

Tugas Besar 1

IF2211 - Strategi Algoritma

Algoritma Greedy

Oleh Kelompok Eco Racing:
Rifqi Naufal Abdjul (13520062)
Amar Fadil (13520103)
Vito Ghifari (13520153)



Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung
2022

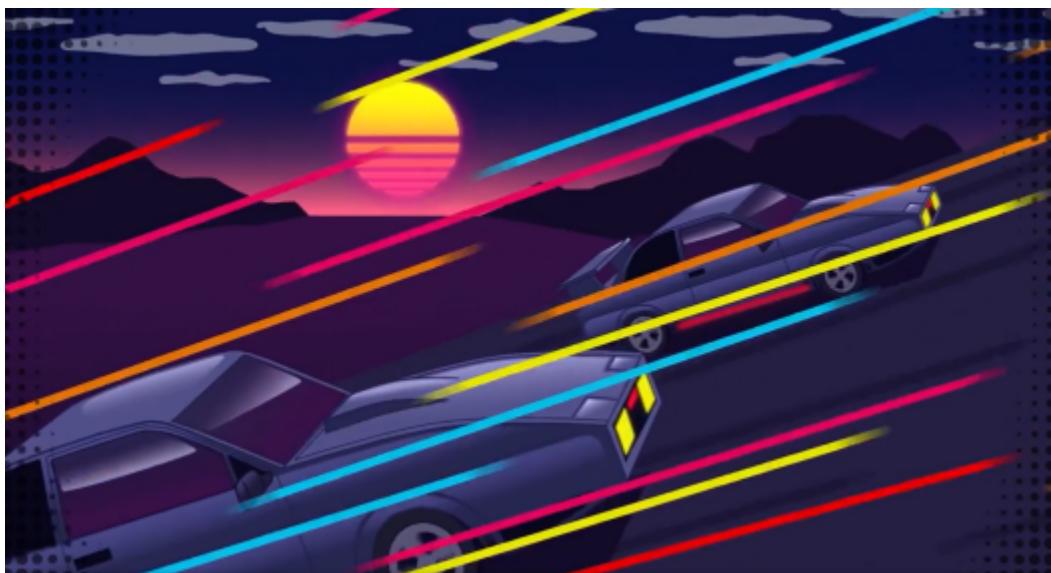
DAFTAR ISI

BAB 1	
Deskripsi Masalah	3
BAB 2	
Landasan Teori	5
Algoritma Greedy	5
Game Engine	5
Bot pada Overdrive	6
BAB 3	
Aplikasi Strategi Greedy	7
Mapping Elemen Greedy	7
Pengelompokan Elemen Greedy	9
Strategi Greedy dan Alternatifnya	9
Jenis Command Fix	9
Jenis Command Dodge	10
Jenis Command Accel	10
Jenis Command Offensive	11
Pemilihan Solusi Greedy	12
BAB 4	
Implementasi dan Pengujian	13
Implementasi Algoritma	13
Struktur Data	16
Analisis Hasil Pengujian	21
BAB 5	
Kesimpulan dan Saran	30
Kesimpulan	30
Saran	30
Lampiran	31
Daftar Pustaka	32

BAB 1

Deskripsi Masalah

Overdrive adalah sebuah game yang mempertandingan 2 bot mobil dalam sebuah ajang balapan. Setiap pemain akan memiliki sebuah bot mobil dan masing-masing bot akan saling bertanding untuk mencapai garis finish dan memenangkan pertandingan. Agar dapat memenangkan pertandingan, setiap pemain harus mengimplementasikan strategi tertentu untuk dapat mengalahkan lawannya.



Gambar 1. Ilustrasi permainan Overdrive

Pada tugas besar pertama Strategi Algoritma ini, gunakanlah sebuah game engine yang mengimplementasikan permainan Overdrive. Game engine dapat diperoleh pada laman berikut:

<https://github.com/EntelectChallenge/2020-Overdrive>.

Tugas mahasiswa adalah mengimplementasikan bot mobil dalam permainan Overdrive dengan menggunakan **strategi greedy** untuk memenangkan permainan. Untuk mengimplementasikan bot tersebut, mahasiswa disarankan melanjutkan program yang terdapat pada *starter-bots* di dalam starter-pack pada laman berikut ini:

<https://github.com/EntelectChallenge/2020-Overdrive/releases/tag/2020.3.4>

Spesifikasi permainan yang digunakan pada tugas besar ini disesuaikan dengan spesifikasi yang disediakan oleh *game engine* Overdrive pada tautan di atas. Beberapa aturan umum adalah sebagai berikut.

1. Peta permainan memiliki bentuk array 2 dimensi yang memiliki 4 jalur lurus. Setiap jalur dibentuk oleh *block* yang saling berurutan, panjang peta terdiri atas 1500 *block*.

Terdapat 5 tipe *block*, yaitu *Empty*, *Mud*, *Oil Spill*, *Flimsy Wall*, dan *Finish Line* yang masing-masing karakteristik dan efek berbeda. *Block* dapat memuat *powerups* yang bisa diambil oleh mobil yang melewati *block* tersebut.

2. Beberapa *powerups* yang tersedia adalah:
 - a. *Oil item*, dapat menumpahkan oli di bawah mobil anda berada.
 - b. *Boost*, dapat mempercepat kecepatan mobil anda secara drastis.
 - c. *Lizard*, berguna untuk menghindari *lizard* yang mengganggu jalan mobil anda.
 - d. *Tweet*, dapat menjatuhkan truk di *block* spesifik yang anda inginkan.
 - e. *EMP*, dapat menembakkan *EMP* ke depan jalur dari mobil anda dan membuat mobil musuh (jika sedang dalam 1 *lane* yang sama) akan terus berada di *lane* yang sama sampai akhir pertandingan. Kecepatan mobil musuh juga dikurangi 3.
3. Bot mobil akan memiliki kecepatan awal sebesar 5 dan akan maju sebanyak 5 *block* untuk setiap *round*. *Game state* akan memberikan jarak pandang hingga 20 *block* di depan dan 5 *block* di belakang bot sehingga setiap bot dapat mengetahui kondisi peta permainan pada jarak pandang tersebut.
4. Terdapat *command* yang memungkinkan bot mobil untuk mengubah jalur, mempercepat, memperlambat, serta menggunakan *powerups*. Pada setiap *round*, masing-masing pemain dapat memberikan satu buah *command* untuk mobil mereka. Berikut jenis-jenis *command* yang ada pada permainan:
 - a. NOTHING
 - b. ACCELERATE
 - c. DECELERATE
 - d. TURN_LEFT
 - e. TURN_RIGHT
 - f. USE_BOOST
 - g. USE_OIL
 - h. USE_LIZARD
 - i. USE_TWEET
 - j. USE_EMP
 - k. FIX
5. *Command* dari kedua pemain akan dieksekusi secara bersamaan (bukan sekuensial) dan akan divalidasi terlebih dahulu. Jika *command* tidak valid, bot mobil tidak akan melakukan apa-apa dan akan mendapatkan pengurangan skor.
6. Bot pemain yang pertama kali mencapai garis *finish* akan memenangkan pertandingan. Jika kedua bot mencapai garis *finish* secara bersamaan, bot yang akan memenangkan pertandingan adalah yang memiliki kecepatan tercepat, dan jika kecepatannya sama, bot yang memenangkan pertandingan adalah yang memiliki skor terbesar.

Adapun peraturan yang lebih lengkap dari permainan *Overdrive*, dapat dilihat pada laman :

<https://github.com/EntelectChallenge/2020-Overdrive/blob/develop/game-engine/game-rules.md>

BAB 2

Landasan Teori

2.1. Algoritma Greedy

Algoritma *greedy* merupakan salah satu cara untuk memecahkan masalah pada suatu persoalan. Algoritma ini cukup populer dan mengandalkan pendekatan yang sederhana untuk persoalan optimasi, yaitu masalah mengenai pencarian maksimum dan minimum.

Prinsip kerja dari algoritma ini adalah “*take what you can get now*”, yaitu mencari solusi optimal pada setiap langkah. Jika solusi yang ingin dicari adalah nilai maksimal, algoritma *greedy* akan mengambil nilai maksimal saat itu tanpa memikirkan konsekuensi pada langkah berikutnya. Sebaliknya, jika solusi yang diinginkan adalah solusi dengan nilai minimal, nilai minimal akan diambil. Nilai maksimal atau minimal sementara ini disebut *local optimum*. Pada langkah berikutnya, algoritma *greedy* akan kembali mencari *local optimum*. Pencarian *local optimum* ini terus diulang hingga langkah terakhir. Pada langkah terakhir, algoritma *greedy* diharapkan dapat menemukan solusi optimal dari permasalahan dengan solusinya disebut sebagai *global optimum*. Hanya saja, sebagian permasalahan yang ada tidak dijamin ditemukan solusi optimalnya dengan algoritma *greedy*.

2.2. Game Engine

Permainan Overdrive dibuat menggunakan bahasa pemrograman Scala. Scala adalah bahasa pemrograman yang diketik secara statis (*statically typed*) dan dapat digunakan dengan paradigma fungsional ataupun *object oriented*. Program yang diketik dengan bahasa Scala dapat di-*compile* menjadi *byte code* layaknya Java, serta dijalankan dengan JVM (Java Virtual Machine). Untuk melakukan kompilasi pada Overdrive, diperlukan SBT, yaitu *interactive build tool* untuk Scala. Namun, *repository* Overdrive mempunyai versi yang sudah dirilis beserta bot permulaan. Artinya, kita tidak perlu melakukan kompilasi sendiri.

