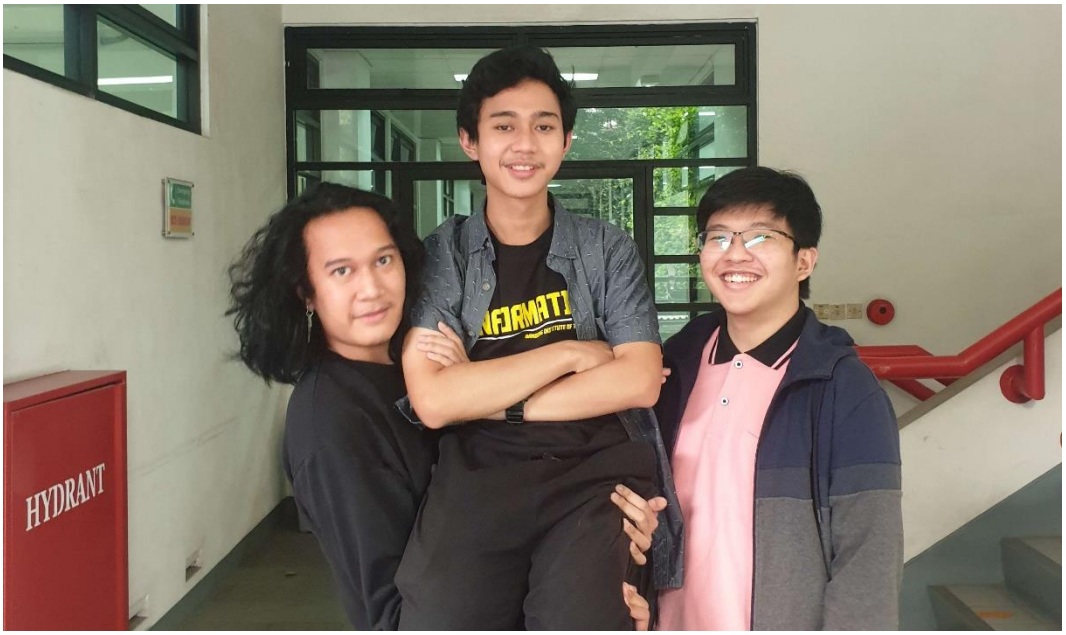
LAPORAN TUGAS BESAR 1
IF2211 STRATEGI ALGORITMA
SEMESTER II 2022-2023

Implementasi Algoritma Greedy pada Galaxio

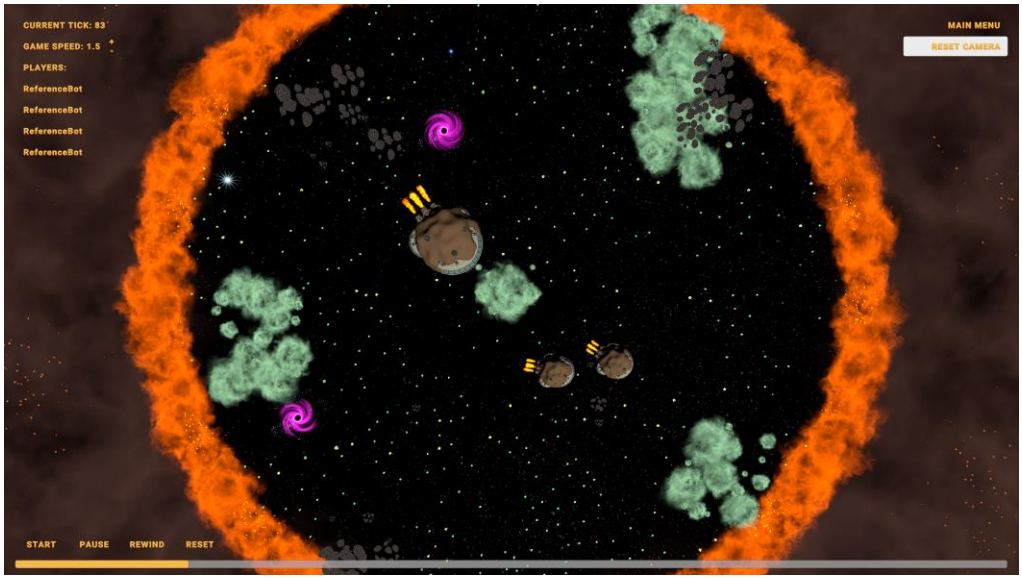
Disusun oleh:
Kelompok 34 (breve)

Ammar Rasyad Chaeroel	13521136
Edia Zaki Naufal Ilman	13521141
Bintang Dwi Marthen	13521144



PROGRAM STUDI
TEKNIK INFORMATIKA
SEKOLAH TEKNIK ELEKTRO
DAN INFORMATIKA
INSTITUT TEKNOLOGI BANDUNG 2023

1. DESKRIPSI TUGAS



Gambar 1.1 Permainan Galaxio

Galaxio adalah sebuah permainan *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 ini dibatasi algoritma *greedy*). Penjelasan lebih lanjut mengenai permainan Galaxio akan dibahas pada bagian 2.2 *Game Engine* Galaxio.

