsi Solusi

Fungsi solusi dalam strategi ini adalah mengevaluasi apakah bot lawan sudah berada dalam jarak tembak optimal. Jika sudah, bot akan menembak dengan kekuatan penuh. Jika belum, bot akan mendekati musuh.

d. Fungsi Seleksi

Fungsi seleksi dalam algoritma ini menentukan bagaimana bot memilih target yang akan diserang. Jika ada lebih dari satu musuh, bot akan menggunakan jarak sebagai faktor utama, mendekati musuh hingga masuk dalam jarak tembak optimal. Jika hanya ada satu musuh, bot akan menggunakan pendekatan yang lebih agresif dengan menabrak dan menembak bot musuh.

e. Fungsi Kelayakan

Fungsi kelayakan dalam strategi ini memastikan bahwa target yang dipilih masih valid untuk diserang.

f. Fungsi Objektif

Fungsi objektif pada strategi ini adalah memaksimalkan poin di akhir pertarungan dengan memprioritaskan bullet damage.

2. Algoritma Greedy Survival

Ide dari strategi ini adalah memaksimalkan poin survival. Hal ini dilakukan dengan melakukan gerakan *random* sehingga serangan lawan akan sulit untuk mengenai bot.

a. Himpunan Kandidat (C)

Himpunan kandidat pada strategi ini adalah pola gerakan yang dapat diambil oleh bot.

b. Himpunan Solusi

Himpunan solusi pada strategi ini adalah pergerakan yang sudah dipilih oleh bot untuk menghindari *damage*, baik dengan bergerak secara *random* untuk menghindari tembakan musuh atau bergerak mundur ketika menabrak tembok.

c. Fungsi Solusi

Fungsi solusi pada strategi ini memastikan bahwa bot selalu dalam keadaan bergerak dan sulit untuk ditembak musuh.

d. Fungsi Seleksi

Fungsi seleksi pada algoritma ini memilih arah dan pola pergerakan untuk menghindari tembakan lawan dan tembok.

e. Fungsi Kelayakan

Fungsi kelayakan pada strategi ini memastikan bot tidak diam di tempat terlalu lama dan memastikan bot tidak terus menabrak dinding.

f. Fungsi Objektif

Fungsi objektif pada strategi ini adalah memaksimalkan poin di akhir pertarungan dengan memprioritaskan komponen survival.

3. Algoritma Melakukan Ramming pada Musuh Berenergi Rendah

Ide dari strategi ini adalah untuk memaksimalkan poin yang diperoleh dari tindakan ramming terhadap bot musuh. Melakukan ramming terhadap bot musuh memberikan dua kali dari damage yang dilakukan dan membunuh bot musuh memberikan ram damage bonus berupa 30% dari damage yang telah dilakukan terhadap bot yang dibunuh.

- a. Himpunan Kandidat
Himpunan kandidat dari strategi ini adalah semua “gerakan” atau target yang bisa dipilih dalam sebuah turn. Dalam strategi ini, kandidat yang ada yaitu:
 - menembak dengan energi tertentu,
 - mengejar atau menge-*ram* musuh,
 - bergerak ke tengah arena (jika menabrak tembok), atau
 - bergerak dalam pola melingkar (pola default).
 - b. Himpunan Solusi
Himpunan solusi dari strategi bisa didefinisikan sebagai tindakan yang akan dilakukan oleh bot untuk suatu turn. Namun, bisa juga didefinisikan sebagai semua tindakan yang telah bot lakukan untuk seluruh ronde.
Dalam kasus himpunan solusi sebagai semua tindakan yang akan dilakukan oleh bot dalam suatu turn, himpunan tersebut bisa jadi terdiri atas rencana saat ini yang terdiri atas berbelok ke arah tertentu, maju dalam kecepatan tertentu, atau menembak dengan energi tertentu.
 - c. Fungsi Solusi
Fungsi solusi menanyakan “apakah semua aksi yang telah dilakukan saat ini bisa memberikan poin terbanyak?”
 - d. Fungsi Seleksi
Fungsi seleksi dalam algoritma ini: jika musuh memiliki energi ≤ 24 , kejar dan ram; jika energi musuh tinggi, tembak. Aksi terbaik yang dilakukan ditentukan secara heuristik (greedy).
 - e. Fungsi Kelayakan
Fungsi memeriksa kelayakan kandidat aksi dengan mencari tahu apakah aksi yang dipilih melanggar *constraint* atau batasan. Batasan yang ada dalam algoritma ini antara lain: kecukupan energi ketika menembak dan keberadaan tembok ketika bergerak.
 - f. Fungsi Objektif
Bot ingin memaksimalkan skor. Dalam algoritma ini, pemaksimalan tersebut dilakukan dengan strategi *ramming*.
4. Algoritma Greedy Menuju Corner dan *Lock* Musuh
Ide dari strategi ini adalah memaksimalkan poin bertahan hidup lebih lama untuk mendapatkan survival score dengan bergerak ke corner dan menge-*lock* musuh hingga mati atau tidak terdeteksi untuk mendapatkan *bullet damage*.
 - a. Himpunan Kandidat (C)
Himpunan kandidat dari strategi ini adalah semua posisi corner (pojok arena) dan semua bot musuh yang terdeteksi melalui radar. Bot akan menilai 4 titik corner arena sebagai potensi lokasi strategis dan mempertimbangkan musuh yang terdeteksi oleh radar.
 - b. Himpunan Solusi
Himpunan solusi dari algoritma greedy ini adalah corner terbaik yaitu yang terdekat dari posisi arah hadap bot saat ini dan target musuh yang terakhir terdeteksi serta mungkin bisa ditembak secara akurat dan efisien berdasarkan posisi dan arah senjata.
 - c. Fungsi Solusi
Fungsi solusi ini memutuskan kapan dan bagaimana bot akan berpindah ke corner terdekat dan menembak musuh.