Setelah melakukan pengunduhan pada versi rilis Overdrive, terdapat folder bernama starter-pack. Di dalam folder tersebut, terdapat beberapa file dan folder. Di antara file-file tersebut adalah file bernama game-engine.jar dan run.bat. Untuk menjalankan Overdrive pada sistem operasi Windows, run.bat dijalankan. Sementara itu, pada sistem Linux, dibutuhkan terminal yang mengarah ke folder starter-pack. Kemudian, pada terminal digunakan perintah ‘make’ untuk menjalankan permainan Overdrive.

2.3. Bot pada Overdrive

Pada folder *starter-pack*, terdapat sebuah folder bernama *starter-bot*. Folder tersebut berisi folder yang terdiri atas beberapa folder kode sumber bot dalam berbagai bahasa pemrograman, salah satunya adalah Java yang akan dipakai. Saat kode sumber bot java ini dibuka, terdapat algoritma bot dasar seperti perintah untuk mempercepat mobil atau *accelerate* dan memperbaiki mobil atau *fix*. Kode sumber ini dapat diubah oleh pengguna agar langkah yang diambil oleh bot mengikuti kehendak pengguna. Kode sumber bot yang menggunakan bahasa Java harus *di-compile* terlebih dahulu agar dapat digunakan untuk bertanding pada permainan Overdrive, yaitu dengan menggunakan Maven. Salah satu perangkat lunak untuk pengembangan dengan bahasa Java, yaitu JetBrains IntelliJ, dapat mendeteksi adanya penggunaan Maven pada suatu proyek yang menggunakan Java. Menggunakan perangkat lunak IntelliJ, kode sumber bot dapat *di-compile* dengan melakukan perintah *build*. Hasil dari *build* menggunakan Maven adalah file dua file dengan format jar. File ini dapat diatur agar digunakan oleh permainan Overdrive, yaitu dengan mengubah file bernama *game-runner-config.json*.

BAB 3

Aplikasi Strategi Greedy

3.1. Mapping Elemen Greedy

Terdapat beberapa identifikasi elemen-elemen dari suatu masalah yang dapat dikerjakan secara sistematis dengan menggunakan strategi greedy:

a. Himpunan Kandidat (C)

Himpunan kandidat pada permasalahan Overdrive ini adalah beberapa *command* yang dapat dipilih secara tidak terbatas oleh bot pada setiap rondenya. Secara formal, himpunan kandidat permasalahan ini adalah
 $C = \{\text{NOTHING}, \text{ACCELERATE}, \text{DECELERATE}, \text{TURN_LEFT}, \text{TURN_RIGHT}, \text{USE_BOOST}, \text{USE_OIL}, \text{USE_LIZARD}, \text{USE_TWEET} <\text{LANE}> <\text{BLOCK}>, \text{USE_EMP}, \text{FIX}\}$.

Terdapat 11 anggota dari himpunan kandidat ini, yakni:

1. NOTHING

Bot tidak akan melakukan apa-apa pada ronde ini.

2. ACCELERATE

Bot akan menambah state *speed*. *Speed* tidak akan berubah jika mobil telah mencapai maksimal *speed* yang diperbolehkan.

3. DECELERATE

Bot akan mengurangi state *speed*. *Speed* tidak akan berubah jika mobil telah berhenti.

4. TURN_LEFT

Bot akan belok ke kiri (*lane* ronde ini = *lane* ronde sebelumnya - 1).

5. TURN_RIGHT

Bot akan belok ke kanan (*lane* ronde ini = *lane* ronde sebelumnya + 1).

6. USE_BOOST

Bot akan menggunakan *powerup boost* sehingga *speed* mobil akan menjadi maksimum *speed* yang diperbolehkan.

7. USE_OIL

Bot akan menumpahkan oli pada *block* dan *lane* yang sama dengan posisi mobil di ronde ini.

8. USE_LIZARD

Bot akan menggunakan *lizard* sehingga mobil akan lompat pada ronde ini. Semua *block* yakni *powerups* dan *obstacle* akan diabaikan sehingga *bot* tidak akan mendapatkan *powerups* atau terkena *damage* dari *obstacle* yang ada, kecuali *block* pada posisi terakhir bot setelah lompat (posisi *block bot* + *speed bot*).

9. USE_TWEET <LANE> <BLOCK>

Bot akan meletakkan *cybertruck* pada posisi *BLOCK* dan *LANE* saat ronde selanjutnya.

10. USE_EMP

Bot akan menggunakan EMP sehingga mobil pemain lain yang berada di depan berhenti pada ronde ini.

11. FIX

Bot akan memberhentikan mobil dan memperbaiki mobilnya sehingga *damage* yang dimiliki mobil akan berkurang sebanyak 2 pada ronde ini.

b. Himpunan Solusi (S)

Himpunan solusi pada permasalahan *Overdrive* ini adalah himpunan yang berisikan anggota himpunan kandidat terurut berdasarkan ronde. Misalnya sebuah himpunan solusi $S = \{\text{ACCELERATE}, \text{TURN_RIGHT}, \text{ACCELERATE}, \text{TURN_LEFT}\}$ akan mengakibatkan *bot accelerate* pada ronde pertama, belok ke kanan pada ronde kedua, *accelerate* pada ronde ketiga, dan belok ke kiri pada ronde keempat.

c. Fungsi Solusi

Fungsi solusi pada permasalahan *Overdrive* ini adalah akhir dari permainan ketika salah satu pemain telah mencapai *block FINISH* atau ronde telah mencapai maksimum. Karena sifatnya yang tidak deterministik maka *bot* akan terus menambah anggota himpunan kandidat ke himpunan solusi hingga *game engine* memutuskan akhir dari permainan yang terlihat pada kode *bot* menggunakan *infinite loop (while true)*.

d. Fungsi Seleksi

Fungsi seleksi pada permasalahan *Overdrive* ini adalah strategi greedy yang digunakan setiap rondenya dalam memilih himpunan kandidat terbaik saat ronde tersebut. Hasil dari fungsi seleksi adalah *command bot* terpilih dari himpunan kandidat yang kemudian menjadi anggota himpunan solusi. Strategi greedy yang digunakan dan berbagai alternatifnya akan dijelaskan lebih lanjut pada sub-bab selanjutnya.

e. Fungsi Kelayakan

Fungsi kelayakan pada permasalahan *Overdrive* ini adalah beberapa *command* yang tidak valid untuk ronde tertentu sehingga mendapatkan penalti berupa pengurangan poin dan pengubahan command menjadi NOTHING. Beberapa *command* yang tidak layak dan kondisinya adalah sebagai berikut:

- TURN_LEFT, jika posisi *lane bot* pada ronde ini adalah 1 (*bot* sudah berada pada posisi paling kiri)
- TURN_RIGHT, jika posisi *lane bot* pada ronde ini adalah 4 (*bot* sudah berada pada posisi paling kanan)
- USE_BOOST, USE_OIL, USE_LIZARD, USE_TWEET, dan USE_EMP, jika *bot* tidak memiliki *powerup* yang bersesuaian, yakni BOOST, OIL, LIZARD, TWEET, dan EMP.

f. Fungsi Objektif

Fungsi objektif pada permasalahan *Overdrive* ini adalah memaksimumkan jarak antara *bot* dengan *bot* lainnya (*myCar.position.block - opponent.position.block*), kemudian memaksimumkan kecepatan mobil, dan terakhir memaksimumkan skor yang didapat dengan menggunakan *powerup*.

3.2. Pengelompokan Elemen Greedy

Berdasarkan eksplorasi kelompok kami, kami menentukan bahwa pengelompokkan *command* ke dalam beberapa kelompok yang akan dipilih sesuai prioritas merupakan cara yang paling benar untuk menghadapi permasalahan ini.

Kami membagi *command* menjadi 4 jenis yaitu,

- Fix

Hanya berisikan command FIX. Jenis *command* ini merupakan prioritas pertama dikarenakan *damage* mobil membatasi kecepatan maksimal mobil tersebut dan melakukan command FIX sebelum command lain tidak mengakibatkan kerugian apapun.

- Dodge

Berisikan *command* TURN_LEFT, TURN_RIGHT, LIZARD, dan DECELERATE. Jenis *command* ini merupakan prioritas setelah FIX dikarenakan jika tidak dilakukan akan mengakibatkan *damage* pada mobil dan mengakibatkan kerugian yang sangat besar.

- Accel

Berisikan *command* ACCELERATE, dan BOOST. Jenis *command* ini dilakukan untuk mengarahkan situasi mobil ke situasi yang ideal, yaitu mempunyai kecepatan maksimal.

- Offensive

Berisikan *command* EMP, OIL, dan TWEET. Jenis *command* ini dilakukan saat situasi mobil telah ideal, dan tidak ada hal yang lebih baik untuk dilakukan selain menggunakan *powerup* yang telah disimpan.

Dari semua *command* yang disediakan dalam permainan, kami memutuskan untuk tidak menggunakan *command* NOTHING karena ACCELERATE tidak memiliki pinalti apapun jika dipakai bahkan saat sesudah *max speed* tercapai, sehingga *default command* yang digunakan oleh bot adalah ACCELERATE..

3.3. Strategi Greedy dan Alternatifnya

Berdasarkan eksplorasi kelompok kami, terdapat beberapa strategi greedy untuk memilih *command* yang sesuai untuk menghasilkan hasil yang optimal. Strategi tersebut dijelaskan sebagai berikut:

3.3.1. Jenis Command Fix

Player akan mengeluarkan *command* FIX jika terdapat *damage* pada mobil dan kecepatan musuh lebih cepat daripada kecepatan maksimum player. Atau, player akan melakukan fix apapun yang terjadi jika *damage* mobil lebih dari 1.

