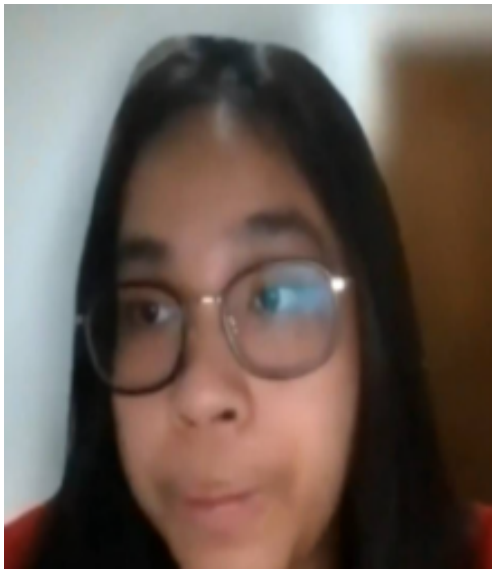
Pemanfaatan Algoritma Greedy dalam Permainan “Overdrive”

LAPORAN TUGAS BESAR

Diajukan Untuk Memenuhi Tugas IF2211 Strategi Algoritma Semester II Tahun 2021/2022



Disusun Oleh :

Nadia Mareta Putri Leiden (13520007)
Daniel Salim (13520008)
Raden Haryosatyo Wisjnunandono (13520070)

TEKNIK INFORMATIKA
INSTITUT TEKNOLOGI BANDUNG BANDUNG
2022

DAFTAR ISI

BAB I
DESKRIPSI TUGAS

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 1.1 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. 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 <lane> <block>
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 II
LANDASAN TEORI

2.1 Algoritma Greedy

Algoritma greedy adalah algoritma yang memecahkan persoalan secara langkah per langkah (step by step) sedemikian sehingga, pada setiap langkah:

1. mengambil pilihan yang terbaik yang dapat diperoleh pada saat itu tanpa memperhatikan konsekuensi ke depan (prinsip "take what you can get now!")

2. dan "berharap" bahwa dengan memilih optimum lokal pada setiap langkah akan berakhir dengan optimum global.

2.1.1 Prinsip Utama Algoritma Greedy

Prinsip utama algoritma greedy adalah "*take what you can get now*". Maksud dari prinsip tersebut adalah pada setiap langkah dalam algoritma greedy, diambil keputusan yang paling optimal untuk langkah tersebut tanpa memperhatikan konsekuensi pada langkah selanjutnya. Solusi tersebut disebut dengan optimum lokal. Kemudian saat pengambilan nilai optimum lokal pada setiap langkah, diharapkan tercapai optimum global, yaitu tercapainya solusi optimum yang melibatkan keseluruhan langkah dari awal sampai akhir.

2.1.2 Elemen Algoritma Greedy

Elemen-elemen yang digunakan dalam penerapan algoritma greedy antara lain:

1. Himpunan kandidat, C: himpunan yang berisi elemen pembentuk solusi atau kandidat yang akan dipilih pada setiap Langkah (misal: simpul/sisi di dalam graf, job, task, koin, benda, karakter, dsb)

2. Himpunan solusi, S: berisi himpunan kandidat yang sudah terpilih.

3. Fungsi solusi: : menentukan apakah himpunan kandidat yang dipilih sudah memberikan solusi. Biasanya dalam bentuk nilai boolean, True jika himpunan solusi yang sudah terbentuk merupakan solusi yang lengkap; False jika himpunan solusi belum lengkap.

4. Fungsi seleksi (selection function): memilih kandidat berdasarkan strategi greedy tertentu. Strategi greedy ini bersifat heuristik.

5. Fungsi kelayakan (feasible): memeriksa apakah kandidat yang dipilih dapat dimasukkan ke dalam himpunan solusi (layak atau tidak) Maksudnya yaitu apakah kandidat tersebut bersama dengan himpunan solusi yang sudah terbentuk tidak melanggar kendala yang ada.

6. Fungsi obyektif : memaksimumkan atau meminimumkan solusi.

Dengan menggunakan elemen-elemen di atas, maka dapat dikatakan bahwa:

Algoritma greedy melibatkan pencarian sebuah himpunan bagian, S, dari himpunan kandidat, C; yang dalam hal ini, S harus memenuhi beberapa kriteria yang ditentukan, yaitu S menyatakan suatu solusi dan S di optimisasi oleh fungsi objektif.

2.2 Game Engine

Game engine adalah sebuah sistem perangkat lunak (software) yang dirancang untuk pembuatan dan pengembangan suatu video game. *Game engine* memberikan kemudahan dalam menciptakan konsep sebuah game yang akan di buat. Mulai dari sistem rendering, *physics*, arsitektur suara scripting, A.I, dan bahkan sistem networking. *Game engine* dapat dikatakan sebagai jiwa dari seluruh aspek sebuah game. Salah satu *game engine* yang paling populer digunakan adalah unity.

2.2.1 Game Engine Overdrive

Game engine dalam game “Overdrive” memiliki komponen-komponen yang dibutuhkan agar game dapat berjalan antara lain:

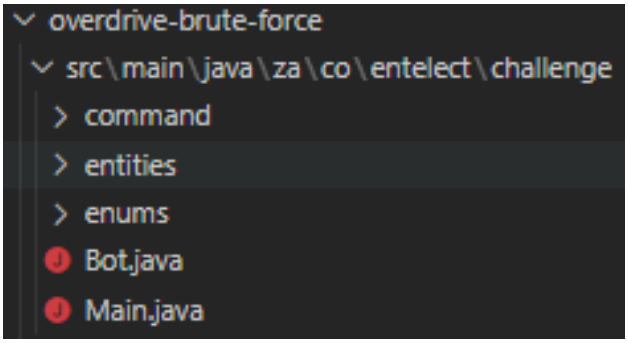
1. *Game Engine Interface*, yaitu *Interface* (antarmuka) yang digunakan oleh *game runner* untuk dapat menjalankan serta mengembangkan game dengan *game engine* yang berbeda.

