

LAPORAN TUGAS BESAR
STRATEGI ALGORITMA

PEMANFAATAN ALGORITMA GREEDY DALAM APLIKASI
PERMAINAN “GALAXIO”

Kelas Mahasiswa K03

Dosen: Ir. Rila Mandala, M.Eng., Ph.D.

Diajukan sebagai tugas besar Mata Kuliah IF2211 Strategi Algoritma pada
Semester II Tahun Akademik 2022/2023



Disusun Oleh:

Kelompok 29 (JAS-029)

Jason Rivalino	13521008
Muhamad Salman Hakim Alfarisi	13521010
Afnan Edsa Ramadhan	13521011

SEKOLAH TEKNIK ELEKTRO DAN INFORMATIKA
INSTITUT TEKNOLOGI BANDUNG
BANDUNG

2022

DAFTAR ISI

DAFTAR ISI	3
BAB I: DESKRIPSI TUGAS	5
BAB II: LANDASAN TEORI	8
I. Teori Dasar Algoritma Greedy	8
II. Cara Kerja Program	9
BAB III: APLIKASI STRATEGI ALGORITMA GREEDY	13
I. Elemen Algoritma Greedy dalam Persoalan Game Galaxio	13
1) Algoritma Greedy dalam permasalahan pengecekan ukuran Bot	13
2) Algoritma Greedy dalam permasalahan mencari makanan Super Food	13
3) Algoritma Greedy dalam permasalahan menembak peluru torpedo	14
4) Algoritma Greedy dalam permasalahan mengaktifkan shield pelindung	15
5) Algoritma Greedy dalam permasalahan menghindari obstacle	15
6) Algoritma Greedy dalam permasalahan teleportasi	16
7) Algoritma Greedy dalam permasalahan ketika world dalam game mengecil	16
II. Eksplorasi Alternatif Solusi Algoritma Greedy pada Bot Permainan Galaxio	17
1) Strategi Algoritma Greedy memfokuskan untuk mencari makan	17
2) Strategi Algoritma Greedy untuk menyerang musuh	18
3) Strategi Algoritma Greedy untuk bertahan dan menghindar	19
III. Analisis Efisiensi dan Efektivitas dari Kumpulan Solusi Algoritma Greedy	20
1) Efisiensi dan efektivitas strategi Algoritma Greedy yang memfokuskan untuk mencari makan	20
2) Efisiensi dan efektivitas strategi Algoritma Greedy untuk menyerang musuh	21
3) Efisiensi dan efektivitas strategi Algoritma Greedy untuk bertahan dan menghindar	21

IV. Strategi Greedy yang Digunakan pada Program Bot	22
BAB IV: IMPLEMENTASI DAN PENGUJIAN	25
I. Implementasi Algoritma Greedy pada Bot Permainan Galaxio	25
II. Penjelasan Struktur Data pada Bot Permainan Galaxio	28
III. Pengujian dan Analisis Bot pada Permainan Galaxio	30
BAB V: KESIMPULAN DAN SARAN	33
I. Kesimpulan	33
II. Saran	33
DAFTAR PUSTAKA	34
LAMPIRAN	35

BAB I

DESKRIPSI TUGAS

Galaxio adalah sebuah *game battle royale* yang mempertandingkan bot kapal anda dengan beberapa bot kapal yang lain. Setiap pemain akan memiliki sebuah bot kapal dan tujuan dari permainan adalah agar bot kapal anda yang tetap hidup hingga akhir permainan. Penjelasan lebih lanjut mengenai aturan permainan akan dijelaskan di bawah. Agar dapat memenangkan pertandingan, setiap bot harus mengimplementasikan strategi tertentu untuk dapat memenangkan permainan.

Pada tugas besar pertama Strategi Algoritma ini, tugas yang diberikan yaitu menggunakan *game engine* untuk mengimplementasikan permainan Galaxio. Implementasi dilakukan pada bot kapal dalam permainan dengan menggunakan Algoritma Greedy untuk memenangkan permainan.

Adapun spesifikasi permainan yang digunakan pada tugas besar ini disesuaikan dengan spesifikasi yang disediakan oleh *game engine Galaxio* pada *repository* Entelect Challenge. Beberapa aturan umumnya antara lain sebagai berikut.

1. Peta permainan berbentuk kartesius yang memiliki arah positif dan negatif. Peta hanya menangani angka bulat. Kapal hanya bisa berada di *integer* x,y yang ada di peta. Pusat peta adalah 0,0 dan ujung dari peta merupakan radius. Jumlah ronde maximum pada game sama dengan ukuran radius. Pada peta, akan terdapat 5 objek, yaitu *Players*, *Food*, *Wormholes*, *Gas Clouds*, *Asteroid Fields*. Ukuran peta akan mengecil seiring batasan peta mengecil.
2. Kecepatan kapal dilambangkan dengan x . Kecepatan kapal akan dimulai dengan kecepatan 20 dan berkurang setiap ukuran kapal bertambah. Ukuran (radius) kapal akan dimulai dengan ukuran 10. *Heading* dari kapal dapat bergerak antar 0 hingga 359 derajat. Efek *afterburner* akan meningkatkan kecepatan kapal dengan faktor 2, tetapi mengecilkan ukuran kapal sebanyak 1 setiap tick. Kemudian kapal akan menerima 1 salvo charge setiap 10 tick. Setiap kapal hanya dapat menampung 5 salvo charge. Penembakan salvo torpedo (ukuran 10) mengurangi ukuran kapal sebanyak 5.

IF2211

Strategi Algoritma

3. Setiap objek pada lintasan punya koordinat x,y dan radius yang mendefinisikan ukuran dan bentuknya. *Food* akan disebar pada peta dengan ukuran 3 dan dapat dikonsumsi oleh kapal *player*. Apabila *player* mengkonsumsi *Food*, maka *Player* akan bertambah ukuran yang sama dengan *Food*. *Food* memiliki peluang untuk berubah menjadi *Super Food*. Apabila *Super Food* dikonsumsi maka setiap makan *Food*, efeknya akan 2 kali dari *Food* yang dikonsumsi. Efek dari *Super Food* bertahan selama 5 tick.
4. *Wormhole* ada secara berpasangan dan memperbolehkan kapal dari *player* untuk memasukinya dan keluar di pasangan satu lagi. *Wormhole* akan bertambah besar setiap tick game hingga ukuran maximum. Ketika *Wormhole* dilewati, maka *wormhole* akan mengecil sebanyak setengah dari ukuran kapal yang melewatinya dengan syarat *wormhole* lebih besar dari kapal *player*.
5. *Gas Clouds* akan tersebar pada peta. Kapal dapat melewati *gas cloud*. Setiap kapal bertabrakan dengan *gas cloud*, ukuran dari kapal akan mengecil 1 setiap tick game. Saat kapal tidak lagi bertabrakan dengan *gas cloud*, maka efek pengurangan akan hilang.
6. *Torpedo Salvo* akan muncul pada peta yang berasal dari kapal lain. *Torpedo Salvo* berjalan dalam lintasan lurus dan dapat menghancurkan semua objek yang berada pada lintasannya. *Torpedo Salvo* dapat mengurangi ukuran kapal yang ditabraknya. *Torpedo Salvo* akan mengecil apabila bertabrakan dengan objek lain sebanyak ukuran yang dimiliki dari objek yang ditabraknya.
7. *Supernova* merupakan senjata yang hanya muncul satu kali pada permainan di antara quarter pertama dan quarter terakhir. Senjata ini tidak akan bertabrakan dengan objek lain pada lintasannya. *Player* yang menembaknya dapat meledakannya dan memberi *damage* ke *player* yang berada dalam zona. Area ledakan akan berubah menjadi *gas cloud*.
8. *Player* dapat meluncurkan *teleporter* pada suatu arah di peta. *Teleporter* tersebut bergerak dalam direksi dengan kecepatan 20 dan tidak bertabrakan dengan objek apapun. *Player* tersebut dapat berpindah ke tempat *teleporter* tersebut. Harga setiap peluncuran *teleporter* adalah 20. Setiap 100 tick *player* akan mendapatkan 1 *teleporter* dengan jumlah maximum adalah 10.
9. Ketika kapal *player* bertabrakan dengan kapal lain, maka kapal yang lebih besar akan mengonsumsi oleh kapal yang lebih kecil sebanyak 50% dari ukuran kapal yang lebih

IF2211

Strategi Algoritma

besar hingga ukuran maximum dari ukuran kapal yang lebih kecil. Hasil dari tabrakan akan mengarahkan kedua dari kapal tersebut lawan arah.