3.3.2. Jenis Command Dodge

3.3.2.1. Lane Flagging

Algoritma “Lane Flagging” yang kami implementasikan bertujuan untuk mendapatkan tile yang mempunyai prioritas paling tinggi. Algoritma “Lane Flagging” adalah seperti berikut:

1. Melakukan iterasi pada seluruh *tiles* yang akan dilewati pada *lane*.
2. Menyimpan *tiles* yang merupakan prioritas paling tinggi mengikuti prioritas WALL/TRUCK > MUD/OIL_SPILLS > POWERUPS > EMPTY.
3. Player akan mengeluarkan *command* mengarah ke *tiles* yang lebih menguntungkan mengikuti prioritas POWERUPS > EMPTY > MUD/OIL_SPILLS > WALL.

Kami melakukan sedikit modifikasi dalam pemilihan jalur untuk menyesuaikan situasi seperti,

- Player akan mengeluarkan *command* LIZARD jika player mempunyai *powerup* LIZARD dan seluruh *lane* terhalangi.
- Jika terdapat penghalang di ujung pergerakan mobil, maka akan ditandai sebagai OIL_SPILLS agar tidak melakukan LIZARD saat menghadapinya.
- Player akan mengeluarkan *command* DECELERATE jika seluruh *lane* terhalang dan “Lane Flagging” sejauh kecepatan hasil perlambatan tidak terhalang.
- Player akan cenderung menuju ke *lane* tengah (*lane* 2 atau 3) ketika jalur kosong dan tidak ada yang dilakukan.

3.3.2.2. Alternatif Lane Flagging

Algoritma yang dapat diimplementasi untuk menggantikan algoritma “Lane Flagging” adalah algoritma berikut:

1. Menilai *lane* yang dapat dituju berdasarkan seluruh *tiles* yang akan dilewati pada *lane* tersebut dengan nilai berikut ini:
 - WALL / TRUCK = -5
 - MUD / OIL_SPILLS = -3
 - EMPTY = 0
 - POWERUPS = +1
2. Setelah menilai seluruh *lane* yang dapat dituju, algoritma akan memilih *lane* yang memiliki nilai yang lebih tinggi. Jika *lane* di kiri player memiliki nilai lebih tinggi, maka akan menjalankan *command* TURN_LEFT, dan sebaliknya.

3.3.3. Jenis Command Accel

3.3.3.1. Algoritma penggunaan BOOST

Player akan selalu mencoba mengeluarkan *command* BOOST selain beberapa kondisi, yaitu:

- Mobil sedang boosting
- Mobil memiliki damage lebih dari 0
- Terdapat *tiles* yang menghalangi (WALL, TRUCK, MUD, OIL_SPILLS)

3.3.3.2. Algoritma prioritas ACCELERATION

Player akan mengeluarkan *command* ACCELERATION jika memenuhi persyaratan dibawah ini:

- Player belum pada kecepatan maksimumnya
- Tidak ada command yang diprioritaskan dijalankan

3.3.4. Jenis Command Offensive

3.3.4.1. Algoritma Penggunaan EMP

Player akan memprioritaskan menggunakan EMP jika memenuhi seluruh syarat dibawah ini:

- Posisi musuh di depan player
- Kecepatan musuh lebih dari 6 (sedang menuju situasi ideal)
- Player mempunyai *powerup* EMP

3.3.4.2. Algoritma Penggunaan TWEET 1

Algoritma ini digunakan ketika posisi player didepan musuh. Algoritma ini pada setiap *round* akan mencari lokasi strategis dalam meletakkan *cybertruck*, yakni beberapa *terrain* pada *block* yang sama dalam lane berbeda (satu garis lurus vertikal) dengan *terrain* yang menjadi *obstacle* (*terrain* berisi *oil spill*, *wall*, *mud*, atau *cybertruck* musuh) berjumlah 2 atau 3 memiliki 1 lokasi strategis yakni *terrain* terakhir yang bukan *obstacle*. Ketika pemain mempunyai *powerup* TWEET, algoritma akan melakukan hal berikut:

1. Menghapus semua posisi strategis yang kurang dari posisi musuh + 10.
2. Jika ada posisi yang tersisa, berada di belakang pemain, dan *cybertruck* yang diletakkan sebelumnya (jika ada) sudah dilewati oleh musuh, letakkan *cybertruck* pada posisi strategis pertama (terdekat) dari musuh.

3.3.4.3. Algoritma Penggunaan TWEET 2

Algoritma ini digunakan ketika posisi player dibelakang musuh. Algoritma pemanggilan *command* ini adalah seperti berikut:

1. Mendapatkan posisi musuh dan tambahkan dengan kecepatan musuh pada saat itu.
2. Letakkan *cybertruck* pada posisi tersebut dengan digeser beberapa *block* (sekitar 1-3 *block*) untuk menghalangi pergerakan musuh selanjutnya.

3.3.4.4. Algoritma Penggunaan OIL

Player akan menggunakan OIL jika pemain berada di depan musuh dan player mempunyai *powerup* oil

3.4. Pemilihan Solusi Greedy

Setelah melakukan testing dan perbandingan dengan alternatif algoritma yang kami eksplorasi, kami mengambil keputusan terhadap beberapa hal, yaitu:

- Menggunakan Lane Flagging dibanding alternatifnya
Kami memutuskan untuk tetap menggunakan Lane Flagging dikarenakan beberapa alasan,
 1. Karena kompleksitas waktu lebih singkat
 2. Lebih mudah di implementasi dikarenakan mengabaikan beberapa detail
 3. Tidak terlalu butuh tweaking (dikarenakan perbandingan diskrit tidak tergantung jumlah terrain)
- Menggunakan 2 jenis algoritma untuk penggunaan power up TWEET
Awalnya, kami hanya menggunakan tweet ketika player mendahului opponent, dikarenakan keterbatasan dalam penglihatan map. Tetapi kami merasa powerup tweet akan menumpuk ketika player didahului opponent. Sehingga kami menambahkan algoritma baru agar power up TWEET tidak menumpuk.

BAB 4

Implementasi dan Pengujian

4.1. Implementasi Algoritma

Bot yang kami buat diimplementasikan dalam bahasa pemrograman Java. Pada file utama, yaitu Bot.java, terdapat kelas utama bernama Bot untuk mengeluarkan perintah-perintah yang ada pada bot. Perintah ini dikembalikan ke file Main.java untuk dieksekusi langsung pada permainan Overdrive. Pada setiap ronde, Kelas bot pada Bot.java akan menerima state dari permainan. Lalu, state tersebut akan digunakan oleh LaneFlagger. ke empat command: fix, dodge, accel, dan offensive. Implementasi algoritma dari LaneFlagger serta empat command ini dijelaskan sebagai berikut.

4.1.1. LaneFlagger

LaneFlagger bertujuan untuk melakukan *flag* pada 3 *lane*, yaitu lajur mobil, dan lajur kanan serta kiri dari mobil tersebut. Jika mobil berada di *lane* 1 atau *lane* 4, *lane* yang tidak valid akan ditandai sebagai dinding. Prioritas untuk menandai suatu lajur adalah sebagai berikut:

WALL/CYBERTRUCK > MUD/OIL SPILL > EMPTY > BOOST.

Hasil dari LaneFlagger akan digunakan pada *command* dodge dan accel.

```
// Iterate through possible lanes on the side of car
for (int i = Math.max(0, car.position.lane - 2); i <= Math.min(3, car.position.lane); i++) {
    lane = map.get(i);

    // If already flagged out of bounds, skip it
    if (flags[i - car.position.lane + 2] == Terrain.WALL) {
        continue;
    }

    // Get a modifier for turning and accelerating
    modifier = -1;
    // If checking the lane in front of the car
    if (i - car.position.lane + 2 == 1) {
        // If car is not going to accel, set modifier as 0
        if (distance == car.getSpeed()) {
            modifier = Math.max(car.getNextSpeed() - car.getSpeed(), 0);
        } else {
            modifier = 0;
        }
    }
}
```

Gambar 4.1 Implementasi LaneFlagger untuk melakukan iterasi pada setiap lajur

4.1.2. Fix

Algoritma *fix* dimulai dengan mengecek kecepatan lawan dan kecepatan mobil sendiri, serta *damage* dari mobil sendiri. Jika kecepatan lawan lebih besar daripada mobil sendiri serta *damage* lebih besar dari nol, mobil akan diperbaiki. Perintah *fix* merupakan perintah dengan prioritas tertinggi sehingga akan selalu dieksekusi jika kedua syarat ini dipenuhi.

```
// Add command to the list of commands
public void update(GameState state) {
    Car player = state.player;
    Car opponent = state.opponent;
    commands.clear();
    if (opponent.speed > player.getMaxSpeed() && player.damage > 0) {
        commands.add(FIX);
    } else {
        if (player.damage > 1) {
            commands.add(FIX);
        }
    }
}
```

Gambar 4.2 Implementasi fix untuk melakukan perbaikan pada mobil

4.1.3. Dodge

Algoritma *dodge* pada bot yang kami buat adalah dengan melakukan pertimbangan dari hasil LaneFlagger. Potongan kode di gambar 4.3 adalah kasus saat lajur mobil mempunyai halangan seperti *mud* dan sejenisnya. Mobil akan berbelok ke kanan atau ke kiri bergantung pada kosongnya lajur di sebelahnya. Selain itu, jika terdapat powerup lizard yang tersimpan, mobil dapat memprioritaskan untuk melompati halangan tersebut.

```
// If there's an obstacle, avoid it
if (!flags[1].equals(Terrain.EMPTY)) {
    // Prioritize to stay on middle (lane 2/3)
    if (carLane < 3 && flags[2].equals(Terrain.EMPTY)) {
        commands.add(TURN_RIGHT);
    } else if (carLane > 1 && flags[0].equals(Terrain.EMPTY)) {
        commands.add(TURN_LEFT);
        // If there's an obstacle on the middle, move to the edge
    } else if (flags[2].equals(Terrain.EMPTY)) {
        commands.add(TURN_RIGHT);
    } else if (flags[0].equals(Terrain.EMPTY)) {
        commands.add(TURN_LEFT);
    }

    // If has LIZARD, use it
    if (Extras.hasPowerUp(PowerUps.LIZARD, car) && !flags[1].equals(Terrain.OIL_SPILL)) {
        commands.add(LIZARD);
    }
}
```

