

IMPLEMENTASI ALGORITMA GREEDY DALAM PEMECAHAN BOT PERMAINAN DIAMOND

Tugas Besar

Diajukan sebagai syarat menyelesaikan mata kuliah Strategi Algoritma (IF2211) Kelas RD
di Program Studi Teknik Informatika, Fakultas Teknologi Industri, Institut Teknologi Sumatera



Oleh Kelompok 7: Laskar Koji

Raditya Alrasyid Nugroho	123140125
Muhamad Rafi Ilham	123140173
Rian Rafael Sangap Tamba	123140190

Dosen Pengampu: Winda Yulita, M.Cs.

**PROGRAM STUDI TEKNIK INFORMATIKA
FAKULTAS TEKNOLOGI INDUSTRI
INSTITUT TEKNOLOGI SUMATERA
2025**

DAFTAR ISI

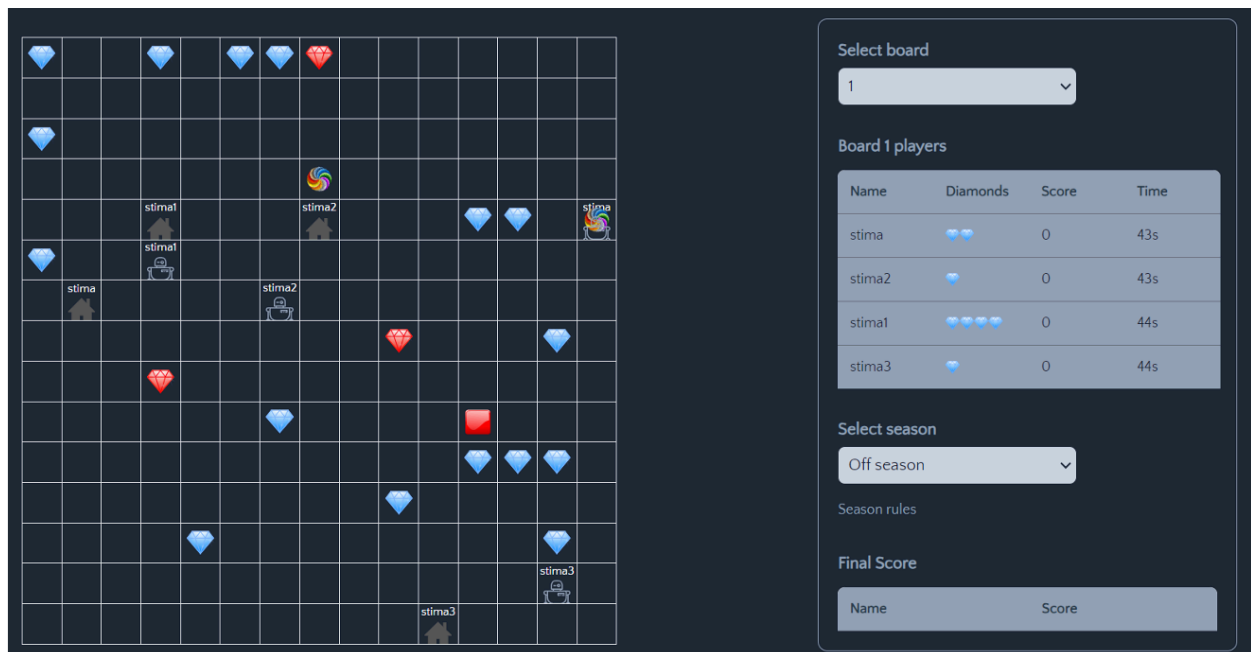
DAFTAR ISI.....	1
BAB I	
DESKRIPSI TUGAS.....	3
BAB II.....	6
LANDASAN TEORI.....	6
2.1 Dasar Teori Algoritma Greedy.....	6
2.2 Cara Kerja Program.....	6
2.3 Implementasi Program.....	7
BAB III.....	12
APLIKASI STRATEGI GREEDY.....	12
3.1 Process Mapping.....	12
3.1.1 Himpunan Kandidat.....	12
3.1.2 Himpunan Solusi.....	14
3.1.3 Fungsi Solusi.....	14
3.1.4 Fungsi Seleksi.....	14
3.1.5 Fungsi Kelayakan.....	15
3.1.6 Fungsi Objektif.....	15
3.2 Eksplorasi Alternatif Solusi Greedy.....	15
3.2.1 Greedy By Highest Value.....	15
3.2.2 Greedy By Highest Density.....	16
3.2.3 Greedy by Tackling.....	16
3.2.4 Greedy by Closest Base.....	16
3.3 Analisis Efisiensi dan Efektivitas Solusi Greedy.....	16
3.4 Strategi Greedy yang Dipilih.....	17
BAB IV.....	18
IMPLEMENTASI DAN PENGUJIAN.....	18
4.1 Implementasi Algoritma Greedy.....	18
4.1.1 Pseudocode.....	18
4.2 Struktur Data yang Digunakan.....	22
4.3 Pengujian Program.....	24
4.3.1 Skenario Pengujian.....	24
4.3.2 Hasil Pengujian dan Analisis.....	25
BAB V.....	27
KESIMPULAN DAN SARAN.....	27
5.1 Kesimpulan.....	27
5.2 Saran.....	28