10. Terdapat beberapa *command* yang dapat dilakukan oleh *player*. Setiap tick, *player* hanya dapat memberikan satu *command*.
11. Setiap player akan memiliki score yang hanya dapat dilihat jika permainan berakhir. Score ini digunakan saat kasus *tie breaking* (semua kapal mati). Jika mengonsumsi kapal *player* lain, maka score bertambah 10, jika mengonsumsi *food* atau melewati wormhole, maka score bertambah 1. Pemenang permainan adalah kapal yang bertahan paling terakhir dan apabila *tie breaker* maka pemenang adalah kapal dengan score tertinggi.

BAB II

LANDASAN TEORI

I. Teori Dasar Algoritma Greedy

Algoritma Greedy adalah algoritma yang memecahkan persoalan secara langkah per langkah. Algoritma ini merupakan algoritma yang paling populer untuk memecahkan persoalan pencarian solusi optimal. Sesuai namanya yaitu *greedy*, yang berarti rakus, tamak, atau serakah, algoritma ini memiliki prinsip “*take what you can get now*”.

Dalam Algoritma Greedy, setiap langkah yang ada akan mengambil pilihan yang terbaik tanpa memperhatikan konsekuensi kedepannya (memilih optimum lokal) dengan harapan langkah sisanya akan berakhir menuju optimum global.

Beberapa elemen yang terdapat pada Algoritma Greedy antara lain:

- 1) Himpunan kandidat (C): himpunan ini berisi kandidat elemen pembentuk solusi yang akan dipilih pada setiap langkah (Contohnya: simpul/sisi dalam graf, koin, dll).
- 2) Himpunan solusi (S): himpunan ini berisi kandidat yang sudah terpilih sebagai solusi persoalan.
- 3) Fungsi solusi: fungsi yang menentukan apakah kandidat yang dipilih sudah memberikan solusi. Fungsi ini akan mengembalikan nilai boolean yaitu true jika memberikan solusi lengkap dan false jika memberikan solusi belum lengkap.
- 4) Fungsi seleksi (*selection function*): fungsi yang memilih kandidat yang paling mungkin untuk mencapai solusi berdasarkan strategi *greedy* tertentu yang bersifat heuristik.
- 5) Fungsi kelayakan: fungsi yang memeriksa kelayakan kandidat yang terpilih apakah dapat dimasukkan ke dalam himpunan solusi atau tidak.
- 6) Fungsi objektif: fungsi yang mengoptimalkan solusi (memaksimumkan atau meminimumkan).

Optimum global yang dihasilkan dari Algoritma Greedy tidak selalu merupakan solusi optimum terbaik (bisa merupakan solusi *sub-optimum* atau *pseudo-optimum*). Hal ini disebabkan karena Algoritma Greedy tidak beroperasi secara menyeluruh terhadap semua kemungkinan solusi dan terdapat beberapa fungsi seleksi yang berbeda. Namun,

IF2211

Strategi Algoritma

jika solusi terbaik mutlak tidak terlalu diperlukan, Algoritma Greedy dapat digunakan untuk menghasilkan hampiran solusi optimal.

Beberapa contoh dari penerapan Algoritma Greedy adalah sebagai berikut:

- 1) Persoalan penukaran uang
- 2) Persoalan memilih aktivitas
- 3) Minimasi waktu dalam sistem
- 4) Persoalan *knapsack*
- 5) Penjadwalan Job dengan tenggat waktu
- 6) Pohon merentang minimum
- 7) Lintasan terpendek
- 8) Kode Huffman
- 9) Pecahan Mesir

II. Cara Kerja Program

Pada program yang disediakan di dalam *Game Engine Galaxio* (Starter-Pack Galaxio), program terbagi menjadi beberapa folder dan file yang masing-masing foldernya memiliki fungsi masing-masing yaitu:

- 1) Folder engine-publish: folder ini berfungsi untuk mengimplementasikan *logic* dan aturan yang ada dalam permainan.
- 2) Folder logger-publish: folder ini berfungsi untuk mencatat dan menyimpan *log* permainan untuk mengetahui hasil permainan. File *log* yang terdapat pada folder ini juga nantinya akan digunakan dalam visualizer.
- 3) Folder reference-bot-publish: folder ini menyediakan *default* bot yang berasal dari developer game yang bisa digunakan untuk mengetes kemampuan bot yang sudah kelompok kami buat. Bot yang kelompok kami buat akan dites dengan diadu melawan bot yang ada pada folder ini.
- 4) Folder runner-publish: folder ini berfungsi untuk menggelar sebuah pertandingan. Folder ini menghubungkan bot dengan engine.
- 5) Folder starter-bots: folder ini merupakan folder yang paling penting karena proses implementasi pembuatan bot kapal akan dilakukan dalam folder ini. Dalam folder

IF2211

Strategi Algoritma

ini, kelompok kami menyusun program bot kapal dengan menggunakan strategi Algoritma Greedy.

- 6) Folder visualiser: folder ini berfungsi untuk menjalankan aplikasi Galaxio. Dalam folder ini, terdapat aplikasi Galaxio untuk melihat visualisasi dari permainan yang sudah dijalankan pada run.bat sebelumnya.
- 7) File run.bat: file ini berfungsi untuk membantu menjalankan program dengan menjalankan game runner, game engine, game logger, dan bots secara bersama-sama.

Proses pengembangan bot dari kelompok kami dilakukan pada folder starter-bots. Dalam folder ini, terdapat beberapa bahasa pemrograman yang bisa dipilih untuk mengembangkan bot. Namun, dalam tugas ini, bahasa yang digunakan adalah bahasa pemrograman Java sehingga folder yang dipilih adalah folder Javabot. Beberapa komponen yang terdapat pada folder Javabot antara lain:

- 1) Folder source code: folder ini berisi source code untuk membangun bot dari kelompok kami. Pada folder ini juga, terdapat sub-folder dengan komponen sebagai berikut:
 - a) Enums: dalam folder ini, terdapat fungsi untuk menyimpan objek dalam game (GameObjects) dan fungsi untuk aksi dari bot player (PlayerActions)
 - b) Models: dalam folder ini, terdapat beberapa konstruktor untuk membangun program, seperti GameObject, GameState, GameStateDto, PlayerAction, Position, dan World. Folder ini berfungsi untuk melakukan pengecekan data kondisi pemain.
 - c) Services: dalam folder ini, terdapat file BotService.java. Dalam file inilah, tepatnya pada fungsi computeNextPlayerAction, kelompok kami akan mengimplementasikan bot kapal dengan menggunakan strategi Algoritma Greedy. Implementasi Algoritma Greedy pada fungsi ini dilakukan dengan memanggil fungsi GameObjects dan PlayerActions pada folder Enums untuk membangun strategi serta fungsi-fungsi lainnya yang terdapat pada file BotService.java. Aksi dari bot dapat berubah-ubah berdasarkan implementasi Algoritma yang ada pada fungsi ini.