Gambar 4.3 Implementasi *dodge* pada kasus lajur mobil tidak kosong

4.1.4. Accel

Perintah *accel* menempati sebagai prioritas ketiga dari keempat *command*. Algoritma *accel* akan memprioritaskan untuk melakukan *boost* apabila terdapat *boost* serta *damage* mobil sama dengan nol. Selain itu, konsekuensi melakukan *boost* akan diperhatikan karena banyaknya *block* yang harus diperhatikan akan bertambah menjadi sejauh kecepatan *boost*. Jika tidak terdapat *boost* atau syarat sebelumnya tidak terpenuhi, *accel* hanya akan mengeluarkan *command* *accelerate*.

```
public void update(GameState state) {
    Terrain[] flags = flagger.getBoostFlags();
    Car car = state.player;
    commands.clear();
    boolean useBoost = true;

    // check if the car has damage or the car is already boosting
    if (car.damage > 0 || car.boosting) {
        useBoost = false;
    } else {
        if (flags[1].equals(Terrain.MUD) || flags[1].equals(Terrain.WALL) || flags[1].equals(Terrain.OIL_SPILL))
            useBoost = false;
    }

    if (useBoost && Extras.hasPowerUp(PowerUps.BOOST, car))
        commands.add(BOOST);

    if (car.getSpeed() < car.getMaxSpeed()) {
        commands.add(ACCELERATE);
    }
}
```

Gambar 4.4 Implementasi accel untuk melakukan akselerasi mobil ataupun melakukan boost

4.1.5. Offensive

Algoritma *offensive* meng-handle perintah-perintah seperti TWEET, OIL, dan EMP. Perintah TWEET digunakan untuk menambah rintangan pada jalan berupa *cyber truck* yang efeknya sama dengan menambahkan *wall* pada satu blok tertentu. Ada dua pendekatan pada TWEET, yaitu pada saat lawan berada di depan mobil dan pada saat lawan berada di belakang mobil. Ketika lawan berada di depan mobil sendiri, *cyber truck* akan ditempatkan di lajur musuh pada blok lawan + kecepatan musuh + 3. Jika lawan berada di belakang mobil sendiri, algoritma ini akan menyimpan posisi dengan 2 atau 3 halangan sehingga *cyber truck* akan ditempatkan sebagai halangan ketiga ataupun keempat. Algoritma ini terdapat pada gambar 4.5. Perintah yang lain untuk menyerang, yaitu OIL, akan digunakan apabila kondisi TWEET dan EMP tidak terpenuhi, serta mobil lawan berada di belakang mobil sendiri. Sementara itu, syarat untuk menggunakan EMP adalah apabila mobil lawan berada di depan.

```
private void generateCybertruckPosition(GameState state) {
    List<Lane[]> map = state.lanes;
    // Scan for any candidate position for cybertruck
    // Get starting block position
    final int startBlock = map.get(0)[0].position.block;
    // Get length of the lane
    final int laneLen = map.get(0).length;
    // Obstacle counter and free lane index getter.
    int countObstacle, freeLane;
    // Searching schema using boolean.
    boolean isEnd = false;
    int j;
    // Starts from block that is not checked until to the end of the block in the map (laneLen).
    for (j = Math.max(lastCheckBlock - startBlock, 0) + 1; j < laneLen; j++) {
        // Reset the candidate lane, obstacle counter, and last checked block
        freeLane = -1;
        countObstacle = 0;
        lastCheckBlock = startBlock + j;
        // Check each lane in the same x position (i)
        for (int i = map.size() - 1; i ≥ 0; i--) {
            // Get every block in lane i
            Lane[] lane = map.get(i);
            // Check if (i, j) terrain is the end. If the end, it will stop the check.
            if (Extras.isEndOfLane(lane[j])) {
                isEnd = true;
                break;
            }
            // Check if (i, j) terrain is obstacle. If obstacle, increment the obstacle counter.
            if (Extras.isObstacle(lane[j])) {
                countObstacle++;
            } else {
                // If not obstacle, set the free lane index to i + 1.
                freeLane = i + 1;
            }
        }
        // If loop breaks because of the end, it will stop the check.
        if (isEnd)
            break;
        // If loop continue and there are 2 or 3 obstacles,
        else if (countObstacle == 2 || countObstacle == 3) {
            // add (freeLane, lastCheckBlock) as the strategic position in TWEET command.
            TWEET.addPosition(freeLane, lastCheckBlock);
        }
    }
    if (isEnd)
        lastCheckBlock--;
}
```

Gambar 4.5 Implementasi penandaan (*mark*) halangan untuk penggunaan TWEET

4.2. Struktur Data

Program ini dibuat dengan nama *package root project za.co.entelect.challenge*.

Terdapat beberapa struktur data berupa kelas yang dipecah menjadi beberapa modul.

4.2.1. Command (command.*)

Merupakan modul yang dapat mengeluarkan string berupa command. Terdapat total 10 command yang ada pada modul ini dengan tiap nama kelasnya memiliki akhiran Command, yakni **AccelerateCommand**, **BoostCommand**, **ChangeLaneCommand**, **DecelerateCommand**, **EmpCommand**, **FixCommand**, **LizardCommand**, **OilCommand**, dan **TweetCommand**. Modul

ini memiliki interface berupa metode render yang mengembalikan nilai String sesuai dengan command. Beberapa command membutuhkan logika tambahan seperti TweetCommand yang menyimpan posisi strategis untuk meletakkan cybertruck, atau ChangeLaneCommand yang dapat diinisialisasi menjadi belok kiri atau belok kanan.

```
1 package za.co.entelect.challenge.command;
2
3 public interface Command {
4     String render();
5 }
```

Gambar 4.2.1. Command pada Java

4.2.2. Command Group (command.groups.*)

Merupakan modul yang menjadi implementasi dari greedy dengan mengkategorikan beberapa command menjadi satu grup. Hal ini dilakukan untuk memprioritaskan beberapa command yang memiliki tujuan sama. Terdapat total 4 *command group* yang menerapkan interface CommandGroup, yakni **Accel**, **Dodge**, **Fix**, dan **Offensive**. Setiap command group memiliki interface berupa metode update yang menerima argumen *state* permainan sehingga membuat kemungkinan daftar command yang mungkin dijalankan berdasarkan state yang diberikan. Selain itu terdapat interface getCommand yang akan mengembalikan daftar command yang telah diproses saat update. Beberapa kategori *command* juga membutuhkan LaneFlags saat *constructor*.

```
1 package za.co.entelect.challenge.command.groups;
2 import za.co.entelect.challenge.command.Command;
3 import za.co.entelect.challenge.entities.GameState;
4
5 import java.util.ArrayList;
6
7 // An interface to a command group.
8 // It should have update and get commands.
9 public interface CommandGroups {
10
11     // Update the group, generate the proper commands
12     // given the current game state, then save it in
13     // commands.
14     public void update(GameState state);
15
16     // Get the commands generated by this group.
17     // Should be called after update.
18     public ArrayList<Command> getCommands();
19 }
```

Gambar 4.2.2. Implementasi CommandGroups

4.2.3. Entities (entities.*)

Merupakan model entitas yang mengonversi deserialisasi state permainan dari representasi JSON ke bentuk objek Java. Terdapat 4 entitas utama dalam program, yakni **Car**, **GameState**, **Lane**, dan **Position**.

```
public class Car {  
    @SerializedName("id")  
    public int id;  
  
    @SerializedName("position")  
    public Position position;  
  
    @SerializedName("speed")  
    public int speed;  
  
    @SerializedName("state")  
    public State state;  
  
    @SerializedName("damage")  
    public int damage;  
  
    @SerializedName("powerups")  
    public PowerUps[] powerups;  
  
    @SerializedName("boosting")  
    public Boolean boosting;  
  
    @SerializedName("boostCounter")  
    public int boostCounter;  
}  
  
public class GameState {  
    @SerializedName("currentRound")  
    public int currentRound;  
  
    @SerializedName("maxRounds")  
    public int maxRounds;  
  
    @SerializedName("player")  
    public Car player;  
  
    @SerializedName("opponent")  
    public Car opponent;  
  
    @SerializedName("worldMap")  
    public List<Lane[]> lanes;  
}  
  
public class Lane {  
    @SerializedName("position")  
    public Position position;  
  
    @SerializedName("surfaceObject")  
    public Terrain terrain;  
  
    @SerializedName("occupiedByPlayerId")  
    public int occupiedByPlayerId;  
  
    @SerializedName("isOccupiedByCyberTruck")  
    public boolean isOccupiedByCyberTruck;  
}  
  
public class Position {  
    @SerializedName("y")  
    public int lane;  
  
    @SerializedName("x")  
    public int block;  
  
    public Position() {  
        this(-1, -1);  
    }  
  
    public Position(int lane, int block) {  
        this.lane = lane;  
        this.block = block;  
    }  
}
```