- d. Fungsi Seleksi
Fungsi seleksi pada algoritma ini bertanggung jawab untuk menentukan lokasi corner yang dipilih serta musuh yang ditembak. Corner terdekat dipilih berdasarkan arah hadap saat ini menggunakan method NearestCorner(). Musuh ditembak hanya jika GunBearingTo mendekati nol derajat (≤ 3) serta memperhitungkan jarak dan energi bot.
- e. Fungsi Kelayakan
Fungsi kelayakan dalam algoritma ini memastikan bahwa musuh yang ditarget masih valid dengan dihitung menggunakan targetLostTurns dan tembakan dilakukan hanya jika gun sudah mengarah serta tidak panas.
- f. Fungsi Objektif
Fungsi objektif pada startegi ini adalah memaksimalkan poin di akhir pertarungan dengan memprioritaskan komponen survival dan bullet damage.

3.3 Analisis Efisiensi dan Efektivitas Alternatif Solusi

Analisis efisiensi dan efektivitas algoritma dapat dilakukan dengan tes performa tiap bot. Proses pengetesan dapat dilakukan dengan menguji bot melawan bot-bot sampel yang sudah tersedia. Teknis pengujian adalah bot yang diuji akan diadu dengan format 1 vs 1 vs 1 vs 1 dengan bot sampel. Bot sampel yang dipilih adalah SpinBot, TrackFire, dan Corners. Pengetesan tersebut diharapkan dapat menjadi pengukuran objektif terhadap efisiensi dan efektivitas algoritma *greedy*.

Berikut adalah hasil pengetesan algoritma greedy lawan jarak dekat:

Tabel 3.3.1 Poin Algoritma Greedy Lawan Jarak Dekat

Percobaan	Poin
1	757
2	686
3	767
4	801
5	484
Rata-rata	699

Berikut adalah hasil pengetesan algoritma greedy survival:

Tabel 3.3.2 Poin Algoritma Greedy Survival

Percobaan	Poin
1	220
2	280
3	209
4	288
5	236
Rata-rata	246,6

Berikut adalah hasil pengetesan algoritma greedy ram musuh dengan energi rendah:

Tabel 3.3.3 Poin Algoritma Ramming Musuh Energi Rendah

Percobaan	Poin
1	826
2	946
3	592
4	638
5	639
Rata-rata	728,2

Berikut adalah hasil pengetesan algoritma greedy corner dan *lock* musuh:

Tabel 3.3.1 Poin Algoritma Greedy Corner dan *Lock* Musuh

Percobaan	Poin
1	610
2	527
3	851
4	606
5	594
Rata-rata	637,6

3.4 Strategi Greedy Terpilih

Berdasarkan data yang diambil pada pengetesan, strategi greedy yang dipilih adalah algoritma *greedy ramming* musuh dengan energi rendah. Algoritma ini dipilih karena menghasilkan rata-rata poin tertinggi dalam pengujian. Pergerakan bot ini juga bisa terbilang terbaik di antara yang lain karena sangat efektif untuk menyerang dan menghindari serangan lawan. Agresivitasnya saat ada musuh dengan energi rendah juga membuat algoritma ini memiliki potensi mendapat *bullet damage bonus* yang sangat tinggi.