2. Game Engine, bertanggung jawab dalam membuat *bot* yang dikembangkan oleh pemain dapat mematuhi aturan yang telah dibuat. *Game engine* bertugas menyalurkan perintah yang dikirimkan oleh *bot* kedalam game untuk diproses jika perintah yang diberikan valid.
3. *Game Runner*, bertanggung jawab dalam menjalankan pertandingan antar pemain (PvP). *Game runner* menerima perintah dari *bot* dan mengirimkannya ke game engine untuk dieksekusi.
4. *Reference Bot*, yaitu *bot* yang dibuat dengan beberapa strategi agar dapat diuji dengan *bot* yang akan dibuat.
5. *Starter Bo*t, yaitu *bot* awal yang memiliki logika dasar yang dapat digunakan sebagai referensi dalam pengembangan *bot*.

Langkah pertama untuk membangun *bot* adalah dengan men-*download* *starter-pack.zip* dari tautan berikut. https://github.com/EntelectChallenge/2020-Overdrive/releases/tag/2020.3.4. Dalam membuat *bot* terdapat banyak bahasa permrograman yang dapat digunakan. Namun, dalam tugas ini akan dibuat *bot* dengna bahasa pemrograman Java. Setelah men-*download*, extract *starter-pack.*zip dan masuk ke dalam folder "starter-bot" (starter-pack\starter-bot\java\src\main\java\za\co \entelect\challenge). Di dalamnya terdapat 3 folder dan 2 file, berikut penjelasannya:

1. Folder command: Folder ini berisikan kelas dari perintah yang dapat digunakan oleh pemain. Perintah yang disediakan adalah Accelerate, Boost, ChangeLaneCommand, Command, DecelerateCommand, DoNothingCommand, EmpCommand, FixCommand, LizardCommand, OilCommand, TweetCommand. Semua kelas tersebut mengimplementasikan interface command.
2. Folder entities: Folder ini berisikan kelas entitas yang ada pada game, contohnya adalah Car, GameState, Lane, Position.
3. Folder enums: Folder ini berisikan entitas yang memiliki nilai berupa konstanta.Contohnya Powerup terdiri dari Boost, Oil, Tweet, Lizard, EMP.
4. Bot.Java: File ini berisikan logika dari bot yang dibangun. Pada file ini akan diimplementasikan strategi greedy dengan memanfaatkan kelas lain yang ada. Semua strategi akan ditulis pada method run() di kelas bot.

5. Main.Java: File ini adalah file yang bertanggung jawab untuk menjalankan Bot.Java dengan menginstansiasi bot pada setiap ronde dan menerima perintah dari bot tersebut untuk dikirimkan ke game engine untuk kemudian diproses.



Gambar 2.1 Isi folder starter-bot

Untuk dapat menjalankan permainan, dibutuhkan beberapa *requirement* dasar sebagai yaitu, Java (minimal Java 8), IntelliJ IDEA, NodeJS. Untuk menjalankan permainan, dapat membuka file "run.bat" (Untuk Windows dapat buka dengan double-click, Untuk Linux/Mac dapat menjalankan command "make run").

Secara default, permainan akan dilakukan diantara reference bot (default-nya berbahasa Java) dan starter bot (default-nya berbahasa JavaScript) yang disediakan. Untuk mengubah hal tersebut, edit file "game-runner-config.json" di dalam folder "starter-pack". Selain itu, kita juga dapat mengubah file "bot.json" dalam direktori "starter-bots" untuk mengatur informasi terkait bot yang dibuat.

Untuk dapat menjalankan visualizer, harus mendownload visualizernya di https://github.com/Affuta/overdrive-round-runner. Setelah menjalankan pertandingan melalui cmd maka hasilnya secara default akan tersimpan di folder match-logs. Kemudian folder pertandingan yang ingin divisualisasikan harus dipindahkan ke dalam folder matches di visualizer. Folder pertandingan memiliki format nama seperti timestamp pertandingannya sehingga hanya perlu mengingat kapan pertandingan tersebut dijalankan untuk mendapatkan folder yang tepat. Setelah memindahkan folder pertandingan ke visualizer, jalankan aplikasi visualizer dan pertandingan sudah siap ditonton dengan menggunakan tampilan GUI.

2.2.3 Mengembangkan Starter Bot

Starter bot telah menyediakan berbagai macam kelas beserta method dan attributenya, namun kelas tersebut masih dapat dikembangkan lagi dan ditambah jumlahnya. Contohnya starter-bot sudah menyediakan kelas untuk perintah menggunakan powerup Boost, Oil, dan EMP namun belum menyediakan perintah untuk menggunakan powerup Tweet dan Lizard.

```java
package za.co.entelect.challenge.command;

public class TweetCommand implements Command {
    private int lane;
    private int block;

    public TweetCommand (int lane, int block) {
        this.lane = lane;
        this.block = block;
    }

    @Override
    public String render() {
        return String.format("USE_TWEET %d %d",lane,block);
    }

}
```

```java
package za.co.entelect.challenge.command;

public class LizardCommand implements Command {

    @Override
    public String render() {
        return String.format("USE_LIZARD");
    }
}
```

Gambar 2.2 Command Tweet (kiri) dan Lizard (kanan)

Setelah mempersiapkan semua kelas dan elemen baru yang diperlukan, langkah selanjutnya adalah memasukkan strategi greedy yang diinginkan pada file bot.java. Starter bot sudah menyediakan beberapa method dan attribute awal yang bisa digunakan dalam pengembangan bot. Method dan attribute yang disediakan juga bisa diubah dan ditambah lagi untuk melengkapi strategy greedy.

Semua strategi greedy yang ditulis akan diimplementasikan pada method run dan pastikan method run harus memberikan return sebuah perintah yang valid agar dideteksi

oleh game engine. Manfaatkan fitur OOP yang disediakan Java untuk membuat strategi greedy. Isi dari bot.java adalah sebagai berikut:

```java
package za.co.entelect.challenge;

import jdk.internal.net.http.websocket.WebSocketImpl;
import za.co.entelect.challenge.command.*;
import za.co.entelect.challenge.entities.*;
import za.co.entelect.challenge.enums.PowerUps;
import za.co.entelect.challenge.enums.Terrain;

import java.util.*;

import static java.lang.Math.max;

public class Bot {

    private List<Integer> directionList = new ArrayList<>();

    private Random random;
    private GameState gameState;
    private Car opponent;
    private Car myCar;
    private int maxSpeed;


    private final static Command ACCELERATE = new AccelerateCommand();
    private final static Command LIZARD = new LizardCommand();
    private final static Command OIL = new OilCommand();
    private final static Command BOOST = new BoostCommand();
    private final static Command EMP = new EmpCommand();
    private final static Command FIX = new FixCommand();

    private final static Command TURN_RIGHT = new ChangeLaneCommand(1);
    private final static Command TURN_LEFT = new ChangeLaneCommand(-1);
```