Gambar 4.2.3 Implementasi entities

4.2.4. Enums (enums.*)

Merupakan modul *enumerable* berupa pilihan set nilai yang telah didefinisikan. Variabel yang memiliki tipe enum ini dapat di-assign dengan beberapa pilihan nilai yang sudah ditentukan. Terdapat 4 enum pada program ini, yakni **Direction**, **PowerUps**, **State**, dan **Terrain**.

```
public enum Direction {  
    FORWARD(0, 1, "FORWARD"),  
    BACKWARD(0, -1, "BACKWARD"),  
    LEFT(-1, 0, "LEFT"),  
    RIGHT(1, 0, "RIGHT");  
  
    public final int lane;  
    public final int block;  
    public final String label;  
  
    Direction(int lane, int block, String label) {  
        this.lane = lane;  
        this.block = block;  
        this.label = label;  
    }  
  
    public String getLabel() {  
        return this.label;  
    }  
}  
  
public enum PowerUps {  
    @SerializedName("BOOST")  
    BOOST,  
    @SerializedName("OIL")  
    OIL,  
    @SerializedName("TWEET")  
    TWEET,  
    @SerializedName("LIZARD")  
    LIZARD,  
    @SerializedName("EMP")  
    EMP;  
}  
  
public enum State {  
    @SerializedName("ACCELERATING")  
    ACCELERATING,  
    @SerializedName("READY")  
    READY,  
    @SerializedName("NOTHING")  
    NOTHING,  
    @SerializedName("TURNING_RIGHT")  
    TURNING_RIGHT,  
    @SerializedName("TURNING_LEFT")  
    TURNING_LEFT,  
    @SerializedName("HIT_MUD")  
    HIT_MUD,  
    @SerializedName("HIT_OIL")  
    HIT_OIL,  
    @SerializedName("HIT_EMP")  
    HIT_EMP,  
    @SerializedName("DECELERATING")  
    DECELERATING,  
    @SerializedName("PICKED_UP_POWERUP")  
    PICKED_UP_POWERUP,  
    @SerializedName("USED_BOOST")  
    USED_BOOST,  
    @SerializedName("USED_OIL")  
    USED_OIL,  
    @SerializedName("FIXED_CAR")  
    FIXED_CAR,  
    @SerializedName("FINISHED")  
    FINISHED  
}  
  
public enum Terrain {  
    @SerializedName("0")  
    EMPTY,  
    @SerializedName("1")  
    MUD,  
    @SerializedName("2")  
    OIL_SPILL,  
    @SerializedName("3")  
    OIL_POWER,  
    @SerializedName("4")  
    FINISH,  
    @SerializedName("5")  
    BOOST,  
    @SerializedName("6")  
    WALL,  
    @SerializedName("7")  
    LIZARD,  
    @SerializedName("8")  
    TWEET,  
    @SerializedName("9")  
    EMP  
}
```

Gambar 4.2.3 Implementasi enums

4.2.5. Utils (utils.*)

Merupakan modul utilitas yang dapat digunakan secara statis di kelas manapun. Berisikan beberapa fungsi yang sering direferensikan secara umum di berbagai kelas dan tidak terhubung erat (*loose couple*) dengan kelas lainnya. Terdapat dua kelas yang termasuk sebagai modul ini, **LaneFlagger** dan **Extras**. LaneFlagger merupakan penandaan *lane* terdekat yang memiliki prioritas tipe terrain tinggi. Extras berisi fungsi untuk memeriksa powerup yang dimiliki, terrain berupa *mud* atau *wall*, terrain berupa *obstacle*, dan terrain berupa akhir garis finish.

```
public class Extras {
    public static Boolean hasPowerUp(PowerUps powerUpToCheck, Car car) {
        for (PowerUps powerUp : car.powerups) {
            if (powerUp.equals(powerUpToCheck)) {
                return true;
            }
        }
        return false;
    }

    public static boolean isMudOrWall(Terrain[] flags, int lane) {
        return flags[lane] == Terrain.MUD || flags[lane] == Terrain.WALL;
    }

    public static boolean isObstacle(Lane lane) {
        return (
            lane.terrain == Terrain.MUD
            || lane.terrain == Terrain.WALL
            || lane.terrain == Terrain.OIL_SPILL
            || lane.isOccupiedByCyberTruck
        );
    }

    public static boolean isEndOfLane(Lane lane) {
        return (lane == null || lane.terrain == Terrain.FINISH);
    }
}
```

Gambar 4.2.5 Implementasi utils

4.2.6. Bot (Bot)

Merupakan kelas *bootstrapper* yang menggabungkan seluruh *command groups* menjadi satu dan inisialisasi kebutuhan lain dalam menjalankan bot. Tersedia dua metode yaitu update untuk mengubah *state* permainan untuk bot dan meng-update *command groups* dan run yang mengembalikan *command* setelah disortir dengan prioritas.