2. LANDASAN TEORI

2.1. Algoritma *Greedy*

Algoritma *greedy* adalah algoritma yang mengkonstruksi suatu solusi utuh permasalahan dengan terus menerus memeilih pilihan terbaik yang dapat diperoleh pada suatu saat tanpa memperhatikan konsekuensi ke depannya dengan harapan optimum lokal tersebut akan membawa kepada optimum global. Karena algoritma *greedy* tidak mengenal *backtracking*, algoritma *greedy* biasanya sangatlah efisien. Akan tetapi, karena tidak mengenal *backtracking* tersebut pulalah algoritma *greedy* menjadi sulit dirancang karena setiap optimum lokal harus dapat mencapai kepada optimum global. Dalam algoritma *greedy* terdapat beberapa elemen (semuanya tidak akan selalu ada):

1. Himpunan Kandidat (*C*)
   Berisikan kandidat pilihan yang akan dipilih setiap langkahnya
2. Himpunan Solusi (*S*)
   Berisi kandidat yang telah dipilih
3. Fungsi Solusi

Menentukan apakah himpunan kandidat yang dipilih sudah memberikan solusi

4. Fungsi Seleksi
   Memilih kandidat solusi berdasarkan strategi *greedy* (bersifat heuristik) tertentu yang digunakan
5. Fungsi Kelayakan
   Memeriksa apakah suatu kandidat dapat dipilih dimasukkan ke dalam himpunan solusi
6. Fungsi Obyektif
   Memaksimumkan atau meminimumkan

2.2. *Game Engine* Galaxio

Game engine yang digunakan pada tugas besar ini merupakan *game engine* yang digunakan pada *Entelect Challenge* 2021 yaitu Galaxio. Seperti yang dipaparkan pada bagian 1. Deskripsi Tugas, Galaxio merupakan sebuah permainan *battle royale* yang mempermainkan sejumlah bot (diatur oleh pengaturan *game engine*) yang diimplementasikan dengan algoritma tertentu. Pengimplementasian bot Galaxio dapat dilakukan melalui berbagai bahasa pemrograman: *C++, Java, JavaScript, Kotlin, NetCore, dan Python*. *Game engine* dan *starter pack* untuk Galaxio dapat diunduh melalui tautan berikut: *https://github.com/EntelectChallenge/2021-Galaxio/releases/tag/2021.3.2*.

Peta atau arena permainan Galaxio adalah sebuah bidang dua dimensi berbentuk lingkaran dengan titik tengah memiliki kordinat (0,0). Pada awalnya, seluruh arena dapat diakses oleh pemain tanpa konsekuensi (dengan pengecualian bila pemain terkena objek tertentu) dan akan mengecil senantiasa dengan sejalannya permainan seperti permainan *battle royale* pada umumnya. Dalam Galaxio, segala sesuatu yang berada di luar zona akan dihilangkan dari permainan dengan pengecualian bot pemain yang akan mengecil sebesar satu ukuran setiap ticknya.

Terdapat berbagai objek dalam permainan Galaxio yang semuanya berbentuk lingkaran: makanan, makanan super, lubang cacing, awan gas, medan asteroid, torpedo, *supernova*, dan *teleporter*. Berikut penjelasan berbagai objek tersebut (segala nilai yang disebutkan merupakan nilai bawaan *game engine*, perlu diperhatikan bahwa terdapat nilai-nilai yang dapat diubah pada peraturan):

1. Makanan
   Makanan merupakan objek statis yang tersebar di arena permainan dengan ukuran tiga. Ketika bot pemain bertubrukan dengan makanan, ukuran bot pemain akan bertumbuh sebanyak tiga ukuran (sesuai dengan ukuran makanan).
2. Makanan Super
   Makanan super merupakan variasi dari makanan yang menyebabkan pemain akan bertumbuh sebanyak enam ukuran bila memakan makanan (dua kali dibanding biasa) selama lima tick sejak memakan makanan super. Setiap makanan yang muncul di arena memiliki peluang tertentu untuk menjadi makanan super.

3. Lubang Cacing
   Lubang cacing merupakan suatu objek yang akan menteleportasi suatu objek yang masuk ke lubang cacing lainnya. Karena sifatnya yang menteleportasi pemain, lubang cacing selalu terdiri dalam suatu pasangan yang tidak dapat diketahui oleh pemain selain melalui percobaan (informasi pasangan suatu lubang cacing dienkapsulasi). Karena terdiri dalam sepasang, bila suatu lubang cacing berada di luar zona maka kedua lubang cacing akan dihapuskan sekaligus. Sepanjang permainan lubang cacing akan membesar hingga suatu ukuran tertentu sesuai pengaturan dan akan mengecil sebesar setengah ukuran objek yang melewatinya. Melewati lubang cacing tidak memiliki penalti apapun, bersifat instan, dan momentum serta arah gerak objek akan dipertahankan.
4. Awan Gas
   Awas gas merupakan objek yang akan tersebar di arena permainan dengan sifat merusak. Awan gas bersifat seperti zona yang berada di dalam arena permainan sehingga ia akan mengurangi ukuran setiap bot yang berada di dalamnya sebanyak satu ukuran setiap ticknya.
5. Medan Asteroid
   Medan asteroid merupakan objek yang akan tersebar di arena permainan dengan sifat menghambat. Setiap bot yang melalui medan asteroid akan menjadi lambat sebanyak setengah dari kecepatan aslinya.
6. Torpedo
   Torpedo merupakan sebuah proyektil yang ditembakkan oleh bot pemain yang akan mengurangi ukuran semua objek yang bertabrakan dengannya. Torpedo memiliki arah gerak lurus, ukuran awal sepuluh, dan kecepatan 60. Torpedo akan terus melaju selama ia masih memiliki ukuran (belum hilang dari permainan), ukurannya sendiri akan dikurangi sebanyak ukuran obyek yang ditabraknya (kecuali lubang cacing dan antar torpedo dapat bertabrakan). Bila torpedo yang ditembakkan oleh suatu bot pemain mengenai bot pemain lain, maka bot pemain yang menembakkannya akan mendapatkan ukuran sebanyak ukuran yang dikurangi dari bot pemain yang terkena torpedo.
7. *Supernova*
   Supernova merupakan sebuah senjata yang hanya akan muncul sekali dalam satu permainan antara saat seperempat awal permainan maupun seperempat akhir permainan. *Supernova* harus diambil oleh bot pemain ketika telah berada di arena. *Supernova* merupakan sebuah bom yang bergerak layaknya sebuah torpedo tetapi ia tidak akan menabrak apapun selama geraknya. *Supernova* harus diaktifkan oleh pemain yang menembakkannya untuk menyebabkan segala bot pemain yang berada di dalam radius ledakaannya menderita serangan luar biasa serta menghasilkan awan gas yang besar di area ledakannya. Ketika bot pemain yang memilik *supernova* mati, maka *supernova* itu akan ikut mati bersama bot pemain tersebut.