Gambar 2.3 Isi file Bot.java

BAB III
APLIKASI STRATEGI GREEDY

3.1 Mapping Persoalan Menjadi Elemen-Elemen Algoritma Greedy

Mapping Elemen-Elemen Greedy:

a. Himpunan Kandidat: Himpunan command yang tersedia pada *game engine*

b. Himpunan Solusi: Command-command yang terpilih

c. Fungsi Solusi: Memeriksa apakah command-command yang dipilih dapat membawa mobil sampai garis finish

d. Fungsi Seleksi: Pilihlah command yang memaksimumkan jarak tempuh dan poin yang didapat dan juga meminimumkan jarak tempuh dan poin mobil lawan

e. Fungsi Kelayakan: Memeriksa apakah command yang dipilih dapat dieksekusi dan apakah himpunan command yang sudah dipilih telah membawa mobil melewati garis finish

f. Fungsi Objektif: Jumlah command yang digunakan minimum

3.2 Eksplorasi Alternatif Solusi Greedy

3.2.1 Greedy by Obstacle/Lane Weight)

Greedy by obstacle adalah strategi pemilihan *lane* dengan jumlah obstacle paling sedikit secara bobot (bukan jumlah). Bobot lane ditentukan oleh penjumlahan dari bobot *block* sebanyak *speed blocks* di depan mobil kita (contoh: jika kecepatan mobil kita 5 maka akan dihitung jumlah bobot 5 *blocks* di depan mobil kita). Bobot *block* ditentukan oleh *terrain/power up* yang dimilikinya. Perincian bobot berdasarkan *terrain/power up* adalah sebagai berikut:

- *Terrain* MUD = 2
- *Terrain* OIL = 2
- *Terrain* WALL = 3
- *Terrain* CYBERTRUCK = 5
- *Power Up* BOOST = (-3)
- *Power Up* TWEET = (-2)
- *Power Up* LIZARD = (-3)
- *Power Up* EMP = (-2)
- *Power Up* OIL_POWER = (-2)

Pemilihan bobot diatas berdasarkan efek dari melewati *block* tersebut. Semakin besar *damage* dan pengurangan kecepatannya semakin besar bobotnya. Semakin mengandung *power up* yang berkorelasi langsung dengan kecepatan mobil (BOOST & LIZARD) maka akan semakin kecil bobotnya. Pada tiap ronde, akan dihitung bobot obstacle dari lane tempat mobil berada serta bobot obstacle dari lane tetangganya (ex: jika mobil di lane 2 maka akan juga dihitung bobot lane 1&3, jika di lane 1 maka hanya bobot lane 2 saja) lalu akan dimasukkan command untuk menyesuaikan arah ke lane yang memiliki bobot paling kecil.

i. Analisis Efisiensi Solusi

Apabila penghitungan bobot tiap *lane* tidak diperhitungkan maka algoritma dari strategi *greedy by obstacle* ini hanya membutuhkan algoritma *sorting* saja untuk menggunakannya. Apabila *sorting* dan penghitungan bobot tidak diperhitungkan maka hanya dibutuhkan 1 buah loop yang dijalankan selama posisi mobil masih belum sampai garis finish dan dalam tiap iterasi loop akan memilih lane dengan bobot terkecil. Oleh karena hanya mengandung 1 buah loop maka kompleksitas waktu strategi ini untuk n buah pilihan command adalah O(n)

ii. Analisis Efektivitas Solusi

Solusi ini efektif apabila:

- Mobil tidak terkena efek power up EMP lawan. Jika terkena efek EMP maka solusi ini tidak dapat terpakai
- Mobil berada di belakang lawan sehingga tidak perlu melakukan penghindaran terhadap potensi EMP
- Mobil tidak memiliki power up yang tersedia

Solusi ini tidak efektif apabila:

- Mobil berada di depan lawan sehingga ada potensi terkena EMP apabila arah geraknya tidak mempertimbangkan posisi lane lawan
- Mobil terkena efek power up EMP
- Mobil memiliki power up

3.2.2 Greedy by Available Command (Power Up)

Greedy by available command adalah strategi pemilihan command berdasarkan command yang tersedia / power up yang dimiliki. Untuk dapat dikategorikan sebagai available command, power up tidak hanya harus dimiliki, tetapi juga kondisi eksekusinya dapat maksimum. Prasyarat suatu command dapat dikategorikan sebagai *available* adalah sebagai berikut:

- USE_BOOST: memiliki power up, pada 15 blok didepan mobil tidak terdapat obstacle (mud/wall/oil/truck), damage mobil = 0
- USE_LIZARD: memiliki power up, blok di depan sejauh *speed* mobil dalam jalur mobil dan *speed* mobil - 1 dalam jalur tetangga memiliki obstacle (mud/wall/oil/truck)

- USE_TWEET: Ketersediaan tweet > 0. Tidak melakukan respawn cybertruck
- USE_EMP: memiliki power up, berada di belakang mobil musuh, dan berada di jalur yang sama dengan musuh
- USE_OIL: Mobil memiliki power up dan berada di depan musuh

i. Analisis Efisiensi Solusi

Strategi ini mengandalkan pengurutan prioritas dari command yang tersedia. Apabila perhitungan pengurutan prioritas dan pengecekan ketersediaan command tidak diperhitungkan dalam penghitungan kompleksitas waktu dalam strategi ini, maka hanya dibutuhkan n buah iterasi dalam pemilihan command yang ingin dieksekusi. Oleh karena itu, strategi greedy by available command ini memiliki kompleksitas waktu O(n).

ii. Analisis Efektivitas Solusi

Solusi ini efektif apabila:

- Mobil memiliki power up yang layak untuk dieksekusi
- Mobil berada di belakang lawan sehingga tidak perlu melakukan penghindaran terhadap potensi EMP
- Sebaran power banyak dengan densitas yang rata

Solusi ini tidak efektif apabila:

- Mobil berada di depan lawan sehingga ada potensi terkena EMP apabila arah geraknya tidak mempertimbangkan posisi lane lawan
- Mobil tidak memiliki power up yang layak untuk dieksekusi
- Mobil berada pada kondisi yang mengubah prioritas power up (ex: jika garis finish <= 15 tile maka lebih menguntungkan menggunakan BOOST ketimbang EMP, dll)