4.2.7. Main Program (Main)

Merupakan program utama yang menghubungkan bot dengan *game runner*. Program akan loop tak hingga dengan mengambil input nomor ronde permainan berupa integer, lalu mengambil *file state* permainan berdasarkan nomor ronde yang berupa string JSON berisi *state* permainan, kemudian di-*deserialize* menjadi objek Java yaitu *GameState*, terakhir menggunakan sebagai argumen untuk update bot. Setelah itu, bot dijalankan dengan metode run dan menghasilkan *Command* yang akan di-*render* sebagai string menjadi perintah untuk *game engine*.

```
try (Scanner sc = new Scanner(System.in)) {
    while (true) {
        try {
            int roundNumber = sc.nextInt();
            String statePath = String.format("./%s/%d/%s", ROUNDS_DIRECTORY, roundNumber, STATE_FILE_NAME);
            String state = new String(Files.readAllBytes(Paths.get(statePath)));
            bot.update(gson.fromJson(state, GameState.class));
            Command command = bot.run();
            System.out.println(String.format("C;%d;%s", roundNumber, command.render()));
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

Gambar 4.2.7. Kode program utama

4.3. Analisis Hasil Pengujian

4.3.1. Pengujian

4.3.1.1. Greedy pada FIX

Dapat dilihat dari gambar di bawah, player bahkan sedang dalam kecepatan yang tinggi tetapi tetap memprioritaskan FIX agar mempersiapkan diri untuk melakukan boost kedepannya.

Round 019

[Reset](#) [Remove this match](#)

[193,1]	[194,1]	[195,1]	[196,1]	[197,1]	[198,1] GOLDBEER	[199,1]	[200,1] GOLDBEER	[201,1]	[202,1]	[203,1]	[204,1]	[205,1]	[206,1]	[207,1]	[208,1]	[209,1]	[210,1]	[211,1]	[212,1]	[213,1]	[214,1]	[215,1] GOLDBEER	[216,1]	[217,1]	[218,1]
[193,2]	[194,2]	[195,2]	[196,2]	[197,2]	[198,2]	[199,2]	[200,2]	[201,2]	[202,2]	[203,2]	[204,2]	[205,2] GOLDBEER	[206,2] GOLDBEER	[207,2]	[208,2]	[209,2]	[210,2]	[211,2]	[212,2]	[213,2]	[214,2]	[215,2] GOLDBEER	[216,2]	[217,2]	[218,2]
[193,3]	[194,3]	[195,3]	[196,3]	[197,3]	[198,3]	[199,3]	[200,3]	[201,3]	[202,3]	[203,3]	[204,3]	[205,3]	[206,3]	[207,3]	[208,3]	[209,3]	[210,3]	[211,3]	[212,3]	[213,3]	[214,3]	[215,3]	[216,3]	[217,3]	[218,3]
[193,4]	[194,4]	[195,4]	[196,4]	[197,4]	[198,4]	[199,4]	[200,4]	[201,4]	[202,4]	[203,4]	[204,4]	[205,4]	[206,4] GOLDBEER	[207,4] GOLDBEER	[208,4]	[209,4]	[210,4]	[211,4]	[212,4]	[213,4]	[214,4]	[215,4] GOLDBEER	[216,4]	[217,4]	[218,4]

Round Details

Max Rounds: 200

Current Round: 19

A - Eco Racing (selected) ▾

Position	Speed	Lane	Distance	Boosts	Boosting	Powerups
1	9	1	198	0	No	TWEET,TWEET,TWEET,EMP,EMP,LIZARD

Bot Command

Command: FIX
 Execution time: 3ms
 Exception: null

Gambar 4.3.1.1. Greedy FIX

4.3.1.2. Greedy pada Dodge

Pada gambar pertama di bawah ini, player selalu mengarah ke arah powerups walau tanpa disadari, mengarah ke *lane* yang lebih buruk karena terdapat mud di depan. Pada gambar kedua, saat player tidak bisa menghindari *obstacles*, player langsung menggunakan powerup LIZARD nya untuk mempertahankan kecepatannya.

Round 004

[Reset](#) [Remove this match](#)

[14, 1]	[15, 1]	[16, 1]	[17, 1]	[18, 1] <HOLD>	[19, 1]	[20, 1]	[21, 1]	[22, 1]	[23, 1]	[24, 1] <WEET>	[25, 1]	[26, 1]	[27, 1]	[28, 1]	[29, 1]	[30, 1]	[31, 1]	[32, 1] <HOLD>	[33, 1] <HOLD>	[35, 1]	[36, 1]	[37, 1]	[38, 1]	[39, 1]			
[14, 2]	[15, 2]	[16, 2] <PLAYER>	[17, 2]	[18, 2]	[19, 2] <PLAYER>	[20, 2]	[21, 2]	[22, 2]	[23, 2]	[24, 2]	[25, 2]	[26, 2]	[27, 2]	[28, 2]	[29, 2]	[30, 2]	[31, 2]	[32, 2]	[33, 2]	[34, 2]	[35, 2]	[36, 2]	[37, 2] <LIZARD>	[38, 2] <BOOST>	[39, 2]		
[14, 3]	[15, 3]	[16, 3]	[17, 3] <HOLD>	[18, 3]	[19, 3]	[20, 3] <WEET>	[21, 3]	[22, 3]	[23, 3]	[24, 3]	[25, 3]	[26, 3]	[27, 3]	[28, 3]	[29, 3]	[30, 3]	[31, 3]	[32, 3]	[33, 3]	[34, 3]	[35, 3]	[36, 3]	[37, 3]	[38, 3]	[39, 3]		
[14, 4]	[15, 4]	[16, 4]	[17, 4]	[18, 4]	[19, 4]	[20, 4]	[21, 4]	[22, 4]	[23, 4]	[24, 4]	[25, 4]	[26, 4]	[27, 4]	[28, 4]	[29, 4] <WEET>	[30, 4]	[31, 4]	[32, 4]	[33, 4]	[34, 4]	[35, 4] <HOLD>	[36, 4] <HOLD>	[38, 4]	[39, 4]	[37, 4]	[38, 4]	[39, 4]

[< First < Prev 4 Next > Last >]

(Click the button that displays the round number to quickly switch rounds)

[93, 1]	[94, 1]	[95, 1]	[96, 1]	[97, 1]	[98, 1]	[99, 1]	[100, 1]	[101, 1]	[102, 1]	[103, 1]	[104, 1]	[105, 1]	[106, 1]	[107, 1]	[108, 1]	[109, 1]	[110, 1]	[111, 1]	[112, 1]	[113, 1]	[114, 1]	[115, 1]	[116, 1]	[117, 1] <HOLD>	[118, 1]
[93, 2] <HOLD>	[94, 2]	[95, 2]	[96, 2]	[97, 2]	[98, 2]	[99, 2]	[100, 2]	[101, 2]	[102, 2]	[103, 2]	[104, 2]	[105, 2]	[106, 2]	[107, 2]	[108, 2]	[109, 2] <HOLD>	[110, 2]	[111, 2]	[112, 2]	[113, 2]	[114, 2]	[115, 2] <BOOST>	[116, 2]	[117, 2]	[118, 2]
[93, 3]	[94, 3]	[95, 3]	[96, 3]	[97, 3]	[98, 3] <PLAYER>	[99, 3]	[100, 3]	[101, 3] <TRUCK>	[102, 3]	[103, 3]	[104, 3]	[105, 3]	[106, 3]	[107, 3]	[108, 3]	[109, 3]	[110, 3]	[111, 3]	[112, 3]	[113, 3]	[114, 3]	[115, 3]	[116, 3]	[117, 3]	[118, 3]
[93, 4]	[94, 4]	[95, 4]	[96, 4] <HOLD>	[97, 4]	[98, 4]	[99, 4]	[100, 4] <BOOST>	[101, 4]	[102, 4]	[103, 4]	[104, 4]	[105, 4]	[106, 4]	[107, 4]	[108, 4] <HOLD>	[109, 4]	[110, 4]	[111, 4]	[112, 4]	[113, 4]	[114, 4]	[115, 4]	[116, 4]	[117, 4]	[118, 4]

Gambar 4.3.1.2. (atas) Greedy mengambil powerup, (bawah) Greedy menggunakan LIZARD

4.3.1.3. Greedy pada Accel

Player akan melakukan akselerasi dengan boost. Seperti pada **Gambar 4.3.1.3.**, jika sebuah lane kosong dan tidak ada perintah dengan prioritas lebih tinggi seperti *dodge* ataupun *fix*, kondisi untuk melakukan boost akan terpenuhi. Bot akan menggunakan boost sehingga kecepatannya menjadi maksimum, yaitu sebesar 15.

Round 036 Reset [Remove this match](#)

[314, 1]	[315, 1]	[316, 1]	[317, 1]	[318, 1]	[319, 1]	[320, 1]	[321, 1]	[321, 1] <LIZARD>	[323, 1]	[324, 1]	[325, 1]	[326, 1]	[327, 1]	[328, 1]	[329, 1]	[330, 1]	[331, 1]	[332, 1]	[333, 1]	[334, 1] <TWEET>	[335, 1]	[336, 1]	[337, 1]	[338, 1]	[339, 1]
[314, 2]	[315, 2]	[316, 2]	[317, 2]	[318, 2]	[319, 2]	[320, 2]	[321, 2]	[322, 2]	[323, 2]	[324, 2]	[325, 2]	[326, 2]	[327, 2] <GOOL>	[328, 2]	[329, 2]	[330, 2]	[331, 2]	[332, 2]	[333, 2]	[334, 2]	[335, 2]	[336, 2]	[337, 2]	[338, 2]	[339, 2]
[314, 3]	[315, 3]	[316, 3] <ALIJD>	[317, 3]	[318, 3]	[319, 3]	[320, 3] <HOLD SPIN>	[321, 3]	[322, 3]	[323, 3]	[324, 3]	[325, 3]	[326, 3]	[327, 3]	[328, 3]	[329, 3] <PLAYER>	[330, 3]	[331, 3]	[332, 3]	[333, 3]	[334, 3]	[335, 3]	[336, 3]	[337, 3]	[338, 3]	[339, 3]
[314, 4]	[315, 4]	[316, 4]	[317, 4]	[318, 4]	[319, 4] <PLAYER>	[320, 4]	[321, 4]	[322, 4]	[323, 4]	[324, 4]	[325, 4]	[326, 4]	[327, 4]	[328, 4]	[329, 4]	[330, 4]	[331, 4]	[332, 4]	[333, 4]	[334, 4]	[335, 4]	[336, 4]	[337, 4]	[338, 4]	[339, 4]

First < Prev 36 Next > Last >

(Click the button that displays the round number to quickly switch rounds)

Round Details Current Round: 36

A - Eco Racing (selected) ▾

Position 2 <small>[x: 319, y: 4]</small>	Speed 9	Lane 4	Distance 319	Boosts 0	Boosting No	Powerups TWEET,TWEET,TWEET,EMP,BOOST,OIL,OIL,EMP,BOOST

Bot Command

Command: USE_BOOST
 Execution time: 3ms
 Exception: null

Gambar 4.3.1.3. Greedy dalam penggunaan BOOST

4.3.1.4. Greedy pada Offensive

Jika prioritas sudah sampai ke offensive, maka pemain dapat menggunakan beberapa pilihan *command* yang tersedia dalam grup *offensive*.

Prioritas utama adalah pemain dapat menggunakan EMP jika mempunyai *power up* EMP dan player berada di belakang musuh. Karena pemain tidak mengalami damage, tidak perlu menghindar dari *obstacle*, sudah memakai boost sehingga prioritas menjadi offensive dan syarat EMP terpenuhi, pemain akan menggunakan EMP sehingga musuh berhenti. Dapat dilihat dengan berhentinya lawan, pemain yang tertinggal menjadi berada di depan lawan.

Round 125

[Reset](#)[Remove this match](#)

[947, 1]	[948, 1]	[949, 1]	[950, 1]	[951, 1]	[952, 1] <MUD>	[953, 1]	[954, 1]	[955, 1]	[956, 1] <WALL>	[957, 1]	[958, 1]	[959, 1]	[960, 1]	[961, 1] <MUD>	[962, 1] <JUD>	[963, 1]	[964, 1]	[965, 1]	[966, 1]	[967, 1]	[968, 1]	[969, 1]	[970, 1]	[971, 1]	[972, 1]
[947, 2]	[948, 2]	[949, 2]	[950, 2]	[951, 2]	[952, 2] <GOOST>	[953, 2] <MUD>	[954, 2]	[955, 2] <TWEET>	[956, 2]	[957, 2] <MUD>	[958, 2]	[959, 2]	[960, 2]	[961, 2]	[962, 2] <PLAYER>	[963, 2]	[964, 2]	[965, 2]	[966, 2]	[967, 2]	[968, 2]	[969, 2]	[970, 2]	[971, 2]	[972, 2]
[947, 3]	[948, 3]	[949, 3]	[950, 3]	[951, 3]	[952, 3] <PLAYERS>	[953, 3]	[954, 3]	[955, 3]	[956, 3]	[957, 3]	[958, 3]	[959, 3]	[960, 3]	[961, 3]	[962, 3]	[963, 3]	[964, 3]	[965, 3]	[966, 3]	[967, 3]	[968, 3]	[969, 3]	[970, 3]	[971, 3]	[972, 3]
[947, 4]	[948, 4]	[949, 4]	[950, 4] <WALL>	[951, 4]	[952, 4]	[953, 4]	[954, 4]	[955, 4]	[956, 4]	[957, 4]	[958, 4]	[959, 4]	[960, 4]	[961, 4]	[962, 4]	[963, 4]	[964, 4]	[965, 4]	[966, 4]	[967, 4]	[968, 4]	[969, 4]	[970, 4]	[971, 4]	[972, 4]

[First](#) [Prev](#) [125](#) [Next](#) [Last](#)

(Click the button that displays the round number to quickly switch rounds)

A - Eco Racing (selected) ✓



Position	Speed	Lane	Distance	Boosts	Boosting	Powerups
2 [x: 952, y: 3]	15	3	952	3	YES	TWEET,TWEET,T WEET,TWEET,E MP,EMP,TWEET, EMP,EMP,OIL,E MP,OIL,TWEET,L IZARD,TWEET,O IL,TWEET,BOOS T,BOOST

Bot Command

Command: USE_EMP
Execution time: 6ms
Exception: null

Gambar 4.3.1.4.a Bot menggunakan EMP saat bot lawan berada di depan