4 IMPLEMENTASI DAN PENGUJIAN

4.1 Implementasi Algoritma Greedy

1. Implementasi Algoritma Greedy Melawan Musuh Jarak Dekat

```
BEGIN Psilat

    // Variabel untuk menyimpan arah putaran bot
    (1 = kiri, -1 = kanan)
    SET turnDirection TO 1

    // Variabel untuk menyimpan jumlah musuh yang
    terdeteksi
    SET enemies TO 0

    // Fungsi utama untuk menjalankan bot
    FUNCTION Main()
        // Membuat instance dari bot Psilat
        CREATE INSTANCE Psilat

        // Memulai eksekusi bot
        CALL Start()

    // Konstruktor untuk menginisialisasi bot
    dengan konfigurasi
    FUNCTION Psilat()
        LOAD BOT CONFIGURATION FROM "Psilat.json"

    // Fungsi utama yang dijalankan terus-menerus
    saat bot aktif
    FUNCTION Run()
        // Mengatur warna bot
        SET BodyColor TO Black
        SET TurretColor TO White
        SET RadarColor TO Black

        // Jika tidak ada musuh yang terdeteksi,
        bot akan berputar mencari musuh
        WHILE Bot is Running
            TURN_LEFT(5 * turnDirection) // Bot
            terus berputar ke kiri atau kanan
            SET enemies TO EnemyCount //
            Menyimpan jumlah musuh yang terdeteksi

        // Fungsi yang dipanggil saat bot mendeteksi
        musuh
        FUNCTION OnScannedBot(event)
            // Bot menghadap ke arah musuh yang
            terdeteksi
            CALL TurnToFaceTarget(event.X, event.Y)
```



```

        // Menghitung jarak ke musuh
        SET distance TO DistanceTo(event.X,
event.Y)

        // Jika terdapat lebih dari satu musuh
        IF enemies > 1 THEN
            // Jika jarak ke musuh lebih dari 200,
            bot maju setengah jarak
            IF distance > 200 THEN
                MOVE_FORWARD(distance / 2)
            // Jika jarak kurang dari atau sama
            dengan 200, bot menembak
            ELSE
                FIRE(3)
            // Jika hanya ada satu musuh, bot akan
            mencoba menabrak
            ELSE
                MOVE_FORWARD(distance + 5)

        // Memindai ulang untuk memastikan target
        tetap terkunci
        CALL Rescan()

        // Fungsi yang dipanggil ketika bot
        bertabrakan dengan musuh
        FUNCTION OnHitBot(event)
            // Bot akan kembali menghadap ke arah
            musuh
            CALL TurnToFaceTarget(event.X, event.Y)

        // Menyesuaikan kekuatan tembakan
        berdasarkan energi musuh
        IF event.Energy > 15 THEN FIRE(3) //
        Tembakan kuat jika musuh punya energi tinggi
        ELSE IF event.Energy > 9 THEN FIRE(2)
        ELSE IF event.Energy > 6 THEN FIRE(1)
        ELSE IF event.Energy > 3 THEN FIRE(0.5)
        ELSE IF event.Energy > 1.5 THEN FIRE(0.1)
        // Tembakan lemah jika musuh hampir mati

        // Bot maju untuk kembali menabrak musuh
        MOVE_FORWARD(40)

        // Fungsi yang dipanggil jika bot menabrak
        dinding
        FUNCTION OnHitWall(event)
            // Bot akan mundur sedikit untuk
            menghindari dinding
            MOVE_BACKWARD(50)

```

```

        // Bot akan berputar 90 derajat untuk
mengubah arah
        TURN_RIGHT(90)

        // Menjalankan pergerakan setelah mengubah
arah
        EXECUTE_MOVE()

        // Fungsi untuk mengubah arah bot agar
menghadap ke target
        FUNCTION TurnToFaceTarget(x, y)
            // Menghitung sudut yang perlu dituju
untuk menghadap musuh
            SET bearing TO BearingTo(x, y)

            // Jika sudut positif, putar ke kiri, jika
negatif, putar ke kanan
            IF bearing >= 0 THEN
                SET turnDirection TO 1
            ELSE
                SET turnDirection TO -1

            // Putar bot ke arah musuh
            TURN_LEFT(bearing)

        END Psilat