3.2.3 Greedy by Enemy Position

Greedy by enemy position adalah strategi pemilihan command berdasarkan posisi musuh. Jika berada di belakang musuh maka mobil akan memilih command yang mengarah pada lane tempat musuh berada. Jika berada di depan musuh maka mobil akan mengeksekusi command untuk berpindah dari lane jika berada pada lane yang sama dengan musuh.

i. Analisis Efisiensi Solusi

Terdapat maksimum 3 buah lane yang dijadikan pertimbangan pada tiap round (3), apabila lane yang terpilih adalah lane tetangga maka harus belok ke arah lane tetangga (1), apabila yang terpilih adalah lane tempat mobil maka akan melakukan "ACCELERATE" (1).

$$T(n) = 3 * 1 = O(1)$$

ii. Analisis Efektivitas Solusi

Solusi ini efektif apabila:

- Mobil musuh memiliki power up EMP dan berada pada di belakang mobil kita pada lane yang sama
- Mobil memiliki power up EMP dan berada pada di belakang mobil musuh pada lane yang berbeda
- Sebaran obstacle pada track merata di semua lane

Solusi ini tidak efektif apabila:

- Kedua mobil tidak ada yang memiliki power up EMP
- Obstacle kebanyakan menumpuk di 1 lane yang sama

3.2.4 Greedy by OffensiveCommands

Greedy by OffensiveCommands adalah strategi yang memprioritaskan eksekusi command-command attack. Command-command attack antara lain adalah USE_TWEET, USE_EMP, dan USE_OIL. Mobil akan memprioritaskan penggunaan command-command penyerangan ini, jika tidak memilikinya, mobil akan memprioritaskan untuk berpindah jalur untuk menyamakan lane mobil dengan lane musuh. Karena sifatnya yang bisa dipakai berulang kali dan dimana saja, command ini akan memprioritaskan command USE_TWEET.

i. Analisis Efisiensi Solusi

Apabila algoritma untuk deployment power up (terutama USE_TWEET) tidak diperhitungkan maka algoritma ini hanya membutuhkan 1 kali iterasi dari himpunan command yang sudah diurutkan berdasarkan prioritasnya. Oleh karena itu solusi greedy by offensive commands ini jika memiliki n buah pilihan command maka kompleksitas waktunya adalah O(n).

ii. Analisis Efektivitas Solusi

Solusi ini efektif apabila:

- Mobil berhasil mendapatkan power up tweet
- Mobil musuh tidak memiliki algoritma penghindaran terhadap cybertruck
- Tidak terlalu banyak terdapat obstacle pada ane awal tempat mobil mendapatkan power up tweet.

Solusi ini tidak efektif apabila:

- Mobil musuh memiliki algoritma penghindaran terhadap cybertruck
- Banyak terdapat obstacle pada ane awal tempat mobil mendapatkan power up tweet.
- Mobil tidak berhasil mendapatkan power up tweet

3.2.5 Greedy by Score (PowerUp Hunt)

Greedy by score adalah strategi yang mengutamakan pencarian dan penggunaan power up. Pada strategi ini akan memprioritaskan penggunaan power up jika memilikinya dan jika tidak memiliki power up maka akan memprioritaskan berbelok ke arah lane-lane yang memiliki blok power up yang banyak. Meskipun efek terhadap damage dan speed sama, algoritma ini memberikan perbedaan bobot terhadap terrain mud dan oil spill karena efek oil terhadap poin lebih besar.

i. Analisis Efisiensi Solusi

Algoritma dalam strategi ini meliputi iterasi sekali dari himpunan command-command yang tersedia untuk memeriksa ketersediaan command power up dan algoritma pemilihan lane yang memiliki power up terbanyak. Jika penghitungan bobot lane tidak masuk dalam kalkulasi kompleksitas waktu algoritma, maka kompleksitas waktu strategi ini dengan n buah power up dan n buah lane yang diperiksa adalah O(n^2)

ii. Analisis Efektivitas Solusi

Solusi ini efektif apabila:

- Sebaran Power up dengan obstacle tidak sama (tile obstacle dengan tile power up tidak berada di lane yang sama)
- Terdapat banyak sebaran power up lizard karena lizard meskipun merupakan power up, tetapi juga dapat menghindari efek negatif obstacle.

Solusi ini tidak efektif apabila:

- Sebaran Power up dengan obstacle sama (tile obstacle dengan tile power up berada di lane yang sama)
- Tidak terdapat banyak sebaran power up lizard karena lizard meskipun merupakan power up, tetapi juga dapat menghindari efek negatif obstacle.

3.2.6 Greedy by Damage

Greedy by damage adalah strategi yang memprioritaskan command FIX dalam solusinya. Fix akan diprioritaskan apabila damage mobil sudah lebih besar atau sama dengan 2 dan tidak ada obstacle di depan lane mobil berada maupun di lane-lane tetangga mobil. Selain itu, strategi ini juga memprioritaskan ACCELERATE apabila speed mobil sekarang masih kurang dari maximum speed mobil pada damage point mobil.

i. Analisis Efisiensi Solusi

Strategi ini mengandalkan pengurutan prioritas dari command yang tersedia serta pengecekan apakah ada obstacle di lane depan mobil dan di lane tetangga mobil. Apabila algoritma untuk mengecek obstacle di lane tidak diperhitungkan dalam penghitungan kompleksitas waktu strategi ini, maka hanya dibutuhkan n buah iterasi dalam pemilihan command yang ingin dieksekusi. Oleh karena itu, strategi greedy by damage ini memiliki kompleksitas waktu O(n).

ii. Analisis Efektivitas Solusi

Solusi ini efektif apabila:

- Musuh seringkali menggunakan cybertruck dan oil karena kenampakan keduanya dalam track tidak dapat diduga dan memperlambat mobil serta memberikan damage kepada mobil
- Sebaran power up dengan obstacle sama (tile obstacle dengan tile power up berada di lane yang sama) sehingga sering mengakibatkan damage akibat mencari power up

Solusi ini tidak efektif apabila:

- Musuh jarang atau tidak memprioritaskan penggunaan cybertruck dan oil karena kenampakan keduanya dalam track tidak dapat diduga dan memperlambat mobil serta memberikan damage kepada mobil

- Sebaran power up dengan obstacle tidak sama (tile obstacle dengan tile power up berada di lane yang sama) sehingga jarang mengakibatkan damage akibat mencari power up

3.3 Strategi Algoritma yang Dipilih dan Pertimbangannya

Pada permainan Overdrive ini terdapat beberapa algoritma-algoritma greedy yang memaksimalkan kemungkinan kita untuk menang. Algoritma - algoritma ini memiliki kelebihan dan kekurangannya masing-masing tergantung pada jenis track dan musuh yang kita dapat. Di antara alternatif algoritma-algoritma greedy yang sudah disampaikan pada sub bab 3.2, Algoritma Greedy by available command, greedy by damage, dan greedy by obstacle lah yang paling efektif dalam kondisi maksimumnya masing-masing.