IF2211

Strategi Algoritma

- d) File `main.java`: file main dari program, berfungsi untuk menyatukan program-program dari berbagai folder sebelumnya. Pada bagian main ini, tepatnya pada baris ke 61, terdapat bagian untuk mengubah nama bot *default* yang disediakan menjadi nama bot yang akan digunakan untuk bertanding.
- 2) Folder `target`: folder ini berfungsi untuk mengubah source code yang sudah dibuat menjadi file `jar` yang dapat digunakan untuk bertanding. Untuk membuat folder ini, kelompok kami menggunakan maven dengan menjalankan command “`mvn clean package`” pada terminal.
- 3) File `dockerfile` dan `pom.xml`

Alur dari proses menjalankan *Game Engine* Galaxio adalah sebagai berikut:

- 1) Setelah melakukan pengeditan program `BotService.java`, buka terminal pada folder `Javabot`.
- 2) Di dalam terminal, ketikkan “`mvn clean package`” (Note: pastikan sudah mendownload maven terlebih dahulu). Tunggu hingga proses selesai.
- 3) Setelah proses selesai, akan terbentuk file `JavaBot.jar` pada folder `target`. File inilah yang merupakan file bot kelompok kami yang akan digunakan untuk bertanding.
- 4) Copy path file `JavaBot.jar`.
- 5) Ubah script pada `run.bat` dengan mengganti reference bot menjadi bot yang sudah dibuat.
- 6) Jalankan `run.bat` untuk melakukan run pada game runner, game engine, dan game logger untuk mendapatkan log permainan, hasil dari log bisa dicek pada `..\starter-pack\logger-publish`, jika run berhasil, akan terbentuk dua file `GameStateLog.json` baru.
- 7) Copy address dari lokasi file `GameStateLog.json`
- 8) Jalankan program Galaxio yang ada pada folder `visualizer`
- 9) Pada menu Game Galaxio, pilih menu options dan paste copy address lokasi file pada step nomor 7 di log files location
- 10) Untuk melihat permainan, pilih menu load dan pilih data `GameStateLog.json` yang telah terbentuk sebelumnya
- 11) Klik start untuk melihat hasil permainan

IF2211
Strategi Algoritma

- 12) Jika ingin bermain lagi, bisa menjalankan file run.bat kembali dan membuka program Galaxio serta melakukan load file seperti step sebelumnya. Jika ingin melakukan perubahan source code pada program yang terdapat di folder src JavaBot, lakukan kembali step yang pertama dan kedua untuk mendapatkan file jar dengan program yang sudah diperbaharui.

BAB III

APLIKASI STRATEGI ALGORITMA GREEDY

I. Elemen Algoritma Greedy dalam Persoalan Game Galaxio

Elemen Algoritma Greedy yang diimplementasikan oleh kelompok kami dalam Bot Game Galaxio antara lain:

1) Algoritma Greedy dalam permasalahan pengecekan ukuran Bot

Nama Elemen	Definisi Elemen
Himpunan Kandidat (C)	Semua kondisi ukuran bot pada suatu <i>tick</i> tertentu yang dicek dengan fungsi <code>getSize()</code>
Himpunan Solusi (S)	Kemungkinan kondisi apakah kondisi ukuran bot memenuhi ukuran tertentu atau tidak.
Fungsi Solusi	Melakukan pengecekan apakah bot sudah mencapai ukuran tertentu atau belum.
Fungsi Seleksi	Memilih <i>command</i> berdasarkan kondisi ukuran bot pada saat <i>tick</i> tertentu.
Fungsi Kelayakan	Memeriksa kelayakan apakah <i>command</i> yang terdapat pada fungsi ini sudah valid atau tidak.
Fungsi Objektif	Menjalankan kondisi <i>command</i> untuk lebih fokus mencari makanan dan bertahan saat kondisi ukuran bot kurang dari 100. Jika sudah lebih, bot akan lebih fokus untuk menyerang.

2) Algoritma Greedy dalam permasalahan mencari makanan *Super Food*

Nama Elemen	Definisi Elemen
Himpunan Kandidat (C)	Semua kondisi yang ada dari <i>command</i> SUPERFOOD dan juga <i>command</i> FOOD atau tidak keduanya.
Himpunan Solusi (S)	Kemungkinan solusi melakukan proses pencarian dengan priority <i>Super Food</i> terlebih dahulu baru

IF2211
Strategi Algoritma

	mencari Food untuk menambah ukuran bot.
Fungsi Solusi	Melakukan pengecekan apakah sudah memakan <i>Super Food</i> atau belum.
Fungsi Seleksi	Memilih <i>command</i> berdasarkan kondisi pengecekan apakah sudah memakan <i>Super Food</i> atau belum.
Fungsi Kelayakan	Memeriksa kelayakan apakah <i>command</i> yang terdapat pada fungsi ini sudah valid atau tidak.
Fungsi Objektif	Menjalankan kondisi <i>command</i> untuk mencari <i>Super Food</i> terlebih dahulu. Jika kondisi memakan <i>Super Food</i> sudah terpenuhi, dilanjutkan dengan memakan Food biasa untuk menambah ukuran Bot.

3) Algoritma Greedy dalam permasalahan menembak peluru torpedo

Nama Elemen	Definisi Elemen
Himpunan Kandidat (C)	Semua kondisi yang ada dari <i>command</i> FIRETORPEDOES atau tidak memenuhi kondisi.
Himpunan Solusi (S)	Kemungkinan solusi melakukan penembakan torpedo berdasarkan kondisi posisi dan perbandingan ukuran bot.
Fungsi Solusi	Melakukan pengecekan kondisi jarak musuh, ukuran bot player, dan juga perbandingan ukuran bot player dengan bot musuh.
Fungsi Seleksi	Memilih <i>command</i> berdasarkan kondisi pengecekan apakah kondisi posisi musuh dan perbandingan ukuran bot sudah terpenuhi.
Fungsi Kelayakan	Memeriksa kelayakan apakah <i>command</i> yang terdapat pada fungsi ini sudah valid atau tidak.
Fungsi Objektif	Menjalankan kondisi <i>command</i> FIRETORPEDOES untuk menembakkan torpedo kepada bot musuh dan mengurangi ukurannya.

4) Algoritma Greedy dalam permasalahan mengaktifkan *shield* pelindung

Nama Elemen	Definisi Elemen
Himpunan Kandidat (C)	Semua kondisi yang ada dari <i>command</i> ACTIVATESHIELD atau tidak memenuhi kondisi.
Himpunan Solusi (S)	Kemungkinan solusi melakukan aktivasi <i>shield</i> pelindung berdasarkan kondisi dari torpedo lawan.
Fungsi Solusi	Melakukan pengecekan kondisi apakah peluru torpedo musuh tersedia dan juga jarak peluru torpedo sudah mencapai kondisi tertentu dan ukuran bot player.
Fungsi Seleksi	Memilih <i>command</i> berdasarkan kondisi pengecekan jarak peluru torpedo, ketersediaan peluru torpedo musuh, dan juga ukuran bot player.
Fungsi Kelayakan	Memeriksa kelayakan apakah <i>command</i> yang terdapat pada fungsi ini sudah valid atau tidak.
Fungsi Objektif	Menjalankan <i>command</i> untuk mengaktifkan <i>shield</i> pelindung melindungi bot dari peluru lawan.