8. *Teleporter*
   Teleporter merupakan sebuah proyektil yang dapat ditembakkan oleh bot pemain untuk berteleportasi. Ketika sebuah bot pemain menembakkan *teleporter*, ia akan dikenakan biaya sebesar 20 ukuran untuk menembakkannya. *Teleporter* bergerak dengan kecepatan 20 dan suatu bot pemain hanya dapat memiliki sepuluh *teleporter* dalam suatu waktu (satu *teleporter* akan diberikan setiap 100 tick).

Dalam Galaxio, bot pemain akan mengambil wujud sebuah kapal luar angkasa. Kapal luar angkasa tersebut akan memiliki ukuran inisial 10 dan kecepatan inisial 20 (akan berkurang seiring dengan membesarnya pemain). Bila sebuah kapal luar angkasa memiliki ukuran di bawah lima, maka ia akan hancur. Berikut penjelasan aksi dan teknis kapal luar angkasa pemain:

1. Gerakan
   Sebuah kapal luar angkasa pada awalnya hanya akan diam hingga diberikan suatu perintah gerakan. Ketika sebuah kapal luar angkasa telah menerima perintah gerakan, ia akan mengikuti perintah gerakan tersebut hingga diberikan perintah stop. Kecepatan pemain ditentukan dengan rumus 200 dibagi ukuran pemain (dibulatkan ke atas dengan kecepatan minimum satu). Arah gerak sebuah kapal luar angkasa dienumerasi dalam derajat dari 0 hingga 359.
2. *Afterburner*
   Afterburner merupakan mekanisme yang meningkatkan kecepatan gerak kapal luar angkasa sebanyak dua kali lipat dari aslinya. Penggunaannya memiliki penalti sebesar satu unit per tick. Perlu diperhatikan bahwa penggunannya dapat menyebabkan kapal luar angkasa untuk hancur bila penalti yang dikenakan menyebabkan ukuran kapal luar angkasa untuk berada di bawah batas ukuran minimum.
3. Torpedo
   Seperti dijelaskan pada bagian objek permainan, sebuah kapal luar angkasa dapat menembakkan torpedo. Sebuah kapal luar angkasa akan mendapatkan satu amunisi torpedo setiap sepuluh tick dengan kapasistas maksimal lima torpedo. Kapal luar angkasa dapat menembakkan sebuah rentetan torpedo yang terdiri dari sepuluh torpedo dengan penalti lima ukuran. Perlu diperhatikan bahwa penembakkan torpedo dapat menyebabkan kapal luar angkasa untuk hancur bila penaltinya menyebabkan ukuran kapal luar angkasa di bawah batas ukuran minimum.
4. Perisai
   Perisai merupakan mekanisme pertahanan kapal luar angkasa dari serangan torpedo. Perisai akan aktif selama dua puluh tick dan memiliki jeda dua puluh tick sebelum dapat digunakan lagi. Penggunaan perisai memberikan penalti sebanyak dua puluh ukuran kepada pemain. Ketika perisai aktif, torpedo yang mengenai kapal luar angkasa akan memantul dan kapal luar angkasa yang terkena akan berubah arah geraknya ke arah gerak awal torpedo.

5. Tabrakan
   Ketika sebuah kapal luar angkasa bertabrakan dengan kapal lain, kapal yang lebih besar akan mengonsumsi ukuran kapal yang lebih kecil sebanyak setengah ukuran kapal yang lebih besar (dengan batas atas ukuran kapal luar angkasa yang lebih kecil). Setelah bertabrakan, arah gerak kedua kapal akan berputar sebanyak seratus delapan puluh derajat.

Bot yang dimainkan dalam Galaxio dalam tugas besar ini diimplementasikan dalam bahasa pemrograman Java dengan pemrograman berbasis objek. Dalam pengimplementasiannya, pemain cukup mengubah fungsi *computerNextPlayerAction* pada berkas *BotService.java* yang terdapat dalam folder *Services* dan pemain dapat menamai botnya dengan mengubah parameter ketiga (pada pemanggilan fungsi *hubConnection.send*) pada baris ke-61 pada berkas *Main.java*. Dalam *game engine* terdapat beberapa fungsi (diluar fungsi OOP) yang telah disediakan untuk mempermudah pemain dalam mengimplementasi strateginya: mencari jarak antara dua obyek dan menghitung arah gerak suatu obyek ke obyek lain.