Pada akhirnya algoritma greedy yang kami pakai merupakan gabungan dari ketiga algoritma greedy tersebut. Kami menamai algoritma greedy final kami dengan greedy by speed. Untuk permulaan kami melakukan greedy by fix untuk mengecek apakah mobil butuh diperbaiki. Selanjutnya apabila mobil tidak perlu diperbaiki maka akan menggunakan greedy by available command yang dicampur dengan greedy by obstacle untuk pertimbangan pindah lane-nya.

Kami tidak menggunakan greedy by offensive attack karena algoritma tersebut terlalu bergantung pada adanya power up offense yang terlalu acak. Kami juga tidak menggunakan algoritma greedy by enemy position karena terlalu bergantung pada strategi apa yang musuh gunakan. Kami juga tidak menggunakan greedy by score karena kompleksitas waktunya paling tinggi dan juga tidak memaksimalkan efek dari tiap power up.

BAB IV
IMPLEMENTASI DAN PENGUJIAN

4.1. Pseudocode

*Note: Pseudocode dibawah ini hanya mencakup strategi greedy utama saja, untuk method-method pendukung dan lebih lengkapnya secara umum dapat dilihat pada source code yang dapat diakses pada link ini https://github.com/nandono206/overdrive-greedy

```
// inisialisasi greedy by obstacle

// Mencari lane dengan bobot paling rendah pada tiap round

function GreedybyObstacle(integer: lane, integer: block, int speed) -> integer

        KAMUS LOKAL

        number, current lane : integer



        //laneRisk adalah fungsi yang menghitung bobot tiap lane (lebih lengkapnya lihat source
        code)

        laneRisk() : function (integer lane, integer block, integer speed) -> integer



        ALGORITMA

        number <- 0

        current_lane <- perhitungan lane risk saat ini



        //Jika berada pada lane paling kiri (1) maka yang dibandingkan hanya lane sekarang dan
        lane sebelah kanan

        if (lane == 1) maka

                bobot lane 2 <- laneRisk()

                jika bobot lane 1 < lane 2 maka

                        number <- 1

                else

                        number <- 2
```

```
//Jika berada pada lane 2 atau 3 maka membandingkan dengan kedua lane tetangga
else if (lane == 2) maka
        bobot lane 1 <- laneRisk()
        bobot lane 3 <- laneRisk()
        if bobot lane 1 < bobot lane 2 dan bobot lane 1 < bobot lane 3
                number <- 1
        else if bobot lane 2 < bobot lane 3 dan bobot lane 2 < bobot lane 1
                number <- 2
        else if bobot lane 3 < bobot lane 2 dan bobot lane 3 < bobot lane 1
                number <- 3
else if (lane == 3) maka
        bobot lane 2 <- laneRisk()
        bobot lane 3 <- laneRisk()
         if bobot lane 2 < bobot lane 3 dan bobot lane 2 < bobot lane 4
                number <- 2
        else if bobot lane 3 < bobot lane 2 dan bobot lane 3 < bobot lane 4
                number <- 3
        else if bobot lane 4 < bobot lane 2 dan bobot lane 4 < bobot lane 3
                number <- 4
else


//Jika berada pada lane paling kiri (4) maka yang dibandingkan hanya lane sekarang dan
lane sebelah kiri
        bobot lane 3 <- laneRisk()
        jika bobot lane 3 < lane 4 maka
                number <- 2
        else
                number <- 4
```

```
-> number

//Strategi utama

function greedy by speed() -> Command

// GREEDY BY DAMAGE

//Jika damage mobil sudah > 5 lakukan fix dulu

       if (damage >= 5)
            return FIX;

 //Jika mobil sedang berhenti accelerate terlebih dahulu
        if (speed == 0)
            return ACCELERATE;



//IF NO OBSTACLE AHEAD AND SPEED < MAX SPEED ACCELERATE
       if (speed != maxSpeed and no obstacle ahead)
            return ACCELERATE

// FIX jika damage lebih 2 dan tidak ada obstacle di depan -> FIX
        if (damage >= 2 && no obstacle ahead)
           Return FIX



//GREEDY BY AVAILABLE COMMANDS
       //JIKA BOOST TERSEDIA --> USE BOOST
        if (IsBoostAvailable())
            return BOOST;

        //JIKA LIZARD TERSEDIA  --> USE LIZARD
        if (isLizardAvailable())
            return LIZARD;

       // JIKA TIDAK TERSEDIA LIZARD MAUPUN BOOST PILIH LANE DENGAN BOBOT TERENDAH UNTUK
DILALUI PADA ROUND BERIKUTNYA
//Jika hasil greedy by obstacle tidak sama dengan lane sekarang maka butuh untuk pindah jalur
if(hasil greedy by obstacle != lane mobil sekarang)
       //Jika mobil di lane 1, belok kanan
            if(lane == 1)
                return TURN RIGHT
```

```
        //Jika mobil di lane 2
            else if(lane == 2)
               //Jika hasil greedy by obstacle adalah 3 maka belok kanan
                if(hasil greedy by obstacles == 3)
                    return TURN RIGHT
               //Jika bukan, belok kiri
                else
                    return TURN LEFT

         //Jika mobil di lane 3
            else if(lane == 3)
               //Jika hasil greedy by obstacle adalah 4 maka belok kanan
                if(hasil greedy by obstacle == 4)
                    return TURN RIGHT
               //Jika bukan, belok kiri
                else
                    return TURN LEFT

              //JIKA DI LANE 4 MAKA BELOK KIRI
               else
                   return TURN LEFT


        /* --- GREEDY BY AVAILABLE COMMAND LANJUTAN --- */
         /* --- EMP --- */
/* Kalo lane sama, mobil kita ada di belakang mobil lawan, dan punya EMP, maka pakai EMP*/
        if (isEMPAvailable())
            return EMP;


     /* --- TWEET --- */
         // IF TWEET > 0 USE TWEET
         if (isTweetAvailable())
             //DEPLOY TWEET AT ENEMY LANE, ENEMY BLOCK + ENEMY SPEED AFTER ACCELERATION + 1
             return USE_TWEET enemyLane, enemyPos + speed + 1;

  /* --- OIL --- */
// IF IN FRONT AND HAS OIL --> USE OIL
        if (isOilAvailable())
            return OIL;
```