LAMPIRAN.....	28
DAFTAR PUSTAKA.....	29

BAB I

DESKRIPSI TUGAS

Diamonds merupakan suatu programming challenge yang mempertandingkan bot yang anda buat dengan bot dari para pemain lainnya. Setiap pemain akan memiliki sebuah bot dimana tujuan dari bot ini adalah mengumpulkan diamond sebanyak-banyaknya. Cara mengumpulkan diamond tersebut tidak akan sederhana itu, tentunya akan terdapat berbagai rintangan yang akan membuat permainan ini menjadi lebih seru dan kompleks. Untuk memenangkan pertandingan, setiap pemain harus mengimplementasikan strategi tertentu pada masing-masing bot-nya. Penjelasan lebih lanjut mengenai aturan permainan akan dijelaskan di bawah.



Pada tugas pertama Strategi Algoritma ini, mahasiswa diminta untuk membuat sebuah bot yang nantinya akan dipertandingkan satu sama lain. Tentunya mahasiswa harus menggunakan strategi greedy dalam membuat bot ini.

1. Game engine, yang secara umum berisi:
 - Kode backend permainan, yang berisi logic permainan secara keseluruhan serta API yang disediakan untuk berkomunikasi dengan frontend dan program bot
 - Kode frontend permainan, yang berfungsi untuk memvisualisasikan permainan.

2. Bot starter pack, yang secara umum berisi:
 - Program untuk memanggil API yang tersedia pada backend.
 - Program bot logic (bagian ini yang akan kalian implementasikan dengan algoritma greedy untuk bot kelompok kalian).
 - Program utama (main) dan utilitas lainnya.

Adapun komponen-komponen dari permainan Diamonds antara lain:

1. Diamonds

Untuk memenangkan pertandingan, kita harus mengumpulkan diamond ini sebanyak-banyaknya dengan melewati/melangkahnya. Terdapat 2 jenis diamond yaitu diamond biru dan diamond merah. Diamond merah bernilai 2 poin, sedangkan yang biru bernilai 1 poin. Diamond akan di-regenerate secara berkala dan rasio antara diamond merah dan biru ini akan berubah setiap regeneration.

2. Red Button/Diamond Button

Ketika red button ini dilewati/dilangkahi, semua diamond (termasuk red diamond) akan di-generate kembali pada board dengan posisi acak. Posisi red button ini juga akan berubah secara acak jika red button ini dilangkahi.

3. Teleporters

Terdapat 2 teleporter yang saling terhubung satu sama lain. Jika bot melewati sebuah teleporter maka bot akan berpindah menuju posisi teleporter yang lain.

4. Inventory

Bot memiliki inventory yang berfungsi sebagai tempat penyimpanan sementara diamond yang telah diambil. Inventory ini memiliki kapasitas maksimum sehingga sewaktu waktu bisa penuh. Agar inventory ini tidak penuh, bot bisa menyimpan isi inventory ke base agar inventory bisa kosong kembali.