Untuk menjalankan permainan Galaxio, pemain perlu memulai tiga instan untuk *game engine*: instan untuk *GameRunner.dll* (terdapat dalam folder *runner-publish* pada *starter-pack*), instan untuk *Engine.dll* (terdapat dalam folder *engine-publish* pada *starter-pack*), dan instan untuk *Logger.dll* (terdapat dalam folder *logger-publish* pada *starter-pack*). Setelah itu pemain menghubungkan bot sebanyak yang diatur pada pengaturan ke permainan (dalam konteks ini adalah bot dalam bentuk berkas .jar). Berikut merupakan contoh berkas *.bat* yang digunakan untuk menghubungkan empat bot dengan struktur folder sebagai berikut:

Struktur Folder

```
.
|----- starter-pack
|       |----- engine-publish
|       |----- logger-publish
|       |----- reference-bot-publish
|       |----- runner-publish
|       |----- starter-bots
|       |----- visualiser
|----- Tubes1_breve
        |----- doc
        |----- src
        |----- target
```

** perlu diperhatikan ketika package Java dibuild file .jar akan berada di dalam folder target*

Isi berkas *.bat* yang akan dijalankan dari direktori Tubes1_breve

```
@echo off
cd ..
cd starter-pack
```

```
cd runner-publish
start dotnet GameRunner.dll
cd ..
cd engine-publish
start dotnet Engine.dll
cd ..
cd logger-publish
start dotnet Logger.dll
cd ..
cd ..
cd Tubes1_breve
cd target
timeout /t 3
start "" java -jar JavaBot.jar
timeout /t 3
start "" java -jar JavaBot.jar
timeout /t 3
start "" java -jar JavaBot.jar
timeout /t 3
start "" java -jar JavaBot.jar
```

Ketika permainan telah selesai, pemain dapat melihat visualisasi dari permainan yang telah dimainkan dengan menjalankan *visualiser* pada folder *visualiser* yang didapatkan dari *starter-pack*. Pemain perlu memuat berkas yang dihasilkan oleh *Logger.dll* pada *visualiser* dan *visualiser* akan memutar visual permainan yang tergambar seperti pada gambar 1.1.

3. APLIKASI STRATEGI GREEDY

3.1. Pemetaan Galaxio ke Elemen-Elemen Algoritma *Greedy*

Dalam permainan Galaxio, seorang pemain dapat melakukan sepuluh aksi: bergerak maju, berhenti, memulai *afterburner*, menghentikan *afterburner*, menembakkan torpedo, menembakkan *supernova*, meledakkan *supernova*, menembakkan *teleporter*, mengaktifkan *teleporter*, dan mengaktifkan perisai. Kesepuluh aksi tersebut dimasukkan ke dalam himpunan kandidat secara implisit melalui aksi-aksi bot dengan beberapa pengecualian yaitu berhenti, mekanisme *supernova*, dan mengaktifkan perisai.

Untuk fungsi solusi, fungsi kelayakan, fungsi solusi, dan fungsi objektif tidak dapat diimplementasikan secara langsung dalam persoalan ini karena permasalahan yang diselesaikan adalah kompetisi antara bot pemain yang strateginya berbeda-beda (tidak dapat diprediksi pada waktu pembuatan algoritma *greedy*). Akan tetapi, bila perlu dipetakan secara implisit maka fungsi objektif adalah menjadikan bot yang ditandingkan menjadi bot terakhir yang bertahan dengan fungsi kelayakan mengecek apakah gerakan yang dipilih mempertahankan status “hidup” dari bot dan fungsi solusi mengecek apakah solusi membawa bot ke kemenangan.

Implementasi fungsi seleksi digunakan dengan mengecek kondisi dari bot yang dimainkan pada suatu saat itu. Dalam implementasinya terdapat beberapa faktor yang digunakan: ukuran bot pemain, ukuran bot lawan, apakah pemain baru saja menembak torpedo, target yang sedang diincar pemain, apakah pemain telah menembakkan teleporter, apakah pemain tersangkut dalam suatu posisi, dan jarak pemain ke lawan. Berikut merupakan tabel kondisi serta keputusan yang akan diambil dengan kondisi yang berada semakin di atas tabel menjadi prioritas yang lebih tinggi (tabel berfungsi sebagai fungsi seleksi):

No	Kondisi	Aksi
1	Pemain berada pada posisi yang sama selama 10 tick	Aktifkan afterburner
2	Pemain baru saja mengaktifkan afterburner	Matikan afterbuner
3	Pemain berada di tepian arena tetapi kondisi tidak mendukung untuk menggunakan teleporter	Bergerak ke tengah arena
4	Terdapat lawan yang lebih kecil dengan *treshold* tertentu dalam jarak yang relatif dekat dengan pemain	Kejar lawan tersebut
5	Terdapat lawan yang lebih besar dengan *treshold* tertentu dan dekat dengan pemain	Kabur dari lawan tersebut
6	Terdapat awas gas yang dekat dengan pemain	Kabur dari awan gas tersebut
7	Default dengan ukuran terlalu kecil untuk menembakkan torpedo	Cari makanan terdekat
8	Default	Tembakkan torpedo ke musuh terdekat dan cari makanan terdekat

Melalui tabel di atas, himpunan kandidatnya adalah: {aktifkan afterburner, matikan afterburner, bergerak ke tengah arena, kejar lawan tersebut, kabur dari lawan tersebut, kabur dari awan gas tersebut, cari makanan terdekat, tembakkan torpedo ke musuh terdekat}.