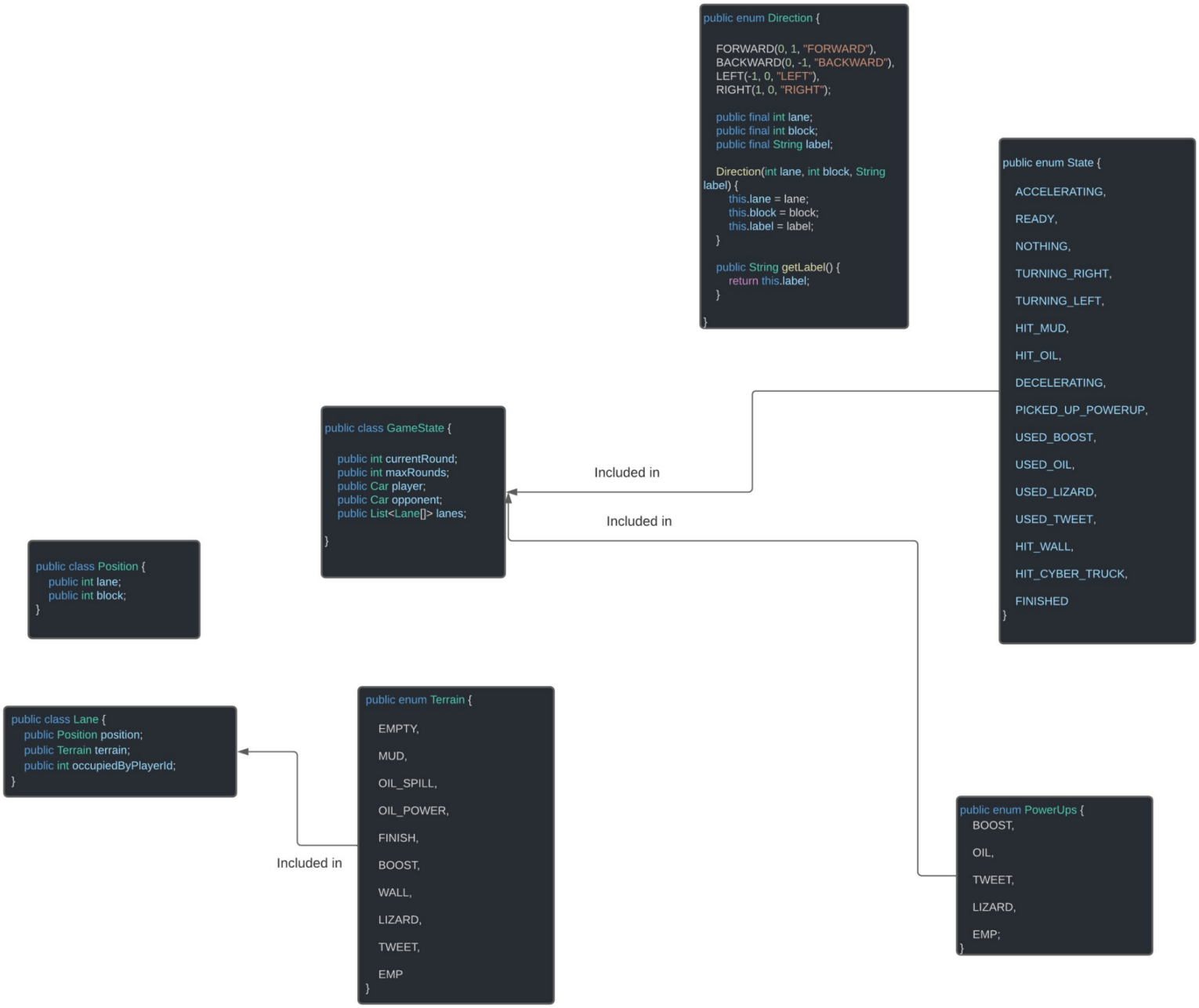
4.2. Struktur Data

4.2.1 Commands

Keterangan:

1. Ada beberapa command yang berhubungan dengan *power-up*: LizardCommand(), TweetCommand(), EmpCommand(), OilCommand()
2. Seluruh class yang ada di bawah *directory* dari *command* mengimplementasikan satu class utama yaitu *public interface Command()*