5. Bots and Bases

Pada game ini kita akan menggerakkan bot untuk mendapatkan diamond sebanyak banyaknya. Semua bot memiliki sebuah Base dimana Base ini akan digunakan untuk menyimpan diamond yang sedang dibawa. Apabila diamond disimpan ke base, score bot

akan bertambah senilai diamond yang dibawa dan inventory (akan dijelaskan di bawah) bot menjadi kosong.

Untuk mengetahui alur dari game ini, berikut ini adalah cara kerja permainan Diamonds.

1. Inisialisasi Pertandingan: Ketika game engine dijalankan, pertandingan akan dimulai dengan meng-host permainan pada hostname yang telah ditentukan.
2. Koneksi antara Engine dan Bot: Setelah engine berhasil dijalankan, engine akan menunggu koneksi dari bot-bot yang dijalankan oleh para pemain. Engine akan melakukan proses logging dan menyiapkan semua informasi yang diperlukan untuk pertandingan.
3. Jumlah bot yang akan berpartisipasi dalam satu pertandingan diatur oleh atribut BotCount yang ada dalam file "appsettings.json" di dalam folder "engine-publish". Pengaturan ini memungkinkan penyesuaian jumlah pemain dalam pertandingan sesuai dengan keinginan.
4. Setiap bot juga memiliki sebuah inventory, dimana inventory berfungsi sebagai tempat penyimpanan sementara diamond yang telah diambil. Inventory ini sewaktu-waktu bisa penuh, maka dari itu bot harus segera kembali ke home base.
5. Apabila bot menuju ke posisi home base, score bot akan bertambah senilai diamond yang tersimpan pada inventory dan inventory bot akan menjadi kosong kembali.
6. Usahakan agar bot anda tidak bertemu dengan bot lawan. Jika bot A menempa posisi bot B, bot B akan dikirim ke home base dan semua diamond pada inventory bot B akan hilang, diambil masuk ke inventory bot A (istilahnya tackle).
7. Selain itu, terdapat beberapa fitur tambahan seperti teleporter dan red button yang dapat digunakan apabila anda menuju posisi objek tersebut.
8. Apabila waktu seluruh bot telah berakhir, maka permainan berakhir. Score masing-masing pemain akan ditampilkan pada tabel Final Score di sisi kanan layar.

BAB II

LANDASAN TEORI

2.1 Dasar Teori Algoritma Greedy

Algoritma Greedy merupakan salah satu pendekatan dalam pemecahan masalah optimasi yang bekerja dengan cara memilih solusi terbaik yang tersedia pada setiap langkah, dengan harapan bahwa pilihan lokal ini akan mengarah pada solusi global yang optimal. Ciri utama dari algoritma ini adalah pengambilan keputusan secara lokal optimal pada setiap tahap, tanpa mempertimbangkan konsekuensi jangka panjang dari keputusan tersebut.

Konsep dasar algoritma greedy berfokus pada prinsip “ambil yang terbaik sekarang, dan lanjutkan.” Meskipun pendekatan ini tidak selalu menghasilkan solusi yang optimal untuk semua jenis masalah, dalam beberapa kasus tertentu seperti Huffman Coding, Prim’s Algorithm, Kruskal’s Algorithm, dan Activity Selection Problem, algoritma greedy dapat memberikan solusi optimal dengan waktu komputasi yang efisien.

Dalam penerapannya, algoritma greedy sering digunakan dalam berbagai masalah optimasi, seperti masalah jalur terpendek, masalah penjadwalan, masalah pemotongan stok, dan banyak lagi. Algoritma Greedy terkadang belum tentu mencapai hasil yang paling optimum karena algoritma ini tidak mengecek seluruh kemungkinan seperti metode exhaustive search. Meskipun algoritma greedy tidak selalu menjamin solusi yang optimal secara global, pendekatan ini seringkali memberikan solusi yang cukup baik dan efisien dalam hal waktu komputasi, terutama untuk masalah-masalah tertentu yang memiliki sifat greedy-choice property dan optimal substructure.

2.2 Cara Kerja Program

Dalam game ini, program dibagi menjadi dua komponen utama: **Game Engine** dan **Bot**. Kedua komponen ini berinteraksi melalui API yang sudah disediakan dalam sistem. Di sisi Bot, program akan menggunakan API tersebut dengan cara mengirimkan permintaan **POST** untuk

mendaftarkan diri menggunakan email dan password. Setelah proses pendaftaran berhasil, Bot bisa **bergabung ke dalam permainan** dengan melakukan POST ke endpoint khusus untuk join, lalu mendapatkan data kondisi board.