3.2. Eksplorasi Solusi Alternatif

Selain kondisi-kondisi yang terdapat pada tabel di subbab 3.1, terdapat beberapa alternatif solusi yang terpikirkan: bermain seagresif mungkin, bermain sedefensif mungkin, dan beberapa perubahan kecil dalam pengambilan keputusan. Bermain seagresif mungkin berarti mengirim bot dalam suatu misi bunuh diri untuk meningkatkan ukurannya dan terus menerus menembakkan torpedo ke lawan tanpa memperhatikan ancaman-ancaman di dekatnya. Bermain sedefensif mungkin berarti mengirim bot dalam suatu aksi gerilya yang terus-menerus kabur dan

hanya memakan makanan yang berada di jalur ia kabur dan menghindari konflik dengan lawan. Beberapa perubahan kecil yang terpikirkan adalah:

1. Penggunaan *afterburner* untuk mengejar lawan
   Afterburner
2. Penggunaan *afterburner* untuk bergerak ke tengah arena
3. Penggunaan *afterburner* dalam kabur dari awan gas
4. Penggunaan perisai
5. Penggunaan *supernova*
6. Penggunaan *teleporter* untuk mengejar lawan yang dapat dimakan dan menjauh dari lawan yang lebih besar
7. Menghindari medan asteroid
8. Menghindari *teleporter* lawan secara aktif

3.3. Analisis Solusi

Solusi yang dipilih tertuang pada tabel yang terdapat pada subbab 3.1 mengenai kondisi tertentu dan aksi yang harus diambil karena telah dikonsiderasi sebagai gaya bermain defensif tetapi tetap seagresif mungkin. Pendekatan defensif tetapi tetap seagresif mungkin berarti berusaha bertahan hidup tetapi juga terus-menerus mencari celah untuk meningkatkan ukuran dan daya serang.

Secara intuitif, bermain seagresif mungkin berarti mengirim bot dalam suatu misi bunuh diri untuk meningkatkan ukurannya dengan terus-menerus memakan makanan dan lawan. Akan tetapi, pendekatan akan menyebabkan bot memiliki masa hidup yang pendek karena ia akan menerjang awan gas dan lawan yang memakannya. Tidak hanya disitu saja, bot juga akan terus menembakkan torpedo meskipun ukurannya tidak cukup sehingga bot akan hancur.

Di lain sisi, bermain sedefensif mungkin berarti mengirim bot untuk terus-menerus kabur dan hanya memakan makanan yang terdapat pada jalur kaburnya. Pendekatan ini mungkin akan bertahan hidup lama karena ia dengan mudah kabur dari berbagai ancaman dengan ukurannya yang kecil sehingga kecepatannya akan tinggi. Akan tetapi, bot tidak akan memiliki ukuran dan daya serang yang cukup pada akhir permainan sehingga bot hanya akan berada di posisi kedua dalam kondisi terbaik.

Pengkalibrasian tingkat defensif dan agresif yang tepat diperlukan untuk mendapatkan suatu solusi yang dapat dikatakan optimal. Berikut merupakan konsiderasi mengapa perubahan-perubahan yang disebutkan pada subbab 3.2 tidak diimplementasikan:

1. Penggunaan *afterburner* untuk mengejar lawan, bergerak ke tengah arena, dan kabur dari awan gas
   Afterburner tergolong boros dalam penggunaannya sehingga ketika pemain ingin menghindari bahaya, ia justru menempatkan dirinya dalam kondisi yang lebih tidak menguntungkan karena borosnya *afterburner*. Selain, *afterburner* dalam mengejar lawan tidaklah efektif karena akan terdapat pengurangan ukuran yang cukup signifikan ketika pemain ingin meningkatkan ukurannya dengan memakan lawan.
2. Penggunaan perisai

Implementasi penggunaan perisai tergolong ke zona abu-abu karena bot harus mendeteksi apakah torpedo yang didekatnya merupakan torpedo tembakannya atau torpedo lawan. Pendeteksian tersebut memerlukan suatu *backtracking* untuk mencatat id dari torpedo yang ditembakannya. Penggunaan pendekatan mendeteksi apakah arah gerak torpedo ke arah bot cukup bermasalah pada *game engine*.