5) Algoritma Greedy dalam permasalahan menghindari *obstacle*

Nama Elemen	Definisi Elemen
Himpunan Kandidat (C)	Semua kondisi yang ada dari variabel nearestObstacle() atau tidak memenuhi kondisi.
Himpunan Solusi (S)	Kemungkinan solusi menghindari <i>obstacle</i> dan mencari <i>Super Food</i> terdekat berdasarkan kondisi posisi bot player.
Fungsi Solusi	Melakukan pengecekan kondisi jarak obstacle dengan bot player.
Fungsi Seleksi	Menjalankan fungsi berdasarkan kondisi pengecekan jarak.
Fungsi Kelayakan	Memeriksa kelayakan apakah <i>command</i> yang terdapat pada fungsi ini sudah valid atau tidak.

Fungsi Objektif	Menjalankan fungsi untuk menghindari <i>obstacle</i> seperti GasCloud dan AsteroidFields
-----------------	--

6) Algoritma Greedy dalam permasalahan teleportasi

Nama Elemen	Definisi Elemen
Himpunan Kandidat (C)	Semua kondisi yang ada dari <i>command</i> FIRETELEPORT dan juga TELEPORT atau tidak keduanya.
Himpunan Solusi (S)	Kemungkinan solusi melakukan teleportasi berdasarkan perbandingan ukuran bot dan juga jarak musuh.
Fungsi Solusi	Melakukan pengecekan kondisi ukuran bot dan juga jarak antara bot player dan juga musuh.
Fungsi Seleksi	Memilih <i>command</i> berdasarkan kondisi pengecekan apakah kondisi posisi musuh dan perbandingan ukuran bot sudah terpenuhi.
Fungsi Kelayakan	Memeriksa kelayakan apakah <i>command</i> yang terdapat pada fungsi ini sudah valid atau tidak.
Fungsi Objektif	Menjalankan <i>command</i> FIRETELEPORT dan juga TELEPORT untuk melakukan proses perpindahan dari satu tempat ke tempat lain.

7) Algoritma Greedy dalam permasalahan ketika *world* dalam game mengecil

Nama Elemen	Definisi Elemen
Himpunan Kandidat (C)	Semua kondisi ukuran world pada suatu <i>tick</i> tertentu yang dicek dengan fungsi getRadius()
Himpunan Solusi (S)	Kemungkinan kondisi apakah jarak world dengan bot player sudah mencapai jarak tertentu atau tidak.
Fungsi Solusi	Melakukan pengecekan kondisi antara jarak bot dengan center world lebih besar daripada jarak bot dengan ukuran world.
Fungsi Seleksi	Menjalankan fungsi berdasarkan kondisi pengecekan

	jarak.
Fungsi Kelayakan	Memeriksa kelayakan apakah <i>command</i> yang terdapat pada fungsi ini sudah valid atau tidak.
Fungsi Objektif	Menjalankan fungsi untuk menjauhi radius world dan mengarah ke center world.

II. Eksplorasi Alternatif Solusi Algoritma Greedy pada Bot Permainan Galaxio

Dalam game Galaxio ini, sebenarnya terdapat banyak sekali implementasi dari Algoritma Greedy. Terdapat banyak sekali *command* yang ada dalam permainan Galaxio yang bisa diimplementasikan dengan Algoritma Greedy untuk membentuk strategi agar dapat memenangkan pertandingan. Beberapa alternatif strategi Algoritma Greedy yang dapat dipergunakan dalam permainan ini antara lain:

1) Strategi Algoritma Greedy memfokuskan untuk mencari makan

Strategi Algoritma pada kondisi ini yaitu strategi untuk membuat bot player menjadi lebih fokus untuk mengincar makanan untuk dapat memperbesar ukuran bot. Setelah ukuran bot player menjadi cukup besar, bot yang ada akan menjadi lebih mudah untuk memangsa bot dari lawan-lawannya. Untuk prioritas dari strategi algoritma ini sendiri dapat dibagi menjadi beberapa kondisi sebagai berikut:

- Bot akan berfokus untuk mencari makanan (food atau *Super Food*) yang terdekat dengan posisi kapalnya pada saat ini. Pada strategi ini, bot akan melakukan pengecekan jarak makanan apapun. Setelah didapat jarak terdekat, bot akan menuju makanan tersebut dan langsung memakannya. Strategi ini tidak membedakan kondisi antara food dan juga *Super Food*. Semua akan dimakan dalam strategi ini asalkan jaraknya terdekat.
- Bot akan berfokus untuk mencari *Super Food*. Dalam strategi ini, bot akan berfokus hanya kepada *Super Food* saja. Strategi ini akan melakukan pengecekan jarak antara bot dengan *Super Food* terdekat. Jika sudah didapat, bot akan langsung bergerak menuju *Super Food* tersebut. Bot hanya akan memakan food yang seadanya didapat pada saat perjalanan dari posisi awal menuju *Super Food* terdekat.

- c) Bot akan berfokus untuk mencari *Super Food*, jika sudah mendapatkan *Super Food*, bot akan berfokus untuk mendapatkan food. Strategi ini berdasarkan pada peraturan permainan yaitu bot akan berfokus untuk memakan *Super Food* sebagai prioritasnya. Jika sudah mendapatkan *Super Food*, bot akan mencari food sebanyak-banyaknya (selama dalam efek *Super Food*, jika memakan food efeknya akan menjadi dua kali lipat). Bot akan kembali memprioritaskan *Super Food* ketika efek dari *Super Food* yang dimakan sebelumnya sudah habis.

2) Strategi Algoritma Greedy untuk menyerang musuh

Strategi Algoritma pada kondisi ini yaitu strategi untuk membuat bot player menjadi lebih fokus untuk menyerang musuh. Strategi ini cocok digunakan ketika bot player sudah memiliki ukuran yang cukup besar. Tujuan dari strategi ini adalah untuk memperkecil atau bahkan melenyapkan bot musuh. Beberapa kondisi yang bisa terjadi dalam kondisi strategi algoritma ini adalah sebagai berikut:

- a) Menyerang musuh secara langsung. Implementasi dari strategi ini yaitu dengan melakukan perbandingan ukuran antara bot player dengan bot musuh yang terdekat. Jika sudah memenuhi selisih jarak tertentu, bot player akan bergerak menuju bot musuh untuk memakannya. Penggunaan *command* AFTERBURNER dapat membantu strategi ini karena bisa membuat kapal bergerak lebih cepat.
- b) Menyerang musuh dengan Torpedo Salvo. Implementasi dari strategi ini yaitu dengan melakukan pengecekan jarak dan ukuran bot untuk mendapatkan kondisi yang optimal untuk melakukan penembakan pada bot kapal musuh. Pengecekan jarak dilakukan agar akurasi lebih baik dan pengecekan ukuran dilakukan agar bot tidak terus mengecil hingga musnah akibat melakukan penembakan karena setiap menembak, ukuran bot kapal akan menyusut sebanyak 5.
- c) Menyerang musuh dengan Teleporter. Implementasi dari strategi ini yaitu dengan mengecek posisi dan ukuran bot yang optimal untuk melakukan penembakan peluru teleport dan langsung berpindah. Pengecekan posisi dilakukan agar akurasi peluru teleport lebih baik dan pengecekan ukuran dilakukan agar saat bot player berpindah dan ingin memakan musuh, bot player sudah dipastikan memiliki ukuran yang lebih besar daripada musuh sehingga saat berpindah, bot player yang akan memakan musuh dan bukan sebaliknya.