```

2. Implementasi Algoritma Greedy Survival

```

BEGIN Pecundunk

// Variabel untuk menyimpan jumlah musuh yang
terdeteksi
SET enemies TO 0

// Variabel untuk mengecek apakah bot sedang
bergerak maju atau mundur
SET movingForward TO true

// Variabel untuk menyimpan energi bot
SET myEnergy TO 0

// Variabel penghitung iterasi (digunakan untuk
pola gerakan)
SET tickCounter TO 0

// Fungsi utama untuk menjalankan bot
FUNCTION Main()
// Membuat instance dari bot Pecundunk
CREATE INSTANCE Pecundunk

```

```

// Memulai eksekusi bot
CALL Start()

// Konstruktor untuk menginisialisasi bot dengan konfigurasi
FUNCTION Pecundunk()
LOAD BOT CONFIGURATION FROM "Pecundunk.json"

// Fungsi utama yang dijalankan terus-menerus saat bot aktif
FUNCTION Run()
// Mengatur warna bot
SET BodyColor TO Pink
SET TurretColor TO White
SET RadarColor TO Pink

// Menyimpan jumlah musuh yang terdeteksi dan energi bot
SET enemies TO EnemyCount
SET myEnergy TO Energy

// Menandakan bahwa bot sedang bergerak maju
SET movingForward TO true

// Loop utama bot saat permainan berlangsung
WHILE Bot is Running
INCREMENT tickCounter // Menambah iterasi

IF enemies > 1 THEN
// Jika ada lebih dari satu musuh, bergerak zig-zag untuk menghindari
SET_FORWARD(200 + (tickCounter MOD 100))
SET_TURN_RIGHT(45 - (tickCounter MOD 90))
ELSE
// Jika hanya ada satu musuh, tetap menghindari tetapi sesekali menembak
SET_FORWARD(150)
SET_TURN_RIGHT(30)

// Menembak setiap 20 iterasi
IF tickCounter MOD 20 == 0 THEN
FIRE(1)

EXECUTE_MOVE()

// Fungsi yang dipanggil saat bot mendeteksi musuh
FUNCTION OnScannedBot(event)
IF enemies == 1 THEN
SET distance TO DistanceTo(event.X, event.Y)

```

```

// Jika musuh sangat dekat, tembak dengan kekuatan
penuh
IF distance < 100 THEN
FIRE(3)
ELSE
FIRE(1) // Jika musuh jauh, tembak dengan kekuatan
kecil

// Fungsi yang dipanggil jika bot bertabrakan
dengan musuh
FUNCTION OnHitBot(event)
IF enemies == 1 THEN
// Jika energi bot lebih besar dari musuh, tembak
dengan kekuatan penuh
IF myEnergy > event.Energy THEN
FIRE(3)
CALL Evade()
ELSE
CALL Evade() // Jika energi lebih rendah,
menghindar

// Fungsi yang dipanggil jika bot menabrak dinding
FUNCTION OnHitWall(event)
// Jika menabrak dinding, ubah arah agar tidak
terjebak
SET movingForward TO NOT movingForward
SET_TURN_RIGHT(90)

// Jika sebelumnya maju, sekarang mundur, dan
sebaliknya
IF movingForward THEN
SET_BACK(200)
ELSE
SET_FORWARD(200)

EXECUTE_MOVE()

// Fungsi yang dipanggil jika bot terkena tembakan
FUNCTION OnHitByBullet(event)
// Jika tertembak, ubah arah agar lebih sulit
ditembak lagi
SET_TURN_RIGHT(60)
SET_BACK(100)
EXECUTE_MOVE()

// Fungsi untuk menghindari jika bertabrakan atau
terkena tembakan
FUNCTION Evade()
SET_TURN_RIGHT(90)
SET_BACK(150)
EXECUTE_MOVE()