Dengan data tersebut, Bot melakukan perhitungan untuk menentukan langkah berdasarkan strategi atau logika yang dimilikinya. Bot lalu mengirimkan langkah tersebut ke backend melalui endpoint **move**. Sebagai balasannya, backend akan mengirimkan kondisi board terbaru setelah langkah dijalankan. Proses ini berulang Bot menganalisis kondisi terkini, menghitung langkah terbaik, lalu mengirimkannya kembali hingga permainan dinyatakan selesai.

2.3 Implementasi Program

Berikut ini adalah langkah-langkah untuk menjalankan program.

1. Pastikan [Node.js](#), Docker, dan Yarn telah terpasang. Pemasangan dapat dilakukan dengan command dibawah.

```
npm install --global yarn
```

2. Jalankan game engine dengan mengunduh folder game engine pada tautan ini <https://github.com/haziqam/tubes1-IF2211-game-engine/releases/tag/v1.1.0>
 - a. Setelah berhasil mengunduh, lakukan ekstraksu file.zip dan masuk ke root dari folder hasil ekstraksi tadi, lalu buka terminal
 - b. Jalankan command berikut ini di terminal untuk masuk ke root directory dari game engine

```
cd tubes1-IF2110-game-engine-1.1.0
```

- c. Install dependencies dengan menggunakan yarn

```
yarn
```

- d. Setup default environment variable dengan menjalankan script berikut

Untuk Windows


```
./scripts/copy-env.bat
```

e. Setup local database dengan membuka aplikasi docker desktop, lalu masukkan command berikut di terminal

```
docker compose up -d database
```

Kemudian jalankan script berikut untuk windows

```
./scripts/setup-db-prisma.bat
```

f. Masukkan command berikut untuk melakukan proses build pada game engine

```
npm run build
```