3) Strategi Algoritma Greedy untuk bertahan dan menghindar

Strategi Algoritma pada kondisi ini yaitu strategi untuk membuat bot player menjadi bermain lebih *defensive* dengan tujuan untuk menghindar dari serangan musuh, obstacle, ataupun world radius. Beberapa kondisi yang dapat terjadi dalam kondisi strategi algoritma ini adalah sebagai berikut:

- a) Bertahan dari musuh dengan melarikan diri dari musuh yang mengejar. Dalam strategi ini, bot player akan berusaha untuk melarikan diri dari musuh yang mengejar. Jika ukuran bot lebih kecil daripada ukuran musuh, bot akan bergerak lebih cepat sehingga bisa melarikan diri dari musuh. Penggunaan *command* AFTERBURNER dapat membantu strategi ini karena bisa membuat kapal bergerak lebih cepat. Kelebihan dari strategi ini adalah tidak terjadi penyusutan saat proses melarikan diri. Kelemahan strategi ini yaitu jika ada musuh yang menggunakan teleport, bot player akan bisa langsung dimakan dengan teleportasi.
- b) Bertahan dari musuh dengan melarikan diri ke arah Wormhole. Dalam strategi ini, bot player akan mengecek jarak dan perbandingan ukuran bot player dengan bot musuh. Selain itu, akan dicek juga jarak bot kapal dengan Wormhole terdekat. Jika wormhole berjarak dekat, bot kapal akan melarikan diri ke Wormhole dan berpindah ke tempat keluar Wormhole lain. Kelebihan dari strategi ini adalah bot player dapat berpindah dengan sangat cepat menuju lokasi yang jauh dari musuh. Kelemahan dari strategi ini adalah ketika di Wormhole keluar terdapat musuh lain yang lebih besar, bot player tidak dapat melarikan diri dan akan langsung termakan.
- c) Bertahan dari musuh dengan melarikan diri menggunakan teleporter. Dalam strategi ini, bot player akan menggunakan *command* teleporter untuk melarikan diri dari musuh yang sedang menyerang. Bot player akan melakukan pengecekan terhadap kondisi optimal agar bot dapat melarikan diri dari musuh terdekat. Kelebihan dari strategi ini adalah bot dapat langsung melarikan diri dengan jarak yang cukup jauh. Kelemahan dari strategi ini adalah penggunaan teleporter akan mengurangi ukuran bot player sebanyak 20. Selain itu, bisa saja setelah teleport terdapat musuh lain ditempat hasil perpindahannya.

- d) Bertahan dari musuh dengan menggunakan *Shield* pelindung. Dalam strategi ini, bot player akan mengaktifkan *Shield* pelindung apabila terkena serangan dari musuh. Bot player akan melakukan pengecekan terhadap ukuran bot dan juga jarak serangan yang diberikan dari musuh. Kelebihan dari strategi ini adalah strategi ini cocok digunakan untuk menangkal serangan Torpedo Salvo dari lawan. Kekurangan dari strategi ini adalah pengaktifan *Shield* akan mengurangi ukuran bot player sebanyak 20. Selain itu, *Shield* tidak mempan untuk menghindari dimakan musuh berdasarkan kondisi perbedaan ukuran. Musuh bisa saja menggunakan *command* teleport untuk menyerang bot player dengan *Shield* dan memakannya.
- e) Menghindari obstacle seperti GasClouds, WormHole dan AsteroidFields. Implementasi dari strategi ini yaitu dengan menghitung jarak obstacle dengan bot player. Jika jarak antara obstacle dan bot player sudah mencapai ukuran tertentu, bot player akan berbalik dan mencari makan pada daerah lain yang tidak terdapat obstacle.
- f) Menghindari world radius yang terus mengecil setiap penambahan tick. Implementasi dari strategi ini yaitu dengan membandingkan jarak antara center world dan bot player dengan jarak antara world radius dan bot player. Jika jarak radius world dan player sudah sangat dekat, bot player akan berbalik arah menuju center world dan menghindari api radius.

III. Analisis Efisiensi dan Efektivitas dari Kumpulan Solusi Algoritma Greedy

Analisis Efisiensi dan Efektivitas yang ada pada kumpulan eksploratif solusi Algoritma Greedy adalah sebagai berikut:

1) Efisiensi dan efektivitas strategi Algoritma Greedy yang memfokuskan untuk mencari makan

Kondisi Strategi	Efisiensi	Efektivitas
Bot berfokus mencari makanan terdekat	Kompleksitas dari algoritma ini $O(1)$. Pencarian dilakukan langsung menuju	Efektivitas akan terjadi ketika bot berada dalam kondisi daerah yang terdapat banyak makanan di sekitarnya sehingga bot dapat

IF2211
Strategi Algoritma

	makanan terdekat.	membesar dengan sangat cepat.
Bot berfokus mencari <i>Super Food</i> terdekat	Kompleksitas dari algoritma ini $O(1)$, pencarian dilakukan langsung menuju <i>Super Food</i> terdekat.	Efektivitas akan terjadi ketika setelah bot memakan <i>Super Food</i> , terdapat banyak food dalam perjalanan menuju <i>Super Food</i> berikutnya.
Bot berfokus mencari <i>Super Food</i> , lalu mencari Food hingga efek <i>Super Food</i> habis	Kompleksitas dari algoritma ini $O(1)$, pencarian dilakukan langsung menuju <i>Super Food</i> terdekat tanpa ada perbandingan.	Efektivitas akan terjadi ketika disekitar <i>Super Food</i> yang dimakan, terdapat banyak food sehingga bot bisa langsung memakan dan bertumbuh besar.

2) Efisiensi dan efektivitas strategi Algoritma Greedy untuk menyerang musuh

Kondisi Strategi	Efisiensi	Efektivitas
Menyerang musuh secara langsung	Kompleksitas dari algoritma ini $O(1)$. Menyerang musuh secara langsung dengan kondisi terdekat.	Efektivitas akan terjadi saat bot musuh berada dalam jarak yang dekat dan tidak punya cukup ukuran untuk menembak peluru
Menyerang musuh dengan Torpedo Salvo	Kompleksitas dari algoritma ini $O(1)$. Menyerang musuh dengan jarak terdekat.	Efektivitas akan terjadi saat bot musuh lebih besar sehingga geraknya lebih lambat serta jaraknya dekat untuk menembak
Menyerang musuh dengan Teleporter	Kompleksitas dari algoritma ini $O(1)$. Menyerang musuh yang berarah yang sesuai dengan peluru teleporter.	Efektivitas akan terjadi saat musuh berukuran lebih kecil sehingga bisa langsung dimakan

3) Efisiensi dan efektivitas strategi Algoritma Greedy untuk bertahan dan menghindar

Kondisi Strategi	Efisiensi	Efektivitas
Bertahan dari musuh dengan	Kompleksitas dari algoritma ini $O(1)$.	Efektivitas akan terjadi saat bot berukuran tidak terlalu besar

IF2211
Strategi Algoritma

melarikan diri	Bertahan dengan memperbesar jarak bot dengan musuh.	sehingga bot dapat bergerak lebih cepat ditambah dengan <i>command</i> AFTERBURNER
Bertahan dari musuh dengan melarikan diri ke Wormhole	Kompleksitas dari algoritma ini $O(n)$. Karena wormhole obstacle, maka dihindari dengan mengecek makanan diluar obstacle	Efektivitas akan terjadi saat jarak bot dengan Wormhole sangat dekat
Bertahan dari musuh dengan melarikan diri menggunakan teleporter	Kompleksitas dari algoritma ini $O(1)$. Teleport kearah yang berlawanan dengan posisi musuh.	Efektivitas akan terjadi saat tembakan hasil teleporter mengarah 180 derajat membelakangi musuh
Bertahan dari musuh dengan menggunakan <i>Shield</i>	Kompleksitas dari algoritma ini $O(1)$. Langsung aktifkan shield berdasarkan jarak musuh terdekat.	Efektivitas akan terjadi saat musuh menembakkan Torpedo Salvo yang banyak hingga mengecilkan botnya. Bot player tidak akan terdampak efek dari Torpedo Salvo
Menghindari obstacle	Kompleksitas dari algoritma ini $O(n)$. Menghindari dengan mengecek makanan diluar obstacle	Efektivitas terjadi saat bot tidak sedang berada didekat obstacle sehingga tidak perlu proses menghindar.
Menghindari radius world	Kompleksitas dari algoritma ini $O(1)$. Menghindari dengan mengarahkan bot menuju tengah-tengah world	Efektivitas terjadi ketika bot tidak sedang berada dipinggir world sehingga tidak perlu proses menghindar.