Jika tidak ada EMP, maka prioritas kedua pemain akan menggunakan TWEET. Dari contoh gambar dibawah, terlihat pemain tidak terkena *damage*, tidak perlu *dodge*, tidak memiliki *boost* sehingga prioritas menjadi *offensive*. Karena pemain tidak punya *power up* EMP, tetapi mempunyai *power up* TWEET, dan pemain berada di belakang lawan, maka pemain akan menerapkan strategi greedy untuk TWEET 2, yakni pemain akan meletakkan *cybertruck* pada posisi *block* musuh + *speed* musuh + 3 dan posisi *lane* yang sama dengan musuh. Pada kasus dibawah, posisi *cybertruck* akan berada pada $(140 + 9 + 3, 2) = (152, 2)$.

Round 020

[Reset](#) [Remove this match](#)

[121, 1]	[122, 1]	[123, 1]	[124, 1]	[125, 1]	[126, 1]	[127, 1]	[128, 1]	[129, 1]	[130, 1]	[131, 1]	[132, 1]	[133, 1]	[134, 1]	[135, 1]	[136, 1]	[137, 1]	[138, 1]	[139, 1]	[140, 1]	[141, 1]	[142, 1]	[143, 1]	[144, 1]	[145, 1]	[146, 1]
[121, 2]	[122, 2]	[123, 2]	[124, 2]	[125, 2]	[126, 2]	[127, 2]	[128, 2]	[129, 2]	[130, 2]	[131, 2]	[132, 2]	[133, 2]	[134, 2]	[135, 2]	[136, 2]	[137, 2]	[138, 2]	[139, 2]	[140, 2]	[141, 2]	[142, 2]	[143, 2]	[144, 2]	[145, 2]	[146, 2]
[121, 3]	[122, 3]	[123, 3]	[124, 3]	[125, 3]	[126, 3]	[127, 3]	[128, 3]	[129, 3]	[130, 3]	[131, 3]	[132, 3]	[133, 3]	[134, 3]	[135, 3]	[136, 3]	[137, 3]	[138, 3]	[139, 3]	[140, 3]	[141, 3]	[142, 3]	[143, 3]	[144, 3]	[145, 3]	[146, 3]
[121, 4]	[122, 4]	[123, 4]	[124, 4]	[125, 4]	[126, 4]	[127, 4]	[128, 4]	[129, 4]	[130, 4]	[131, 4]	[132, 4]	[133, 4]	[134, 4]	[135, 4]	[136, 4]	[137, 4]	[138, 4]	[139, 4]	[140, 4]	[141, 4]	[142, 4]	[143, 4]	[144, 4]	[145, 4]	[146, 4]

(Click the button that displays the round number to quickly switch rounds)

Round 020

[Reset](#) [Remove this match](#)

[135, 1]	[136, 1]	[137, 1]	[138, 1]	[139, 1]	[140, 1]	[141, 1]	[142, 1]	[143, 1]	[144, 1]	[145, 1]	[146, 1]	[147, 1]	[148, 1]	[149, 1]	[150, 1]	[151, 1]	[152, 1]	[153, 1]	[154, 1]	[155, 1]	[156, 1]	[157, 1]	[158, 1]	[159, 1]	[160, 1]
[135, 2]	[136, 2]	[137, 2]	[138, 2]	[139, 2]	[140, 2]	[141, 2]	[142, 2]	[143, 2]	[144, 2]	[145, 2]	[146, 2]	[147, 2]	[148, 2]	[149, 2]	[150, 2]	[151, 2]	[152, 2]	[153, 2]	[154, 2]	[155, 2]	[156, 2]	[157, 2]	[158, 2]	[159, 2]	[160, 2]
[135, 3]	[136, 3]	[137, 3]	[138, 3]	[139, 3]	[140, 3]	[141, 3]	[142, 3]	[143, 3]	[144, 3]	[145, 3]	[146, 3]	[147, 3]	[148, 3]	[149, 3]	[150, 3]	[151, 3]	[152, 3]	[153, 3]	[154, 3]	[155, 3]	[156, 3]	[157, 3]	[158, 3]	[159, 3]	[160, 3]
[135, 4]	[136, 4]	[137, 4]	[138, 4]	[139, 4]	[140, 4]	[141, 4]	[142, 4]	[143, 4]	[144, 4]	[145, 4]	[146, 4]	[147, 4]	[148, 4]	[149, 4]	[150, 4]	[151, 4]	[152, 4]	[153, 4]	[154, 4]	[155, 4]	[156, 4]	[157, 4]	[158, 4]	[159, 4]	[160, 4]

(Click the button that displays the round number to quickly switch rounds)

A - Eco Racing

(selected) ▾

Position 2	Speed 9	Lane 3	Distance 126	Boosts 0	Boosting No	Powerups TWEET,OIL,TWEET,OIL
[x: 126, y: 3]						
Bot Command						
Command: USE_TWEET 2 152						
Execution time: 11ms						
Exception: null						

Round 021

[Reset](#) [Remove this match](#)

[144, 1]	[145, 1]	[146, 1]	[147, 1]	[148, 1]	[149, 1]	[150, 1]	[151, 1]	[152, 1]	[153, 1]	[154, 1]	[155, 1]	[156, 1]	[157, 1]	[158, 1]	[159, 1]	[160, 1]	[161, 1]	[162, 1]	[163, 1]	[164, 1]	[165, 1]	[166, 1]	[167, 1]	[168, 1]	[169, 1]
[144, 2]	[145, 2]	[146, 2]	[147, 2]	[148, 2]	[149, 2]	[150, 2]	[151, 2]	[152, 2]	[153, 2]	[154, 2]	[155, 2]	[156, 2]	[157, 2]	[158, 2]	[159, 2]	[160, 2]	[161, 2]	[162, 2]	[163, 2]	[164, 2]	[165, 2]	[166, 2]	[167, 2]	[168, 2]	[169, 2]
[144, 3]	[145, 3]	[146, 3]	[147, 3]	[148, 3]	[149, 3]	[150, 3]	[151, 3]	[152, 3]	[153, 3]	[154, 3]	[155, 3]	[156, 3]	[157, 3]	[158, 3]	[159, 3]	[160, 3]	[161, 3]	[162, 3]	[163, 3]	[164, 3]	[165, 3]	[166, 3]	[167, 3]	[168, 3]	[169, 3]
[144, 4]	[145, 4]	[146, 4]	[147, 4]	[148, 4]	[149, 4]	[150, 4]	[151, 4]	[152, 4]	[153, 4]	[154, 4]	[155, 4]	[156, 4]	[157, 4]	[158, 4]	[159, 4]	[160, 4]	[161, 4]	[162, 4]	[163, 4]	[164, 4]	[165, 4]	[166, 4]	[167, 4]	[168, 4]	[169, 4]

(Click the button that displays the round number to quickly switch rounds)

Gambar 4.3.1.4.b Bot menggunakan tweet saat bot lawan berada di depan pemain (strategi 1)

Kasus kedua dalam tweet adalah ketika kondisi TWEET memenuhi, tetapi pemain berada di depan musuh dan tersedia lokasi strategis, maka pemain akan menggunakan strategi greedy untuk TWEET 1. Pada gambar dibawah, terdapat posisi strategis yakni di posisi (660, 1) merupakan tempat kosong yang pada block sama namun lane berbeda memiliki 3 mud sehingga akan penuh dengan obstacle dan musuh dijamin mendapat kerusakan.

Round 088

[Reset](#) [Remove this match](#)

[641, 1]	[642, 1]	[643, 1]	[644, 1]	[645, 1]	[646, 1]	PLAYED	[647, 1]	[648, 1]	[649, 1]	[650, 1]	[651, 1]	[652, 1]	[653, 1]	PLAYED	[655, 1]	[656, 1]	[657, 1]	PLAYED	[659, 1]	[660, 1]	[661, 1]	[662, 1]	[663, 1]	[664, 1]	[665, 1]
[641, 2]	[642, 2]	[643, 2]	[644, 2]	[645, 2]	[646, 2]	PLAYED	[647, 2]	[648, 2]	[649, 2]	[650, 2]	[651, 2]	[652, 2]	[653, 2]	[654, 2]	[655, 2]	[656, 2]	[657, 2]	[658, 2]	[659, 2]	[660, 2]	[661, 2]	[662, 2]	[663, 2]	[664, 2]	[665, 2]
[641, 3]	[642, 3]	[643, 3]	[644, 3]	[645, 3]	[646, 3]	[647, 3]	[648, 3]	[649, 3]	[650, 3]	[651, 3]	[652, 3]	[653, 3]	[654, 3]	[655, 3]	[656, 3]	[657, 3]	[658, 3]	[659, 3]	[660, 3]	[661, 3]	[662, 3]	[663, 3]	[664, 3]	[665, 3]	
[641, 4]	[642, 4]	[643, 4]	[644, 4]	[645, 4]	[646, 4]	[647, 4]	[648, 4]	[649, 4]	[650, 4]	[651, 4]	[652, 4]	[653, 4]	[654, 4]	[655, 4]	[656, 4]	[657, 4]	[658, 4]	[659, 4]	[660, 4]	[661, 4]	[662, 4]	[663, 4]	[664, 4]	[665, 4]	

| < First | < Prev | 88 | Next > | Last > |

(Click the button that displays the round number to quickly switch rounds)

Round 088

[Reset](#) [Remove this match](#)

[666, 1]	[669, 1]	[670, 1]	[671, 1]	[672, 1]	[673, 1]	PLAYED	[674, 1]	[675, 1]	[676, 1]	[677, 1]	[678, 1]	[679, 1]	PLAYED	[681, 1]	[682, 1]	[683, 1]	[684, 1]	[685, 1]	[686, 1]	[687, 1]	[688, 1]	[689, 1]	[690, 1]	[691, 1]	[692, 1]	[693, 1]
[666, 2]	[669, 2]	[670, 2]	[671, 2]	[672, 2]	[673, 2]	PLAYED	[674, 2]	[675, 2]	[676, 2]	[677, 2]	[678, 2]	[679, 2]	PLAYED	[681, 2]	[682, 2]	[683, 2]	[684, 2]	[685, 2]	[686, 2]	[687, 2]	[688, 2]	[689, 2]	[690, 2]	[691, 2]	[692, 2]	[693, 2]
[666, 3]	[669, 3]	[670, 3]	[671, 3]	[672, 3]	[673, 3]	[674, 3]	[675, 3]	[676, 3]	[677, 3]	[678, 3]	[679, 3]	PLAYED	[681, 3]	[682, 3]	[683, 3]	[684, 3]	[685, 3]	[686, 3]	[687, 3]	[688, 3]	[689, 3]	[690, 3]	[691, 3]	[692, 3]	[693, 3]	
[666, 4]	[669, 4]	[670, 4]	[671, 4]	[672, 4]	[673, 4]	[674, 4]	[675, 4]	[676, 4]	[677, 4]	[678, 4]	[679, 4]	PLAYED	[681, 4]	[682, 4]	[683, 4]	[684, 4]	[685, 4]	[686, 4]	[687, 4]	[688, 4]	[689, 4]	[690, 4]	[691, 4]	[692, 4]	[693, 4]	

| < First | < Prev | 88 | Next > | Last > |

(Click the button that displays the round number to quickly switch rounds)

A - Eco Racing

(selected) ▾

						
Position 1 [x: 673, y: 1]	Speed 15	Lane 1	Distance 673	Boosts 5	Boosting YES	Powerups TWEET,TWEET,T WEET,EMP,TWE ET,EMP,TWEET,T WEET,TWEET,E MP,EMP,BOOST, BOOST

Bot Command

Command: USE_TWEET 1 660

Execution time: 7ms

Exception: null

Gambar 4.3.1.4.c. Penggunaan TWEET jika pemain berada di depan lawan (Strategi 2)

Jika tidak dapat menggunakan EMP dan TWEET dan pemain berada di depan musuh, maka pemain akan menggunakan oil jika memiliki power up tersebut.

Round 086

[Reset](#) [Remove this match](#)

[653, 1]	[654, 1]	[655, 1]	[656, 1]	[657, 1]	[658, 1]	[659, 1]	[660, 1]	[661, 1]	[662, 1]	[663, 1]	[664, 1]	[665, 1]	[666, 1]	[667, 1]	[668, 1]	[669, 1]	[670, 1]	[671, 1]	[672, 1]	[673, 1]	[674, 1]	[675, 1]	[676, 1]	[677, 1]	[678, 1]
[653, 2]	[654, 2]	[655, 2]	[656, 2]	[657, 2]	[658, 2]	[659, 2]	[660, 2]	[661, 2]	[662, 2]	[663, 2]	[664, 2]	[665, 2]	[666, 2]	[667, 2]	[668, 2]	[669, 2]	[670, 2]	[671, 2]	[672, 2]	[673, 2]	[674, 2]	[675, 2]	[676, 2]	[677, 2]	[678, 2]
[653, 3]	[654, 3]	[655, 3]	[656, 3]	[657, 3]	[658, 3]	[659, 3]	[660, 3]	[661, 3]	[662, 3]	[663, 3]	[664, 3]	[665, 3]	[666, 3]	[667, 3]	[668, 3]	[669, 3]	[670, 3]	[671, 3]	[672, 3]	[673, 3]	[674, 3]	[675, 3]	[676, 3]	[677, 3]	[678, 3]
[653, 4]	[654, 4]	[655, 4]	[656, 4]	[657, 4]	[658, 4]	[659, 4]	[660, 4]	[661, 4]	[662, 4]	[663, 4]	[664, 4]	[665, 4]	[666, 4]	[667, 4]	[668, 4]	[669, 4]	[670, 4]	[671, 4]	[672, 4]	[673, 4]	[674, 4]	[675, 4]	[676, 4]	[677, 4]	[678, 4]

< First < Prev 86 Next > Last >

(Click the button that displays the round number to quickly switch rounds)

A - Eco Racing (selected) ✓

Position	Speed	Lane	Distance	Boosts	Boosting	Powerups
1 [x: 658, y: 1]	15	1	658	5	YES	TWEET,TWEET,T WEET,EMP,OIL,T WEET,EMP,TWE ET,TWEET,TWE ET,EMP,EMP,BOO ST,BOOST,BOOS T

Bot Command

Command: USE_OIL
Execution time: 5ms
Exception: null

Gambar 4.3.1.4.d. Penggunaan OIL

4.3.2. Analisis

4.3.2.1. Efisiensi

Efisiensi algoritma yang kami implementasi cenderung linear pada bagian yang tidak kompleks, tetapi pada beberapa bagian yang dijelaskan di bawah ini mempunyai kompleksitas lain.

4.3.2.1.1. Lane Flagging

Dapat dilihat algoritma di bawah ini yang merupakan lanjutan dari **Gambar 4.1.** algoritma ini melakukan iterasi 2 tingkat, iterasi pada *lane* dan iterasi pada *block* berdasarkan kecepatan player. Tetapi, *lane* yang di cek berjumlah statis, yaitu lane kiri, tengah, dan kanan dari player. Sehingga kompleksitas algoritmanya adalah $O(n)$ dengan n adalah kecepatan player dilengkapi dengan sifat heuristik.