g. Masukkan command berikut untuk memulai game engine

```
npm run start
```

h. Jika berhasil, maka tampilan terminal akan seperti ini

```

[0] [Nest] 10712 - 31/05/2025, 21.51.17 LOG [NestFactory] Starting Nest application...
[0] [Nest] 10712 - 31/05/2025, 21.51.17 LOG [InstanceLoader] AppModule dependencies initialized +41ms
[0] [Nest] 10712 - 31/05/2025, 21.51.17 LOG [RoutesResolver] BoardsController {/api/boards}: +104ms
[0] [Nest] 10712 - 31/05/2025, 21.51.17 LOG [RouterExplorer] Mapped {/api/boards, GET} route +4ms
[0] [Nest] 10712 - 31/05/2025, 21.51.17 LOG [RouterExplorer] Mapped {/api/boards/:id, GET} route +1ms
[0] [Nest] 10712 - 31/05/2025, 21.51.17 LOG [RoutesResolver] BotsController {/api/bots}: +1ms
[0] [Nest] 10712 - 31/05/2025, 21.51.17 LOG [RouterExplorer] Mapped {/api/bots/:id, GET} route +0ms
[0] [Nest] 10712 - 31/05/2025, 21.51.17 LOG [RouterExplorer] Mapped {/api/bots, POST} route +1ms
[0] [Nest] 10712 - 31/05/2025, 21.51.17 LOG [RouterExplorer] Mapped {/api/bots/recover, POST} route +0ms
[0] [Nest] 10712 - 31/05/2025, 21.51.17 LOG [RouterExplorer] Mapped {/api/bots/:id/join, POST} route +1ms
[0] [Nest] 10712 - 31/05/2025, 21.51.17 LOG [RouterExplorer] Mapped {/api/bots/:id/move, POST} route +0ms
[0] [Nest] 10712 - 31/05/2025, 21.51.17 LOG [RoutesResolver] HighscoresController {/api/highscores}: +0ms
[0] [Nest] 10712 - 31/05/2025, 21.51.17 LOG [RouterExplorer] Mapped {/api/highscores/:seasonId, GET} route +1ms
[0] [Nest] 10712 - 31/05/2025, 21.51.17 LOG [RoutesResolver] RecordingsController {/api/recordings}: +0ms
[0] [Nest] 10712 - 31/05/2025, 21.51.17 LOG [RouterExplorer] Mapped {/api/recordings/seasons/:seasonId, GET} route +1ms
[0] [Nest] 10712 - 31/05/2025, 21.51.17 LOG [RouterExplorer] Mapped {/api/recordings/score/last, GET} route +0ms
[0] [Nest] 10712 - 31/05/2025, 21.51.17 LOG [RouterExplorer] Mapped {/api/recordings/:id, GET} route +1ms
[0] [Nest] 10712 - 31/05/2025, 21.51.17 LOG [RoutesResolver] SeasonsController {/api/seasons}: +0ms
[0] [Nest] 10712 - 31/05/2025, 21.51.17 LOG [RouterExplorer] Mapped {/api/seasons, GET} route +0ms
[0] [Nest] 10712 - 31/05/2025, 21.51.17 LOG [RouterExplorer] Mapped {/api/seasons/current, GET} route +0ms
[0] [Nest] 10712 - 31/05/2025, 21.51.17 LOG [RouterExplorer] Mapped {/api/seasons/:id, GET} route +0ms
[0] [Nest] 10712 - 31/05/2025, 21.51.17 LOG [RouterExplorer] Mapped {/api/seasons/:id/rules, GET} route +1ms
[0] [Nest] 10712 - 31/05/2025, 21.51.17 LOG [RoutesResolver] SlackController {/api/slack}: +0ms
[0] [Nest] 10712 - 31/05/2025, 21.51.17 LOG [RouterExplorer] Mapped {/api/slack/seasons, POST} route +0ms
[0] [Nest] 10712 - 31/05/2025, 21.51.17 LOG [RouterExplorer] Mapped {/api/slack/season, POST} route +1ms
[0] [Nest] 10712 - 31/05/2025, 21.51.17 LOG [RouterExplorer] Mapped {/api/slack/teams, POST} route +0ms
[0] [Nest] 10712 - 31/05/2025, 21.51.17 LOG [RouterExplorer] Mapped {/api/slack/team, POST} route +0ms
[0] [Nest] 10712 - 31/05/2025, 21.51.17 LOG [RouterExplorer] Mapped {/api/slack/interact, POST} route +0ms
[0] [Nest] 10712 - 31/05/2025, 21.51.17 LOG [RoutesResolver] TeamsController {/api/teams}: +0ms
[0] [Nest] 10712 - 31/05/2025, 21.51.17 LOG [RouterExplorer] Mapped {/api/teams, GET} route +0ms
[0] [Nest] 10712 - 31/05/2025, 21.51.17 LOG [NestApplication] Nest application successfully started +5ms

```

3. Jalankan bot starter pack dengan mengunduh folder bot pada tautan ini

<https://github.com/haziqam/tubes1-IF2211-bot-starter-pack/releases/tag/v1.0.1>

a. Setelah berhasil diunduh, lakukan ekstraksi file.zip tersebut dan masuk ke root dari folder hasil ekstraksi tadi, lalu buka terminal

b. Jalankan command berikut pada terminal untuk masuk ke root directory dari bot starter pack

```
cd tubes1-IF2211-bot-starter-pack-1.0.1
```

c. Install dependencies dengan pip

```
pip install -r requirements.txt
```

d. Gunakan command berikut untuk menjalankan satu bot

```
python main.py --logic MyBot --email=your_email@example.com --name=your_name
```

```
--password=your_password --team etimo
```

User juga dapat menjalankan lebih dari satu bot dengan menjalankan script berikut
Untuk Windows

```
./run-bots.bat
```

Untuk Linux/macOS

```
./run-bots.sh
```

User dapat mengatur script yang ada di run-bots.bat atau run-bots.s dari **logic**, **email**, **nama**, dan **password** yang digunakan.