```

```
END Pecundunk
```

3. Implementasi Algoritma Greedy Ramming Musuh dengan Energi Rendah

```
START Psatir

    SET color bodi = Biru
    SET color turret = Biru
    SET color radar = Hitam
    SET color scan = Kuning
    SET centerX = ArenaWidth / 2
    SET centerY = ArenaHeight / 2
    SET hittingWall = false

    WHILE Bot masih hidup:
        IF tidak menabrak dinding:
            SET belok ke kiri 1000 derajat
            SET maju sejauh 1000
            GO (jalankan perintah)
        ELSE
            Panggil fungsi HandleHittingWall()

    ON ScannedBot(event):
        hitung jarak ke musuh (distance)

        IF distance < 100:
            SET firepower = 3
        ELSE IF distance > 300:
            SET firepower = 0.5
        ELSE:
            SET firepower = 3 - ((distance - 100) /
200 * 2.5)

        IF energi musuh < 20:
            belok ke arah musuh (TurnToFaceTarget)
            maju ke arah musuh (SetForward)
            SetRescan (agar scan lagi setelah
selesai)

        IF energi sendiri > 20:
            Tembak dengan firepower
            GO (jalankan perintah)

    ON HitBot(event):
        belok ke arah musuh
        tentukan kekuatan tembak berdasarkan energi
musuh
        maju 40 untuk menabrak
        GO
```

```

ON HitWall:
    SET hittingWall = true
    GO

HandleHittingWall():
    IF jarak ke tengah > 100:
        belok ke tengah arena
        maju 10000
    ELSE:
        SET hittingWall = false
    GO

TurnToFaceTarget(x, y):
    hitung bearing ke posisi x, y
    belok ke arah musuh
    GO

END Psatir

```

4. Implementasi Algoritma Greedy Corner dan *Lock* Musuh

```

START Prundung

    SET warna bot dan senjata

    SET cornerIndex = corner terdekat berdasarkan
    arah bot
    PANGGIL MoveToCorner()

    WHILE bot masih hidup DO
        IF hasTarget = true THEN
            HITUNG bearingFromGun ke targetX,
            targetY
            HITUNG distance ke targetX, targetY

            ARAHKAN senjata ke target
            TENTUKAN firepower berdasarkan jarak:
                - Jika dekat (<100) → firepower = 3
                - Jika sedang (<300) → firepower =
2                - Jika jauh → firepower = 1

            IF arah senjata sudah cukup akurat dan
            senjata tidak panas THEN
                TEMBAK dengan firepower sesuai
            energi

            INCREMENT counter targetLostTicks
            IF targetLostTicks > 20 THEN
                RESET hasTarget dan targetLostTicks
            ELSE

```

```

        PUTAR senjata ke kiri (scan)

    END IF
END WHIL

ON ENEMY DETECTED (OnScannedBot):
    SET hasTarget = true
    SET koordinat target = posisi musuh
    RESET targetLostTicks = 0

ON HIT BY BULLET:
    Pindah ke corner berikutnya (putar indeks)
    Reset target
    Panggil MoveToCorner()
    Scan ulang (putar radar)

ON WON ROUND:
    Lakukan victory dance dengan memutar ke
    kiri 36000 derajat

FUNGSI NearestCorner():
    Cek sudut bot saat ini
    Bandingkan dengan 4 corner: 0, 90, 180, 270
    RETURN sudut corner yang paling dekat

FUNGSI MoveToCorner():
    Hitung sudut belok ke corner
    Belok ke arah corner
    Maju ke dinding pertama (2000)
    Belok kanan 90 derajat
    Maju ke sudut corner (2000)
    Putar senjata 90 derajat (siap scanning)

END Prundung

```

4.2 Penjelasan Struktur Data

Program ini menggunakan beberapa struktur data yang telah tersedia di dalam game dan beberapa variabel yang memanfaatkan struktur data tersebut. Berikut adalah rincian struktur data dan variabel yang dipakai dalam program.

1. Struktur data dan Variabel

```

bool hittingWall; // menyimpan state apakah bot
menabrak dinding
double centerX, centerY; // posisi tengah arena
double firepower; // besar kekuatan peluru
var distance; // jarak bot dengan musuh atau titik
var bearing; // arah tembak
ScannedBotEvent e; // objek event dari musuh yang
terdeteksi

```

```
HitBotEvent e; // objek dari musuh yang tertabrak
HitWallEvent e; // objek dari dinding yang tertabrak
```

2. Fungsi dan Prosedur

```
Run(); // fungsi utama bot
OnScannedBot(ScannedBotEvent e); // fungsi untuk
menghandle hasil radar
OnHitBot(HitBotEvent e); // fungsi untuk menghandle
ketika menabrak musuh
TurnToFaceTarget(double x, double y); // fungsi
untuk mengatur arah hadap bot ke target
OnHitWall(HitWallEvent e); // fungsi untuk
menghandle ketika menabrak dinding
HandleHittingWall(); // fungsi untuk menghandle
ketika akan menabrak dinding
```

4.3 Analisis Desain Solusi

Pengujian terhadap algoritma *greedy* yang dipilih dilakukan dengan pertandingan antara algoritma terpilih dan semua algoritma yang dibuat dengan format 1 vs 1 vs 1 vs 1. Pengujian ini akan dilakukan dalam 5 ronde.