3. Penggunaan *supernova*
   Implementasi pencarian pengambilan *supernova* bermasalah karena bot akan mendeteksi zona munculnya *supernova* meskipun belum ada *supernova*.
4. Penggunaan *teleporter* untuk mengejar lawan yang dapat dimakan
   Langkanya jumlah *teleporter* menyebabkan penggunaan yang difokuskan untuk hal-hal defensif sesuai dengan pendekatan defensif sebagai utama. Menghindari menembakkan *teleporter* ke arah lawan yang dapat dimakan merupakan perwujudan konservasi dari *teleporter* untuk hal-hal defensif yang jauh lebih pasti dibandingkan berusaha memakan lawan. Terlebih lagi, terdapat beberapa *instance* dimana perintah *fire teleporter* dan *teleport* tidak bekerja seperti semestinya. Sehingga agar program lebih konsisten dan bekerja seperti yang seharusnya, dibuat keputusan untuk tidak mengimplementasikan strategi ini.
5. Menghindari medan asteroid
   Dengan menghindari medan asteroid juga, terdapat terlalu banyak hal yang harus dihindari oleh bot. Hal tersebut mungkin sejalan dengan prinsip defensif dari bot, tetapi ia akan menjadi terlalu defensif. Medan asteroid tidak bersifat merusak tetapi hanya bersifat menghambat sehingga bot akan menerjangnya sebagai wujud dari pendekatan agresifnya. Ketika bot akan menghindari medan asteroid yang pada permainan tidak terlalu banyak, ia tidak akan berkembang dengan cukup untuk mendapatkan daya saing di akhir permainan (perlu diingat bahwa tujuan permainan adalah menjadi pemain terakhir yang bertahan sehingga otomatis pada fase akhir permainan pemain harus dapat memenangkan *duel* dengan lawan).
6. Menghindari *teleporter* lawan secara aktif
   Penghindaran *teleporter* lawan dapat dikatakan sebagai pendekatan yang terlalu defensif seperti pada poin 5 menghindari medan asteroid. Selain itu, *teleporter* akan terhindar secara pasif seiring dengan jalannya bot bila dapat dihindari.
7. Tidak melakukan prediksi ke depan dengan perhitungan
   Mengacu pada QnA tugas besar ini, tepatnya pertanyaan nomor 14 “Apakah boleh melakukan prediksi perhitungan? (Contoh: menghitung total tik yang dibutuhkan untuk sampai ke titik tertentu jika state saat ini diasumsikan tetap untuk setiap tik tersebut)” dengan jawaban “Tidak Boleh. Alasan: Berdasarkan implementasinya, prediksi perhitungan dapat dikategorikan sebagai algoritma greedy ataupun yang lain (zona abu-abu). Jadi agar memastikan algoritma yang dipakai

adalah greedy, cukup memanfaatkan nilai dari state pada saat itu bukan prediksi state didepannya." maka segala perhitungan prediksi ke depan dianggap sebagai zona abu-abu yang menyebabkan tidak diimplementasikan.

Pengambilan keputusan berdasarkan tabel pada subbab 3.1 dinilai optimal dengan pertimbangan berikut:

1. Pengaktifkan *afterburner* untuk keluar dari posisi yang sama merupakan salah satu solusi untuk mengatasi *bug* bot terus menerus berpindah *heading* karena terdapat dua target yang ingin ia capai dengan jarak yang sama (seperti terdapat dua makanan dengan jarak yang sama dari bot)
2. Penonaktifan *afterburner* dengan segera merupakan wujud konservasi dari ukuran bot karena borosnya *afterburner*
3. Pengejaran lawan yang cukup kecil dalam jarak yang cukup dekat merupakan pendekatan efisiensi dalam meningkatkan ukuran, ketimbang mengejar makanan yang hanya meningkatkan tiga ukuran maka lebih efektif bila memakan lawan yang ukurannya minimal lima dalam jarak dekat (perlu diingat bahwa peningkatan ukuran ketika memakan lawan adalah sebanyak ukuran lawan)
4. Menghindari lawan yang lebih besar dan awan gas yang cukup dekat serta berhenti menembakkan torpedo ketika ukuran terlalu kecil merupakan pendekatan intuitif untuk meningkatkan durasi hidup dari bot dalam permainan dengan menghindari hancurnya bot
5. Mencari makanan terdekat adalah intuisi dasar dalam meningkatkan ukuran bot sebagai modal awal dalam berkompetisi dengan bot lainnya karena dalam Galaxio ukuran memainkan peran yang penting dalam menentukan daya saing bot
6. Terdapat *treshold* tertentu bagi lawan yang dinilai dapat dimakan dan lawan yang dianggap sebagai ancaman karena bila menggunakan ukuran secara mentah pergerakan bot akan menjadi terlalu tidak stabil. Ketika ia ingin mengejar lawan yang lebih kecil sedikit saja daripada dirinya, ia akan berhenti mengejarnya ketika bot tersebut telah berhasil memakan makanan ketika dirinya belum yang menyebabkan kondisi berputar terbalik. Di lain sisi, bila semua lawan yang lebih besar dianggap mentah-mentah sebagai ancaman maka bot akan bermain terlalu defensif yang menyebabkan bot hanya akan kabur sepanjang permainan sehingga bot tidak akan terlalu berkembang.

4. IMPLEMENTASI DAN PENGUJIAN

4.1. Implementasi Strategi

4.1.1. computeNextPlayerAction

```
procedure computeNextPlayerAction(output
playerAction: PlayerAction)
{Mengkalkulasi aksi berikutnya untuk bot}

DEKLARASI
protocol: integer
```

```
prey: list of game object
predator: list of game object

ALGORITMA
playerAction.Action <- FORWARD
protocol <- checkProtocol() {mengambil protocol}
switch protocol
      case 1: {Protocol offensive}
            prey <- List player yang ukurannya lebih
kecil pada jarak tertentu
            if prey tidak kosong AND bot size cukup
AND jumlah salvo tidak kosong then
                  target <- prey terdekat
                  playerAction.action    <-    FIRE
TORPEDOES
            playerAction.heading                  <-
getHeadingBetween(target) {memasukkan arah musuh}
            break
      case 2: {protocol defensive}
            predator <- List player yang ukurannya
lebih besar pada jarak tertentu
            if predator tidak kosong then
                  if jumlah teleporter tidak kosong
AND bot size cukup then
                        escape() {kabur dari lawan}
                  else
                        if afterburner tidak aktif
AND bot size cukup then
                              toggleafterburner()
            playerAction.heading<- arah berlawanan
dengan target
      case -1:
            playerAction.heading<- arah tengah dunia
            target <- tengah dunia
      default:
            playerAction.heading   <-   findtarget()
{cari target lain}
      setPlayerAction(playerAction)  {set  perintah
dan arah ke bot}
```