IV. Strategi Greedy yang Digunakan pada Program Bot

Adapun strategi algoritma yang diimplementasikan dalam bot player kelompok kami adalah kami menentukan strategi mencari makan, menyerang, dan bertahan berdasarkan ukuran bot player. Jika ukuran bot masih kurang dari 100, strategi akan

IF2211

Strategi Algoritma

difokuskan untuk bertahan dan mencari makan. Jika sudah lebih dari 100, strategi akan lebih menyerang bot lawan.

1. Strategi saat ukuran bot kurang dari 100

- a) Untuk strategi mencari makan, bot kelompok kami menggunakan strategi memprioritaskan *Super Food* lalu memakan food hingga efek dari *Super Food* sudah habis. Alasan kami memilih strategi ini karena untuk mengoptimalkan proses makan karena kondisi food yang juga lebih banyak tersebar dalam peta. Strategi ini yang paling penting karena strategi ini dapat dipakai dalam semua ukuran bot, tidak seperti strategi lain yang memiliki keterbatasan berdasarkan ukuran dari bot.
- b) Untuk strategi bertahan, bot kelompok kami menggunakan *Shield* untuk melindungi diri dari serangan Torpedo Salvo musuh. Strategi ini dipilih karena serangan Torpedo Salvo biasanya memiliki jarak yang cukup dekat dengan bot sehingga lebih baik ditangkal dengan *Shield* daripada dihindari. Akan tetapi, jika ukuran bot sudah sangat kecil, terdapat kondisional untuk tidak mengaktifkan *Shield* agar bot tidak lenyap.
- c) Untuk strategi campuran, ada teleporter yang bisa digunakan untuk memakan musuh yang berukuran lebih kecil maupun bisa digunakan untuk menghindari musuh dengan menembakan peluru teleporter ke arah yang berlawanan dengan posisi bot musuh. Aktivasi teleporter memperhatikan ukuran bot sehingga teleport tidak dapat diaktifkan jika ukuran bot sangat kecil.
- d) Pada saat kondisi ini juga, terdapat sedikit strategi untuk menyerang dengan menggunakan Torpedo Salvo untuk menembaki lawan agar ukuran bot player menjadi membesar. Namun, kondisi menyerang ini juga memperhatikan ukuran bot. Apabila bot player sudah sangat kecil, maka strategi ini tidak bisa dijalankan.
- e) Kondisi menghindari obstacle dan radius world dipakai dalam kondisi ini untuk mengoptimalkan pertumbuhan bot.

2. Strategi saat ukuran bot lebih dari 100

- a) Untuk strategi menyerang, kelompok kami menggunakan Torpedo Salvo dan Teleporter untuk menyerang dan memakan lawan. Torpedo Salvo digunakan untuk menyusutkan ukuran lawan dan memperbesar ukuran bot player.

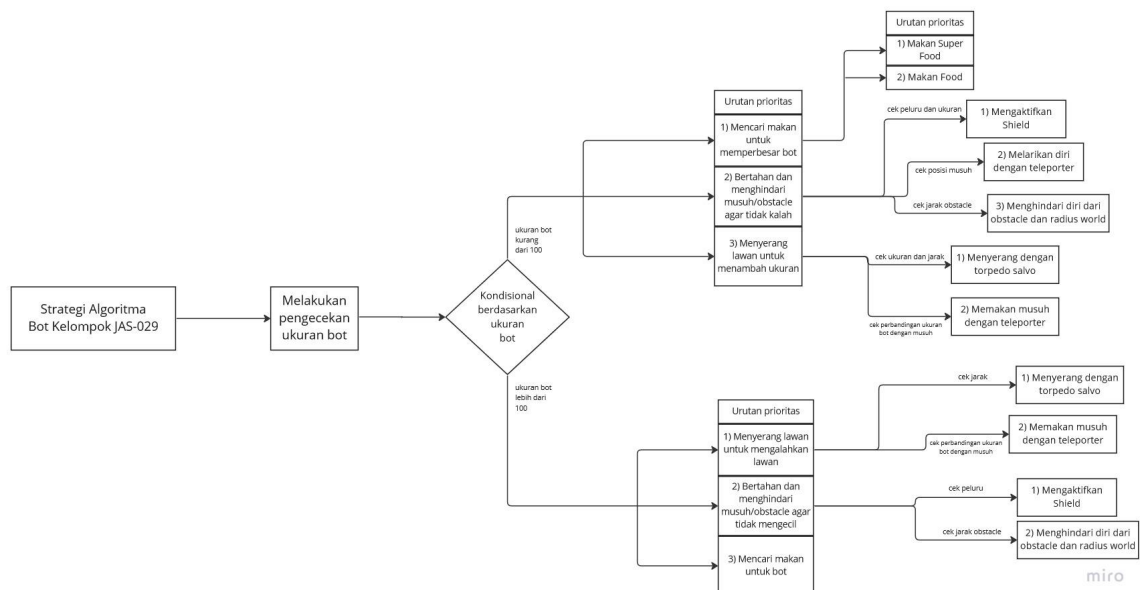
IF2211

Strategi Algoritma

Sedangkan, teleporter digunakan untuk berpindah ke tempat lawan dan langsung memakannya. Tentunya untuk teleporter diperlukan pengecekan perbandingan ukuran bot player dengan bot lawan agar bot player tidak berbalik dimakan oleh bot musuh.

- b) Pada saat kondisi ini juga, terdapat strategi untuk menghindari serangan Torpedo Salvo dari lawan dengan menggunakan *Shield* sehingga peluru torpedo tidak mengenai bot player.
- c) Kondisi menghindari radius world dipakai dalam kondisi ini.

Untuk gambaran diagram dari strategi kelompok kami adalah sebagai berikut:



BAB IV

IMPLEMENTASI DAN PENGUJIAN

I. Implementasi Algoritma Greedy pada Bot Permainan Galaxio

Pseudocode implementasi Algoritma Greedy pada bot yang kelompok kami buat adalah sebagai berikut:

```
Implementasi fungsi computeNextPlayerAction  
KAMUS LOKAL:  
bot : GameObject  
playerAction : PlayerAction  
gameState : GameState  
isTeleport, isMakanSuper, teleportAvailable : Boolean  
tik, tikTeleport : integer  
  
ALGORITMA FUNGSI COMPUTENEXTPLAYER:  
-> FORWARD  
if (not gameState.getGameObjects().isEmpty()) then  
    food <- gameState.getGameObjects(SUPERFOOD)  
    foodd <- gameState.getGameObjects(FOOD)  
    obstacle <- gameState.getGameObjects(GASCLOUD) or  
    gameState.getGameObjects(ASTEROIDFIELDS)  
    enemy <- gameState.getPlayerGameObjects(PPLAYER)  
    nearestFood <- getDistanceBetween(food)  
    nearestFoodd <- getDistanceBetween(foodd)  
    nearestObstacle <- getDistanceBetween(obstacle)  
    nearestEnemy <- getDistanceBetween(enemy)  
    listTorpedo <- gameState.getPlayerGameObjects(TORPEDOSALVO)  
  
    if (tikTeleport + 100 = gameState.getWorld().getCurrentTick()) then  
        teleportAvailable <- true  
  
    if (bot.getSize() < 100)  
    if(isMakanSuper) then  
        headingToFoodd <- getHeadingBetween(nearestFoodd)  
        playerAction.heading <- headingToFoodd  
        if(tik+5 = gameState.getWorld().getCurrentTick()) then  
            isMakanSuper <- false  
    else if (isTeleport) then  
        double jarak <- getRealDistance(bot, nearestEnemy)  
        double velo <- 20  
        int timee <- (int)jarak/(int)velo
```