e. Untuk merancang bot, buatlah file baru pada direktori/game/logic (misalnya MyBot.py)

f. Buatlah sebuah kelas yang meng-inherit kelas BaseLogic, lalu implementasikan constructor beserta method next_move pada kelas tersebut

```
class Pendekar(BaseLogic):
    def __init__(self):
        self.directions = [(1, 0), (0, 1), (-1, 0), (0, -1)]
        self.goal_position: Optional[Position] = None
        self.current_direction = 0

    def next_move(self, board_bot: GameObject, board: Board):
        props = board_bot.properties
        base = props.base
        radius = board.height // 2
        time_left = props.milliseconds_left / 1000
        bot_position = board_bot.position
```

g. Import kelas yang telah dibuat tadi dalam *main.py* dan daftarkan pada dictionary *CONTROLLERS*

```

main.py > ...
1  import argparse
2  from time import sleep
3
4  from colorama import Back, Fore, Style, init
5  from game.api import Api
6  from game.board_handler import BoardHandler
7  from game.bot_handler import BotHandler
8  from game.logic.random import RandomLogic
9  from game.util import *
10 from game.logic.base import BaseLogic
11 from game.logic.mybot import Pendekar
12 from game.logic.pasukan.highestValue import HighestValue
13 from game.logic.pasukan.density2 import Panglima
14 from game.logic.pasukan.shortestDistance import shortestdistance
15 from game.logic.pasukan.closestBase import Jendral
16
17
18 init()
19 BASE_URL = "http://localhost:3000/api"
20
21 DEFAULT_BOARD_ID = 1
22 CONTROLLERS = {
23     "Random": RandomLogic,
24     "pendekar": Pendekar,
25     "closestbase": Jendral,
26     "density": Panglima,
27     "shortdistance": shortestdistance
28 }

```

h. Jalankan program seperti pada step d pada bagian 3 dengan menyesuaikan argumen logic pada argumen logic pada command/script dengan nama yang terdaftar pada *CONTROLLERS*. User juga masih bisa menjalankan satu bot saja atau beberapa bot dengan run-bots.bat atau run-bots.sh.

BAB III

APLIKASI STRATEGI *GREEDY*

3.1 Proses Mapping

Dalam algoritma greedy, proses mapping memiliki peran penting karena membantu merepresentasikan data dalam bentuk yang lebih mudah diproses oleh program. Mapping memungkinkan kita untuk mengubah informasi yang kompleks, seperti nama objek atau posisi tertentu, menjadi format yang lebih terstruktur seperti indeks, bobot, atau nilai numerik lainnya. Tujuan utama dari mapping adalah untuk mempermudah proses pengambilan keputusan lokal terbaik di setiap langkah algoritma. Karena greedy bekerja dengan prinsip memilih solusi optimal secara bertahap, mapping sangat dibutuhkan agar algoritma dapat menilai dan membandingkan setiap pilihan dengan cepat dan akurat. Selain itu, mapping juga memudahkan dalam melakukan pelacakan status dari elemen-elemen yang sudah dipilih atau belum, sehingga mencegah duplikasi atau kesalahan dalam pemilihan. Dengan representasi data yang baik, algoritma juga menjadi lebih efisien baik dari segi waktu maupun penggunaan memori. Oleh karena itu, proses mapping merupakan tahap penting yang mendukung keberhasilan pendekatan greedy dalam menyelesaikan berbagai permasalahan optimasi.

3.1.1 Himpunan Kandidat

Dalam permasalahan ini terdapat beberapa yang bisa dimasukkan dalam himpunan kandidat, antara lain:

Nama Objek	Penjelasan
Base	Objek ini berfungsi sebagai markas utama bagi bot. Ketika bot kembali ke markas sambil membawa N buah diamond, maka bot akan memperoleh N poin sesuai jumlah diamond yang dibawanya. Selain itu, markas ini juga menjadi tempat respawn bagi bot apabila bot tersebut terkena tackle oleh bot lain.
Red Button	Ketika bot melewati objek ini, seluruh