4.1.2. checkProtocol

```
function checkProtocol() -> integer
{mendapatkan protokol yang tepat pada situasi sekarang}

DEKLARASI
-

ALGORITMA
if radius dunia != null AND jarak antara batas dunia
dengan bot cukup dekat then
      -> -1 {protocol evade border}
else if terdapat musuh yang lebih kecil pada jarak
tertentu then
```

```
      -> 1 {protocol offensive}
else if terdapat musuh yang lebih besar pada jarak
tertentu then
      -> 2 {protocol defensive}
-> 0
```

4.1.3. findTarget

```
function findTarget() -> integer
{mencari target bot}

DEKLARASI
heading: integer
nearestFood: list of game object
nearestGasCloud: list of game object
target: list of game object

ALGORITMA
heading <- 0
nearestFood <- makanan terdekat
nearestGasCloud <- gas coud terdekat

if gas cloud terdekat ada AND jarak antar bot dan gas
cloud cukup dekat then
      heading <- arah berlawanan dari gas cloud
      if afterburner tidak aktif AND bot size cukup then
            toggleAfterburner()
else if makanan terdekat ada then
      if afterburner aktif then
            toggleAfterburner()
      heading<- arah makanan
      target <- makanan terdekat
-> heading
```

4.1.4. toggleAfterburner

```
procedure toggleAfterburner()
{Menyalakan atau mematikan afterburner bot}

DEKLARASI

ALGORITMA
if afterburner aktif then
      afterburner dinonaktifkan
else
      afterburner diaktifkan
```

4.2. Struktur Data

No.	Struktur Data	Atribut	Deskripsi
1	GameObject	UUID	Identifier khusus dari suatu objek permainan
		size	Ukuran dari suatu objek permainan

		speed	Kecepatan gerak suatu objek permainan
		currentHeading	Arah gerak objek permainaan saat ini
		position	Posisi objek permainan dalam bidang dua dimensi
		gameObjectType	Jenis dari objek permainan (apakah makanan, torpedo, atau lainnya)
		Effects	Identifier enumerasi efek yang sedang dialami (bitmapping efek, contoh: efek awan gas adalah 2 dan efek medan asteroid adalah 1, sehingga efek kumulatifnya adalah 3)
		TorpedoSalvoCount	Jumlah torpedo yang sedang dimiliki oleh kapal atau bot
		SupernovaAvailable	Apakah supernova dimiliki oleh kapal atau bot
		TeleporterCount	Jumlah teleporter yang sedang dimiliki oleh kapal atau bot
		ShieldCount	Jumlah perisai yang dapat digunakan oleh kapal atau bot
2	PlayerAction	playerId	Identifier unik dari pemain
		action	Aksi yang sedang atau akan dilakukan oleh pemain
		heading	Arah kapal pemain
3	GameState	world	Arena permainan Galaxio (menyimpan tick sekarang, radius zona, dan kordinat pusat arena)
		gameObjects	Objek permainan yang sedang berada di dalam permainan
		playerGameObjects	Objek permainan yang berasal dari pemain atau bot (seperti teleporter dan torpedo)

4.3. Pengujian

4.3.1. Protokol *Offensive*

Pada implementasi protokol *offensive*, bot diprogram untuk mengarah ke bot musuh yang ukurannya lebih kecil dari bot player atau selisih antar kedua bot yang cukup kecil, kemudian menembakkan torpedo ke arah tersebut. Sebelum ditembaknya torpedo ke bot player lain, dipastikan terlebih dahulu jika ukuran bot dan jumlah salvo yang tersedia pada bot mencukupi. Jika kondisi tidak terpenuhi, maka torpedo tidak akan ditembakkan.

Implementasi ini merupakan bentuk dari strategi *attack* pada bot untuk mengeliminasi bot player lain. Implementasi ini memiliki banyak kekurangan yang berhubungan dengan akurasi penembakkan. Program pada protocol *offensive* tidak memprediksi dan mengkalkulasi gerakan bot musuh sehingga terdapat beberapa kasus di mana torpedo yang ditembakkan meleset. Tak hanya itu, karena

tidak adanya pembatasan dalam penembakan torpedo, bot akan menembakkan torpedo secara *rapid* sehingga dapat mengurangi ukuran bot secaa signifikan dalam jangka waktu yang kecil.

Disamping kekurangannya, program protocol *offensive* ini berjalan dengan sesuai. Pada jarak tertentu, bot akan menembakkan torpedo pada bot player lain yang kemudian akan terkena dengan torpedo tersebut dan ukurannya berkurang. Ukuran yang berkurang dari bot musuh akan kemudian diambil oleh bot player sehinnga memperbesar kesempatan bot untuk mengonsumsi player lain.

4.3.2. Protokol *Defensive*

Pada implementasi protokol *defensive*, bot diprogram untuk mengarah berlawanan dengan bot musuh yang ukurannya lebih besar dan jaraknya dengan bot player cukup dekat. Setelah menentukan arah untuk menghindari bot musuh, akan diaktifkan *afterburner* untuk selama waktu yang cukup sebentar untuk menjauh dari musuh dengan kecepatan yang cukup besar. Alasan dinyalakannya *afterburner* selama waktu yang sangat kecil adalah agar ukuran bot player tidak berkurang secara signifikan dalam jangka waktu dekat dan menjadi sebuah kesempatan bagi bot player lain untuk mengonsumsi bot.