IF2211

Strategi Algoritma

```
        if(tik+timee <= gameState.getWorld().getCurrentTick()) then
            playerAction.action <- PlayerActions.TELEPORT;
isTeleport <- false
        else
            headingToFood <- getHeadingBetween(nearestFood)
            playerAction.heading <- headingToFood
if(getRealDistance(bot, nearestFood) < 10) then
            isMakanSuper <- true
            tik <- gameState.getWorld().getCurrentTick()

if (getRealDistance(bot, nearestEnemy) < 100 and bot.getSize() > 25 and
nearestEnemy.getSize() > bot.getSize() and not isTeleport) then
            headingToEnemy <- getHeadingBetween(nearestEnemy)
playerAction.setHeading(headingToEnemy)
playerAction.action = PlayerActions.FIRETORPEDOES;
if(bot.size > 50 && teleportAvailable and (bot.getSize() - 30 >
nearestEnemy.getSize()) and getRealDistance(bot, nearestEnemy) > 100 and not
isTeleport) then
            headingToEnemy <- getHeadingBetween(nearestEnemy)
playerAction.setHeading(headingToEnemy)
playerAction.action <- PlayerActions.FIRETELEPORT
tik <- gameState.getWorld().getCurrentTick()
tikTeleport <- gameState.getWorld().getCurrentTick()
isTeleport <- true
teleportAvailable <- false
isMakanSuper <- false
        if (getRealDistance(bot, nearestObstacle) < 50) then
int i <- 0
while (getRealDistance(bot, foodd.get(i)) < 30) do
    i++
            playerAction.heading <- getHeadingBetween(foodd.get(i)),
if(distanceFromWorldCenter(bot) + (2.5 * bot.getSize()) >
gameState.getWorld().getRadius()) then
playerAction.heading <- (int)getDirectionPosition(bot,
gameState.getWorld().getCenterPoint())
if (listTorpedo.size() > 0) then
nearestTorpedo <- listTorpedo.stream().min(Comparator.comparing(gameObject ->
getDistanceBetween(gameObject, bot))).get();
            headingToTorpedo <- getHeadingBetween(nearestTorpedo);
            int headingTorpedoToMe <- nearestTorpedo.getCurrentHeading();
            if (getRealDistance(bot, nearestTorpedo) < 50 and ((headingToTorpedo >
(headingTorpedoToMe+10)%360 or headingToTorpedo < (headingTorpedoToMe-10)%360)) and
bot.getSize() > 80) then
                playerAction.action <- PlayerActions.ACTIVATESHIELD

else
```

IF2211

Strategi Algoritma

```
if(isTeleport)then
    double jarak <- getRealDistance(bot, nearestEnemy)
double velo <- 20
int timee <- (int)jarak/(int)velo
if(tik+timee <= gameState.getWorld().getCurrentTick())then
-> TELEPORT
isTeleport <- false
Else
if (getRealDistance(bot, nearestEnemy) < 250 and bot.getSize() > 25 and
nearestEnemy.getSize() > bot.getSize()) then
headingToEnemy <- getHeadingBetween(nearestEnemy)
playerAction.setHeading(headingToEnemy)
playerAction.action <- PlayerActions.FIRETORPEDOES
else if(teleportAvailable and (bot.getSize() - 30 > nearestEnemy.getSize()) and
getRealDistance(bot, nearestEnemy)>250) then
headingToEnemy <- getHeadingBetween(nearestEnemy)
playerAction.setHeading(headingToEnemy)
playerAction.action <- PlayerActions.FIRETELEPORT
tik <- gameState.getWorld().getCurrentTick()
tikTeleport <- gameState.getWorld().getCurrentTick()
isTeleport <- true
teleportAvailable <- false
else
headingToEnemy <- getHeadingBetween(nearestEnemy)
playerAction.setHeading(headingToEnemy)
if (listTorpedo.size() > 0) then
nearestTorpedo <- listTorpedo.stream().min(Comparator.comparing(gameObject ->
getDistanceBetween(gameObject, bot))).get()
int headingToTorpedo <- getHeadingBetween(nearestTorpedo)
int headingTorpedoToMe <- nearestTorpedo.getCurrentHeading()
if (getRealDistance(bot, nearestTorpedo) < 50 and (headingToTorpedo >
(headingTorpedoToMe+10)%360 or headingToTorpedo < (headingTorpedoToMe-10)%360)) then
playerAction.action <- PlayerActions.ACTIVATESHIELD
if(distanceFromWorldCenter(bot) + (2.5 * bot.getSize()) >
gameState.getWorld().getRadius()) then
playerAction.heading <- (int)getDirectionPosition(bot,
gameState.getWorld().getCenterPoint())

this.playerAction <- playerAction
```

II. Penjelasan Struktur Data pada Bot Permainan Galaxio

Dalam permainan Galaxio, terdapat beberapa struktur data yang penting untuk dipahami, yaitu *models* dan *enums*. *Models* adalah struktur data yang digunakan untuk merepresentasikan objek-objek dalam permainan, seperti *game object*, *player action*, *game state*, *game state DTO*, *position*, dan *world*. *Models* ini dapat digunakan untuk melakukan manipulasi data dalam permainan, seperti melakukan aksi selama permainan berlangsung.

Sementara itu, *enums* atau *enumeration* adalah struktur data yang digunakan untuk merepresentasikan kumpulan nilai yang mungkin untuk sebuah variabel. Dalam permainan Galaxio, enum digunakan untuk merepresentasikan jenis objek yang ada dalam permainan dan aksi apa saja yang dapat dilakukan.

a. Models

Nama Kelas	Deskripsi
Game Object	Kelas yang berisi semua data yang dimiliki oleh game object seperti : id, size speed, currentHeading, position, gameObjectType, torpedoSalvoCount
Player Action	Kelas yang berisi id, action, dan heading dari bot dan akan meng-assign aksi dari bot
Game State	Kelas yang akan menyimpan seluruh state dari permainan selama permainan berlangsung
Game State DTO	Kelas yang dirancang khusus untuk mengirim data game state antara klien dan server game
Position	Kelas yang berisi perintah-perintah untuk mengatur dan mendapatkan suatu posisi dari game object dalam bentuk kartesian (x,y)
World	Kelas dari arena yang digunakan selama permainan, menyimpan radius arena, waktu berjalannya permainan dan posisi dari pusat arena

IF2211
Strategi Algoritma

b. Enums

Nama Kelas	Deskripsi
ObjectTypes	Kelas yang berisi enumerasi dari tipe-tipe object yang ada di dalam permainan Galaxio yaitu: PLAYER, FOOD, WORMHOLE, GASCLOUD, ASTEROIDFIELD, TORPEDOSALVO, SUPERFOOD, SUPERNOVAPICKUP, SUPERNOVABOMB, TELEPORTER, SHIELDPICKUP.
PlayerActions	Kelas yang berisi enumerasi dari aksi-aksi yang bisa dilakukan oleh bot di dalam permainan yaitu: FORWARD, STOP,TARTAFTERBURNER, STOPAFTERBURNER, FIRETORPEDOES, FIRESUPERNOVA, DETONATESUPERNOVA, FIRETELEPORT, TELEPORT,ACTIVATESHIELD

c. Services

Atribut	Deskripsi
private GameObject bot	Game Object bertipe player yang akan dijalankan oleh program
private PlayerAction playerAction	Aksi yang dilakukan oleh bot selama permainan
private GameState gameState	State untuk menyimpan jalannya permainan