	<p>diamond yang ada di papan permainan akan di-regenerate dan posisinya akan diacak ulang. Selain itu, posisi dari tombol merah (red button) juga akan berubah secara acak setelah objek ini dilewati.</p>
Diamond	<p>Saat bot melewati objek ini, ia akan menambahkan sejumlah diamond ke dalam inventory-nya sebanyak N. Nilai N ditentukan berdasarkan jenis diamond yang diambil — jika bot mengambil red diamond, maka akan mendapatkan 2 diamond, sedangkan jika yang diambil adalah blue diamond, maka hanya mendapatkan 1 diamond.</p>
Teleporter	<p>Dalam setiap permainan, hanya ada dua teleporter yang aktif pada satu waktu. Ketika bot melewati objek ini, bot akan “masuk” ke dalam teleporter A dan secara otomatis akan muncul kembali di teleporter B.</p>
Inventory	<p>Inventory adalah tempat penyimpanan sementara bagi diamond yang dikumpulkan bot selama permainan berlangsung. Bot hanya bisa membawa maksimal 5 diamond dalam inventory-nya. Jika bot mencoba mengambil diamond saat slot sudah penuh, maka diamond tersebut tidak akan bisa diambil. Untuk bisa mengambil diamond lagi, bot harus kembali ke base terlebih dahulu untuk mengosongkan inventory-nya.</p>
Bot Musuh	<p>Dalam setiap permainan, akan ada bot musuh yang juga ikut bertanding. Bot musuh memiliki karakteristik yang sama seperti bot kita, termasuk dalam hal base, penggunaan teleporter, perolehan diamond, dan sistem inventory. Mereka bisa melakukan tackle dengan cara melewati posisi bot kita. Jika bot kita terkena tackle, maka seluruh diamond yang ada di inventory akan diambil alih oleh bot musuh. Sebaliknya, kita juga bisa men-tackle bot musuh dan mendapatkan semua diamond yang mereka bawa di inventory.</p>

3.1.2 Himpunan Solusi

Dengan pemilihan objek-objek tersebut maka dapat dikatakan bahwa himpunan solusi dari algoritma ini adalah path yang diambil oleh bot.

3.1.3 Fungsi Solusi

Algoritma ini memeriksa jarak terdekat untuk ke Base, RedButton, dan Diamond dan memilih berdasarkan syarat pada fungsi seleksi.

3.1.4 Fungsi Seleksi

Seleksi tujuan dalam permainan dapat disimpulkan sebagai berikut :

Nama Objek	Sebagai Syarat
Base	Bot akan memprioritaskan kembali ke base jika: <ol style="list-style-type: none">1. Inventory sudah penuh, yaitu saat bot membawa 5 diamond.2. Bot membawa lebih dari 2 diamond dan posisi base berada dalam jarak kurang dari 2.3. Bot sedang membawa diamond dan waktu yang tersisa kurang dari 10 detik.
Red Button	Bot akan memilih red button sebagai tujuan jika jumlah diamond yang tersisa di papan kurang dari atau sama dengan 6, dan jarak ke red button lebih dekat dibandingkan dengan diamond terdekat.
Diamond	Meskipun terdapat beberapa diamond di board, bot akan memilih diamond yang paling dekat sebagai target.
Teleporter	Bot akan memilih teleporter sebagai tujuan jika:

	<ol style="list-style-type: none"> 1. Total jarak dari bot ke teleporter A ditambah jarak dari teleporter B ke diamond lebih pendek dibandingkan jarak langsung ke diamond. 2. Total jarak dari bot ke teleporter A ditambah jarak dari teleporter B ke base lebih pendek dibandingkan jarak langsung ke base.
Bot Musuh	<p>Bot akan menargetkan bot musuh jika:</p> <ol style="list-style-type: none"> 1. Bot musuh sedang membawa lebih dari 2 diamond dan berada dalam jarak kurang dari 3 satuan. 2. Bot musuh berada tepat 1 satuan dari posisi bot kita.

3.1.5 Fungsi Kelayakan

Memeriksa jumlah diamond yang dibawa tidak melebihi 4 poin setiap kali akan bergerak. Jika telah mencapai 4 poin, maka bot akan langsung bergerak kembali menuju base.

3.1.6 Fungsi Objektif

Jumlah diamond yang didapat maksimum

3.2 Eksplorasi Alternatif Solusi Greedy

Beberapa Alternatif Solusi Greedy yang kami eksplorasi untuk persoalan ini adalah sebagai berikut.