Meskipun implementasi strategi ini terbukti berguna dan fungsionalnya sesuai dengan tujuannya, terdapat beberapa kekurangan dan kasus dimana strategi ini tidak efektif. Pada banyak kasus dimana bot dihimpit oleh dua atau lebih hadangan, baik itu berupa bot player lain, gas cloud, ataupun batas dunia, bot gagal untuk menghindari ancaman dari hadangan tersebut. Karena bot hanya deprogram untuk mengarah berlawanan musuh terdekat, maka jika dihadapi dua atau lebih hadangan, bot akan terjebak dan tidak dapat menghindar.

4.3.3. Protokol *Evasive*

Pada implementasi protokol *evasive*, bot diprogram untuk menghindari benda rintangan atau hadangan di dunia seperti batas dunia dan gas cloud. Jika bot menemukan sebuah hadangan pada suatu jarak yang relatif dekat, bot akan mengarah pada arah yang berlawanan dari arah asal bot sehingga terhindar dari ancaman dan pengurangan ukuran bot.

Implementasi dari strategi ini terbukti berfungsi dan bekerja, namun sama halnya dengan protocol *defensive*, arah menghindari bot terbatas sehingga jika diapit oleh dua atau lebih hadangan, bot player pada banyak kasus tidak dapat menghindar. Terdapat juga kasus dimana bot menghindar sedikit telat sehingga bot mengalami *damage* terlebih dahulu oleh hadangan selama waktu tertentu, kemudian baru melakukan pemutaran arah.

4.3.4. Protokol *Feed*

Pada implementasi protokol *feed*, bot diprogram untuk mengonsumsi sumber makanan terdekat dari bot. Sumber makanan yang dapat dikonsumsi dapat berupa *food*, *superfood*, dan bahkan bot player lain yang ukurannya lebih kecil dari bot player. Protokol ini

merupakan protocol *default* pada bot, di mana jika suatu kondisi bot pasa suatu waktu tidak memenuhi terjadinya inisiasi protokol *offensive, defensive,* dan *evasive,* maka protocol yang diinisiasi adalah protocol *feed*.

Terdapat kekurangan yang dapat diamati dari implementasi strategi ini pada beberapa kasus, diantaranya adalah ketika makanan terdekat yang dideteksi berposisi pada suatu hadangan seperti gas cloud atau batasan dunia. Pada kasus tersebut, bot akan tetap mencoba mengonsumsi makanan tersebut sehingga bot mengalami damage dalam proses konsumsi. Namun, berkat adanya protokol *evasive*, *damage* yang dialami bot tidak seburuk yang seharusnya.

5. KESIMPULAN DAN SARAN

5.1. Kesimpulan

1. Diperlukan suatu keseimbangan antara gaya bermain agresif dan defensif untuk mencapai kemenangan
2. Gaya bermain defensif dapat membawa pemain ke posisi yang cukup tinggi akan tetapi tidak memenangkan permainan karena daya serang dan ukurannya yang kecil ketika di momen-momen akhir permainan
3. Terdapat beberapa mekanisme dari Galaxio yang sulit diimplementasi dengan batasan algoritma *greedy* karena terdapat dalam zona abu-abu (terdapat pendekatan seperti *backtracking*, pencatatan *state* sebelumnya, dan sebagainya)

5.2. Saran

1. *Source code* program sangat minim dokumentasi sehingga mahasiswa perlu mengeksplor mandiri setiap fungsi yang ada pada *game engine*
2. Terdapat beberapa celah dalam *game engine* yang dapat dieksploitasi oleh mahasiswa dalam permainan seperti torpedo yang akan tembus kepada kapal yang masih diam
3. Terdapat banyak zona abu-abu apa yang dianggap sebagai *greedy* dan apa yang tidak sehingga menyebabkan keraguan dalam implementasi algoritma tersebut
4. Terdapat banyak *bug* pada *game engine* dan diperlukan pula konfigurasi lebih laju *source code* untuk membuat bot yang dihasilkan sesuai dengan harapan
5. Terdapat ketidaksesuaian bobot penilaian pada spesifikasi tugas (laporan 50%, tetapi hanya bab 3 saja yang menjadi konsiderasi, bagian lainnya tidak masuk dalam pembobotan yang tertera)

6. Referensi

Asisten, S. A. (2023, February 17). *QnA Stima*. Retrieved from Google Documents: https://bit.ly/TugasStimaQnA

Asisten, S. A. (2023, Februari 7). *Tugas Besar 1 IF2211 Strategi Algoritma 2023.docx*. Retrieved from Google Documents: https://bit.ly/SpekTubes1Stima

Gozali, W., & Aji, A. F. (2019). *Pemrograman Kompetitif Dasar*. Yogyakarta: CV. Nulisbuku Jendela Dunia. Retrieved from TLX.

Laaksonen, A. (2018). *Competitive Programmer's Handbook Draft.*

Munir, R. (2023, Februari 7). *Algoritma Greedy*. Retrieved from Homepage Rinaldi Munir: https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2020-2021/Algoritma-Greedy-(2021)-Bag1.pdf

ThePhoenixGuy. (2021, Juni 30). *Galaxio Game Rules*. Retrieved from GitHub: https://github.com/EntelectChallenge/2021-Galaxio/blob/develop/game-engine/game-rules.md

7. Lampiran

7.1. Tautan *Repository* GitHub
https://github.com/Marthenn/Tubes1_breve

7.2. Tautan Video YouTube
https://youtu.be/-Fc40UH2SfA