```
// Iterate through blocks in the lane
for (int j = Math.max(car.position.block - StartBlock, 0); j <= car.position.block - StartBlock
    + distance + modifier; j++) {

    // No need to check player position
    if (i - car.position.lane + 2 == 1 && j == 5) {
        continue;
    }

    debug += String.format("distance: %d, modifier: %d\n", distance, modifier);
    debug += String.format("\nlane %d block %d: (%s) ", i, j, lane[j].terrain);

    // If out of bounds, skip it
    if (Extras.isEndOfLane(lane[j])) {
        break;
    }

    // Flag the lane as oil spills if it cannot be lizard'd
    if (i - car.position.lane + 2 == 1 && j == car.position.block - StartBlock + car.getSpeed()) {
        if (lane[j].terrain == Terrain.OIL_SPILL || lane[j].terrain == Terrain.WALL
            || lane[j].terrain.equals(Terrain.MUD)) {
            flags[i - car.position.lane + 2] = Terrain.OIL_SPILL;
            debug += " OIL_SPILL";
            break;
        }
    }

    // If there's a wall or anything occupied by cybertruck or opponent that slower
    // than player, flag it as a wall
    if (lane[j].terrain.equals(Terrain.WALL) || lane[j].isOccupiedByCyberTruck
        || (lane[j].occupiedById > 0 && lane[j].occupiedById != car.id
            && (opponent.position.block
                - car.position.block < (car.getSpeed() - opponent.speed)))) {
        debug += " WALL";
        flags[i - car.position.lane + 2] = Terrain.WALL;
    }
}
```

Gambar 4.3.2.1.1 Lanjutan dari algoritma lane flagging

4.3.2.1.2. CyberTruck Check

Dapat dilihat dari algoritma pada **Gambar 4.5.** algoritma dalam pengecekan lokasi CyberTruck yang cocok sehingga menghalangi jalan merupakan algoritma dengan kompleksitas $O(n \times m)$ dengan n lebar view (jumlah lane) dan m merupakan panjang view (jarak pandang).

4.3.2.2. Efektivitas

Algoritma bot yang kami implementasikan ini efektif dan sesuai tujuan. Pada permasalahan menghindari rintangan, mobil dapat menghindari rintangan tersebut dengan berbelok ke kanan, ke kiri, melakukan penggereman (*decelerate*) ataupun menggunakan *lizard*. Permasalahan mengenai kecepatan juga dapat diatasi dengan perintah *boost* dan

accelerate jika kondisinya memungkinkan. Selain itu, bot ini juga dapat melakukan interaksi dengan bot lawan melalui perintah-perintah yang menyerang lawan. Perintah seperti *tweet*, *oil*, dan *emp* digunakan sesuai kondisinya. Contohnya perintah *emp* digunakan untuk mengejar mobil lawan dengan menghentikan mobil lawan saat posisi bot yang kami buat dalam posisi tertinggal. Untuk *oil*, hanya digunakan apabila posisinya sedang berada di depan mobil lawan. Perintah *tweet* digunakan secara kondisional bergantung pada posisi lawan.

BAB 5

Kesimpulan dan Saran

5.1. Kesimpulan

Dalam pengimplementasian algoritma penyelesaian masalah, dapat digunakan beberapa jenis perspektif. Salah satunya adalah perspektif greedy, dengan melakukan “*take what you can get*” yaitu melakukan hal yang paling menguntungkan pada posisi saat itu tanpa terlalu lama memperhitungkan situasi kedepannya. Penyelesaian greedy ini akan cenderung lebih mudah, cepat, dan efisien tetapi cenderung salah dikarenakan tidak memperhitungkan situasi kedepannya. Karena itu, algoritma greedy cenderung digunakan sebagai salah satu bagian dari sesuatu bagian yang besar. Tergantung kondisi, greedy terhadap apa, untuk memaksimalkan hasil.

5.2. Saran

Dari hasil yang kami peroleh dari implementasi algoritma *greedy* untuk bot Overdrive, kami mempunyai saran sebagai berikut:

1. Pada penggerjaan bot Overdrive dalam waktu yang cukup singkat ini, gunakan lebih banyak referensi bot yang telah ada sehingga tidak banyak memakan waktu pada implementasi yang lebih sederhana.
2. Setiap perubahan yang dibuat, lebih baik dilakukan perbandingan apakah perubahan tersebut lebih baik atau tidak dengan melakukan pengujian dengan bot sendiri tanpa perubahan.

Lampiran

Link Github Repository

<https://github.com/marfgold1/EcoRacing>

Link Video YouTube

https://www.youtube.com/watch?v=l1_Ru5OeYtw

Daftar Pustaka

Munir, Rinaldi (2021). “Algoritma Greedy (Bagian 1)”. [https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2020-2021/Algoritma-Greedy-\(2021\)-Bag1.pdf](https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2020-2021/Algoritma-Greedy-(2021)-Bag1.pdf). Diakses tanggal 13 Februari 2022.

Entelect (2020). “EntelectChallenge/2020-Overdrive - Main repository for Entelect Challenge 2020”. <https://github.com/EntelectChallenge/2020-Overdrive>. Diakses tanggal 2 Februari 2022.

van den Berg, Franché (2020). “Entelect Challenge 2020 - Replayer”. <https://entelect-replay.raezor.co.za/>. Diakses tanggal 2 Februari 2022.