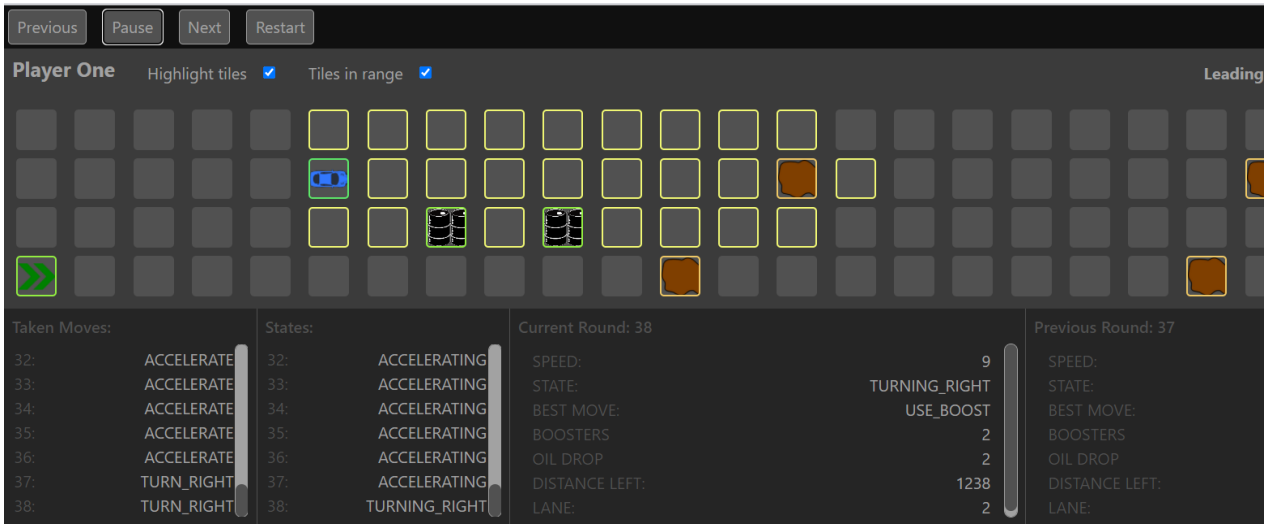
4.2.2 Entities



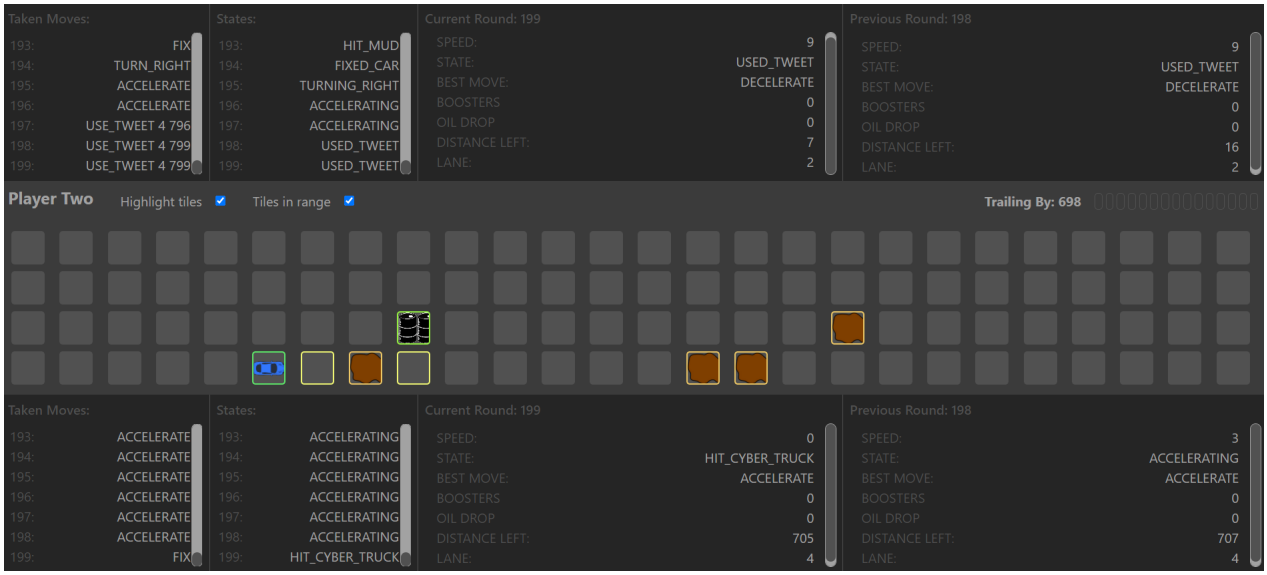
Keterangan:

1. Ada beberapa *entities* yang menyusun game ini yaitu: *Position, Game State, Lane*
2. *Entities* tersebut disusun oleh beberapa enum yang terdiri dari: *State, Direction, Terrain, PowerUps*

4.3. Analisis dan Pengujian

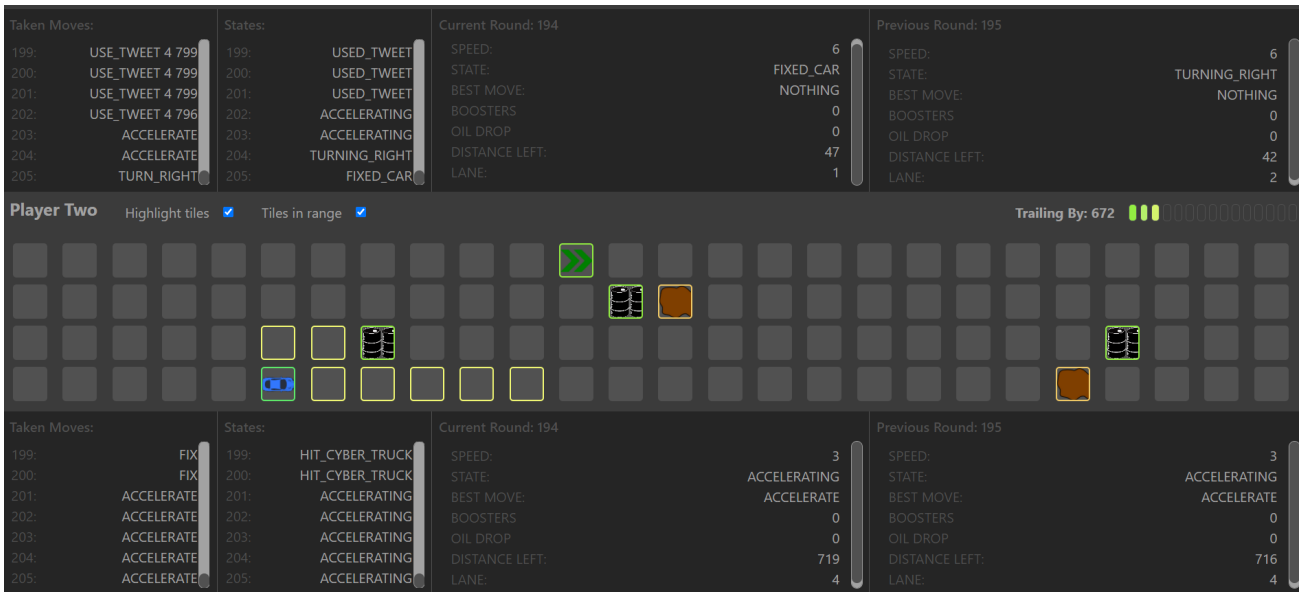


Pada gambar di atas dapat dilihat bahwa bot yang sudah diprogram dengan strategi *greedy* melakukan turn-right untuk mengambil boost berupa oil. Mobil juga dalam kecepatan 9 dan telah menggunakan boost yang ada di dalam game.