Berikut adalah hasil pengujian:

Tabel 4.3.1 Pengujian Algoritma Terpilih

Percobaan	Poin			
	Pecundunk	Psilat	Psatir	Prundung
1	276	424	833	604
2	399	470	551	689
3	264	219	1016	604
4	185	505	977	746
5	217	471	1318	375
Rata-rata	268,2	417,8	939	603,6

Pada pengujian ini algoritma melawan musuh jarak dekat berada di bot Psilat, algoritma survival berada di bot Pecundunk, algoritma *ramming* musuh dengan energi rendah berada di bot Psatir, dan algoritma *corner and lock* musuh berada di bot Prundung. Berdasarkan data pengujian, terlihat bahwa algoritma yang dipilih dalam bot Psatir bekerja cukup optimal dalam mengumpulkan poin. Bot ini menjadi pemenang dalam 4 dari 5 pertandingan. Bot ini juga menjadi bot dengan rata-rata poin tertinggi dengan 939 poin, berbeda lebih dari 50 persen dengan bot peringkat kedua yang hanya memiliki rata-rata 603,6. Oleh karena itu, algoritma yang dipilih dapat dikatakan lebih baik daripada algoritma lain yang sudah dibuat, sehingga pemilihan algoritma sudah tepat.

5 KESIMPULAN

5.1 Kesimpulan

Kelas algoritma greedy yang memiliki heuristik pemecahan masalah dengan memilih pilihan yang paling optimal pada tiap instan waktu cocok untuk diterapkan sebagai sebuah basis sederhana untuk bot permainan, dalam kasus ini Robocode Tank Royale. Dalam permainan Robocode Tank Royale, algoritma menge-ram musuh yang berenergi rendah dilengkapi dengan teknik penghindaran pasif yaitu berputar-putar terbukti cukup efektif setelah dibandingkan dengan bot-bot beralgoritma lain yang telah penulis buat. Solusi tersebut ditemukan cukup efektif untuk kelas algoritma greedy dan mungkin masih bisa dikembangkan lagi lebih lanjut.

5.2 Saran

Penulis ingin menyampaikan apresiasi yang sebesar-besarnya kepada para asisten mata kuliah atas bimbingan dan dukungannya selama proses pengerjaan tugas ini. Masukan serta arahan yang diberikan sangat membantu dalam menyelesaikan tugas ini dengan baik. Namun, penulis juga menyarankan agar penjelasan mengenai cara kerja program Robocode Tank Royale lebih diperjelas dan diperdetil dan supaya tautan menuju API-nya diletakkan di lokasi yang lebih mencolok. Selain itu, penulis juga menyarankan batasan dan aturan algoritma dan struktur data untuk lebih diperjelas lagi dan untuk dipastikan bahwa tidak ada bug ketika rilis tugas besar.

5.3 Komentar

Tugas besar ini memberikan banyak pengalaman pelajaran bagi kami. Dengan adanya tugas besar ini, kami jadi mengetahui bagaimana memprogram dengan bahasa C# dan *framework* pemrograman .NET, bagaimana membangun sebuah algoritma untuk suatu bot permainan, dan juga menggunakan API yang telah disediakan oleh pemrogram lain. Kami juga menjadi lebih paham dan tahu cara menerapkan konsep algoritma *greedy* yang telah kami pelajari di kelas.

5.4 Refleksi

Penulis menyadari bahwa riset untuk pengerjaan tugas besar dilakukan secepatnya dan dari jauh-jauh hari supaya cara kerja dari program Robocode Tank Royale dapat lebih dipahami dan supaya detil-detil kecil yang ada pada API bisa tidak terlewatkan. Penulis juga belajar untuk memfokuskan pikiran ke kode yang bisa ditulis saat ini dan melepaskan hasil kompetisi nanti.

6 LAMPIRAN

6.1 Referensi

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

<https://robowiki.net/>

<https://learn.microsoft.com/en-us/dotnet/>

6.2 Tautan Repository Github

https://github.com/mraifalkautsar/Tubes1_SateAmbalPakTino

6.3 Tautan Video

<https://www.youtube.com/watch?v=eRdzR2ro9I0>

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	