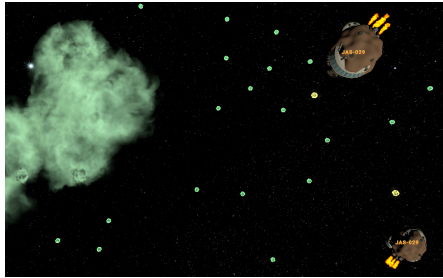
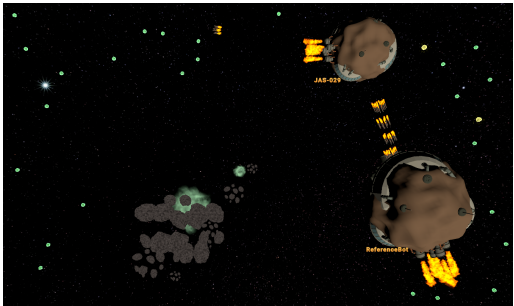
Method	Deskripsi
public void computeNextPlayerAction(PlayerAction playerAction)	Method untuk menjalankan algoritma dari bot yang sudah dibuat
private int getDirectionPosition(GameObject bot, Position position)	Method untuk mendapatkan arah suatu game game object dengan suatu posisi
private double distanceFromWorldCenter(GameObject object)	Method untuk menghitung jarak antara game object dengan pusat dari arena

IF2211
Strategi Algoritma

private double getDistanceBetween(GameObject object1, GameObject object2)	Method untuk menghitung jarak antara dua game object
private int getHeadingBetween(GameObject otherObject)	Method untuk mendapatkan arah dari antara bot dengan suatu game object
private double getRealDistance(GameObject object1, GameObject object2)	Method untuk menghitung jarak antara dua game object

III. Pengujian dan Analisis Bot pada Permainan Galaxio

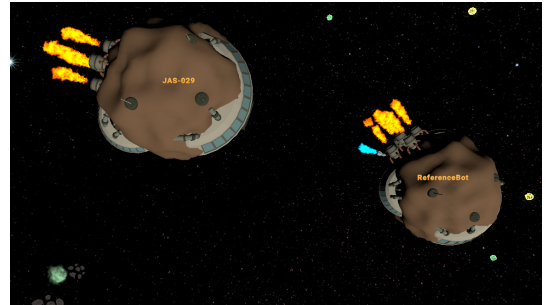
Pengujian dari bot yang sudah kelompok kami buat akan melawan bot *default* yang disediakan oleh Developer Entelect Challenge yaitu ReferenceBot. Pertandingan akan dilakukan secara dua melawan dua antara bot yang kelompok kami buat melawan ReferenceBot. Secara hasil pertandingan, presentase kemenangan bot kami sudah cukup tinggi dengan hanya mengalami satu kali kekalahan dalam lima pertandingan. Berikut merupakan visualisasi hasil pertandingan yang didapat dari hasil implementasi:

Kondisi Bot	Foto
Kondisi bot player priority mengincar <i>Super Food</i>	
Kondisi bot player menembakkan Torpedo Salvo pada bot musuh	

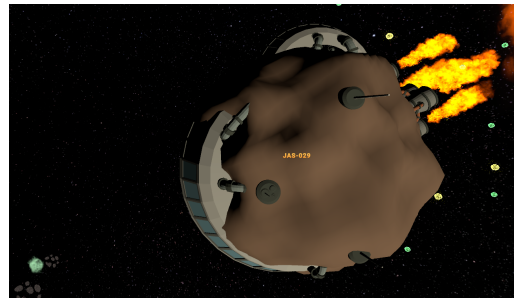
IF2211
Strategi Algoritma

Kondisi bot player menembakkan peluru teleporter dan langsung berpindah serta memakan musuh

Saat bot player menembakkan peluru teleport:



Setelah bot player melakukan teleportasi:

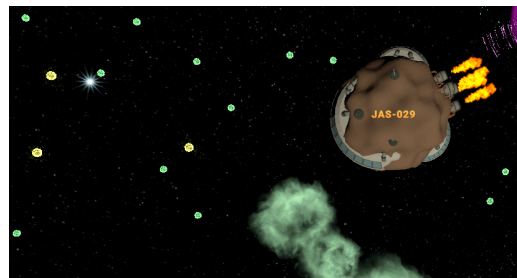



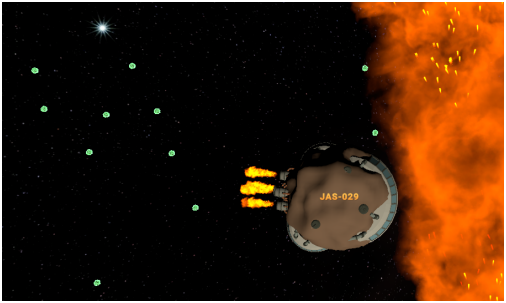
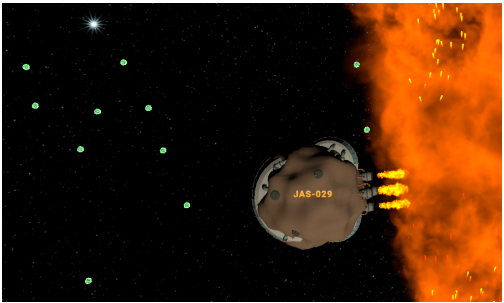
Kondisi bot player mengaktifkan *Shield* saat ditembaki Torpedo Salvo oleh bot lawan



Kondisi bot player ketika berbalik arah untuk menghindari obstacle Gas Clouds

Posisi bot sebelum berbalik arah:



	<p>Posisi bot setelah berbalik arah:</p> 
<p>Kondisi bot player ketika berbalik arah untuk menghindari radius world</p>	<p>Posisi bot sebelum berbalik arah:</p>  <p>Posisi bot setelah berbalik arah:</p> 

BAB V

KESIMPULAN DAN SARAN

I. Kesimpulan

Dari tugas besar ini dapat disimpulkan hal-hal sebagai berikut,

- 1) Algoritma Greedy dapat dipergunakan untuk proses penyusunan strategi bot kapal yang ada dalam game Galaxio.
- 2) Dalam Game Galaxio, terdapat berbagai macam opsi kejadian yang mungkin terjadi pada bot sehingga setiap kemungkinan kejadian perlu dievaluasi.

II. Saran

Saran untuk penyelesaian masalah ini untuk kedepannya adalah sebagai berikut,

- 1) Terdapat bug dalam program yang menyebabkan adanya objek yang tidak muncul dalam map permainan (Contoh: supernova) sehingga bug seperti ini bisa diperbaiki kedepannya.
- 2) Terdapat beberapa kondisi tertentu yang masih belum bisa dihandle oleh program sehingga program dapat dikembangkan kedepannya agar menjadi lebih baik lagi.

DAFTAR PUSTAKA

- [1] R. Munir (2021). Algoritma Greedy Bagian 1 [Powerpoint Slides]. Available: [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)
- [2] “*Algoritma Greedy*.”. Amethyst Aiko. <https://www.amethystaiko.com/algoritma-greedy/> (Diakses pada tanggal 11 Februari 2023)

IF2211
Strategi Algoritma

LAMPIRAN

Pranala Github : https://github.com/jasonrivalino/Tubes1_JAS-029

Video Demonstrasi : https://youtu.be/6f4_4flntqM