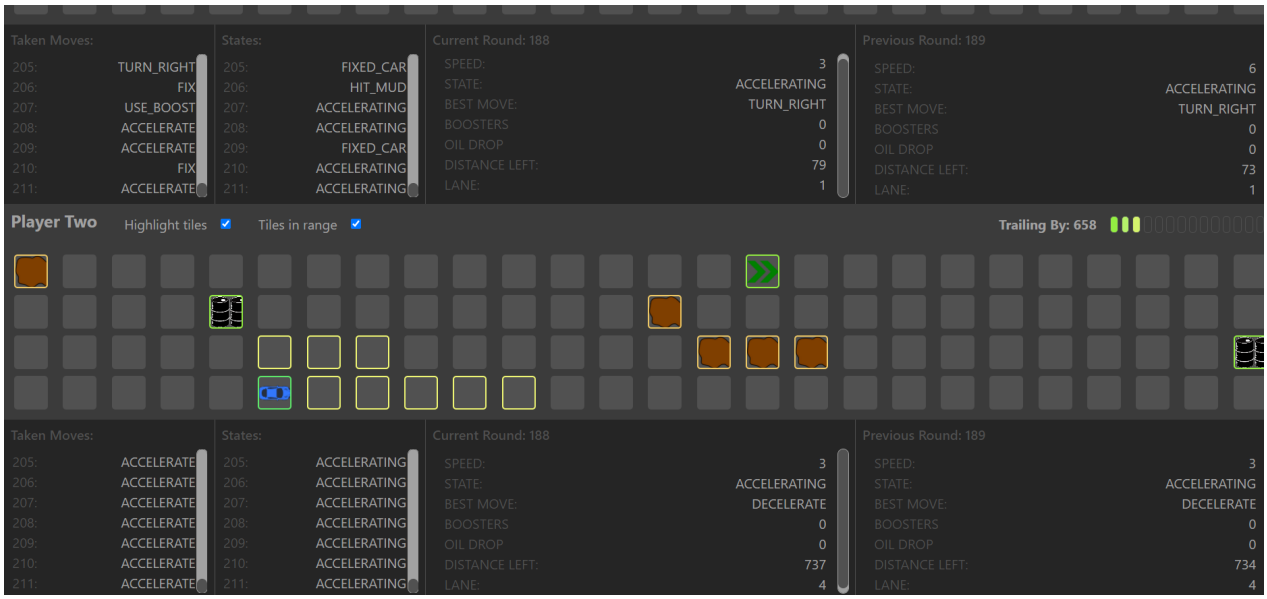


Pada round terakhir dapat dilihat bahwa kecepatan akhir dari bot yang sudah diprogram dengan strategi *greedy* adalah 9, sedangkan *reference bot* pada ronde 198 memiliki kecepatan sebesar 3. Berarti dapat dilihat bahwa penggunaan command Accelerate() kurang maksimal pada

reference-bot, ditambah lagi pada ronde 199 terlihat jelas bahwa adanya penggunaan TweetCommand() oleh bot yang sudah diprogram dengan strategi *greedy*, sehingga dapat dilihat bahwa akhirnya *reference-bot* menabrak Cybertruck yang telah dikeluarkan oleh bot yang dioptimalisasi.



Terlihat pada gambar di atas bahwa state dari *reference bot* berkali-kali menabrak Cyber Truck yang digunakan oleh bot kami. Terdapat perbedaan kecepatan yang signifikan dan penggunaan Command yang lebih variatif pada bot yang telah diprogram dengan strategi *greedy*. Sedangkan pada bot yang *reference bot*, terlihat bahwa Command yang digunakan monoton karena hanya menggunakan Accelerate().



Pada baris “distance left” terlihat bahwa perbedaannya cukup jauh. Pada *reference bot* jarak yang tersisa masih sekitar 737 block, sedangkan pada bot yang sudah diprogram dengan *greedy* hanya memiliki 79 block menuju garis *finish*.

BAB V
KESIMPULAN DAN SARAN

5.1 Kesimpulan

Dalam pengerjaan tugas besar IF2211 Strategi Algoritma semester 2 2021/2022 berjudul "Pemanfaatan Algoritma Greedy dalam Permainan "Overdrive" kami berhasil membuat sebuah bot dalam permainan yang memanfaatkan algoritma greedy. Pada implementasinya, kami menggunakan kombinasi berbagai strategi greedy supaya hasil yang terjadi seoptimal mungkin. Setelah mengerjakan tugas besar ini kami menjadi tahu cara membuat bot di dalam permainan video dan salah satu caranya adalah menggunakan strategi greedy.

5.2 Saran

Saran-saran yang dapat kami berikan untuk tugas besar IF2211 Strategi Algoritma semester 2 2021/2022 adalah sebagai berikut:

- Masih terdapat ruang untuk lebih mengoptimasi algoritma yang ada dalam program ini. Program yang dibuat masih merupakan perkiraan terhadap solusi optimum global dan masih perlu dibuktikan secara matematis
- Algoritma yang digunakan pada Tugas Besar ini masih memiliki kekurangan - kekurangan seperti penulisan algoritma yang sama berulang kali yang dapat di efisienkan lagi menggunakan fungsi-fungsi dan objek-objek sehingga program lebih modular
- Memperjelas *comment* pada source code sehingga memudahkan pihak lain dalam memahaminya sekaligus mempermudah kolaborasi antar pihak

LINK GITHUB DAN YOUTUBE

Link Repository GitHub (https://github.com/nandono206/overdrive-greedy)
Link YouTube (https://youtu.be/e7O4hI-Paig)

DAFTAR PUSTAKA

Affuta. (2020, April 10). overdrive-round-runner. https://github.com/Affuta/overdrive-round-runner (diakses tanggal 14 Februari 2022)
(Visualizier)

EntelectChallenge. (2020, Agusuts 17). 2020-Overdrive https://github.com/EntelectChallenge/2020-Overdrive (diakses tanggal 11 Februari 2022)
(Entellect Github)

Munir, Rinaldi. (2021). Algotitma Greedy, https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2020-2021/Algoritma-Greedy-(2021)-Bag1.pdf (diakses tanggal 17 Februari 2022).
(Materi tentang Greedy)

Apa itu game engine. (2011, Juni 26). https://menariksekali.wordpress.com/ 2011/06/22/ apa-itu-game-engine/ (diakses tanggal 18 Februari 2022).
(Game Engine)