3.2.1 Greedy By Highest Value

Greedy by Highest Value adalah algoritma Greedy yang fokus mencari diamond dengan nilai poin tertinggi di board. Karena dalam game ini hanya ada dua jenis poin diamond—1 poin (Blue Diamond) dan 2 poin (Red Diamond)—maka bot akan memprioritaskan Red Diamond terdekat selama masih tersedia di board. Jika semua Red Diamond sudah diambil, bot akan beralih ke Blue Diamond terdekat, dan akan kembali mengejar Red Diamond begitu mereka muncul lagi.

3.2.2 Greedy By Highest Density

Greedy by Highest Density adalah sebuah strategi greedy yang mengutamakan diamond yang memiliki value poin per jarak terbesar. Bot ini akan menghitung terlebih dahulu besarnya poin per jarak dari setiap diamond yang ada di papan permainan dan memilih diamond yang memiliki value poin per jarak yang terbesar,

3.2.3 Greedy by Tackling

Greedy by Point per Tackling adalah algoritma Greedy yang mencari bot musuh terdekat yang sedang membawa diamond. Bot akan bergerak menuju posisi bot lawan tersebut dengan tujuan untuk melakukan tackling dan mengambil diamondnya.

3.2.4 Greedy by Closest Base

Greedy by Closest Base adalah algoritma Greedy yang memilih diamond yang lokasinya paling dekat dengan base. Tujuan dari strategi ini adalah supaya bot nggak terlalu jauh dari base, jadi bisa balik lebih cepat dan menyimpan diamond sebagai poin tanpa buang-buang waktu.

3.3 Analisis Efisiensi dan Efektivitas Solusi Greedy

No	Alternatif Solusi (Greedy by)	Kelebihan	Kekurangan
1	Highest Density	<ol style="list-style-type: none">1. Mengutamakan rasio poin/biaya gerak2. Efektif diawal game3. Cocok untuk area yang memiliki banyak diamond	<ol style="list-style-type: none">1. Mengabaikan nilai diamond2. Apabila diamond langka, dapat pergi jauh dari base tanpa hasil yang maksimal3. Tidak dapat memperhitungkan kapan waktunya untuk kembali ke base
2	Highest Value	<ol style="list-style-type: none">1. Diamond yang diambil selalu yang memiliki poin tinggi2. Cocok untuk area yang sedikit namun bernilai tinggi	<ol style="list-style-type: none">1. Sering kembali ke base dengan perjalanan yang terlalu panjang2. Tidak adaptif, apabila diamond dengan nilai tinggi terambil oleh musuh dapat

			menimbulkan kebingungan
3	Closest Base	<ol style="list-style-type: none"> 1. Cocok untuk map yang memiliki banyak teleporter atau obstacle yang menyebabkan waktu kembali ke base lama 2. Selalu memprioritaskan jarak ke base 	<ol style="list-style-type: none"> 1. Skor per trip rendah sering sekali kalah poin dengan strategi lain 2. Sering kembali ke base tanpa hasil yang maksimal
4	Tackling	<ol style="list-style-type: none"> 1. Adaptif, dimana diamond yang ada di jalur pulang akan diambil juga 2. Sangat cocok untuk map yang linear 	<ol style="list-style-type: none"> 1. Tidak selalu mendapatkan diamond dengan nilai tinggi 2. Rentan terkena blocking oleh musuh

3.4 Strategi Greedy yang Dipilih

Setelah mengevaluasi berbagai opsi strategi greedy yang telah dibahas pada bagian 3.1, kelompok kami memilih untuk menggunakan strategi Greedy by Highest Density sebagai pendekatan utama dalam mengembangkan bot. Pilihan ini diambil karena mempertimbangkan durasi pertandingan Diamonds yang berlangsung selama 60 detik, dengan asumsi satu aksi dapat dilakukan tiap detik. Dalam situasi seperti ini, kecepatan dan ketepatan dalam mengumpulkan diamond menjadi faktor yang sangat penting. Strategi Greedy by Highest Density memungkinkan bot memprioritaskan diamond dengan rasio nilai terhadap jarak tertinggi, sehingga lebih optimal dalam memaksimalkan poin. Pendekatan ini juga memiliki kelebihan dalam hal memaksimalkan total poin yang bisa dikumpulkan oleh bot. Dengan memberikan prioritas pada diamond yang memiliki rasio poin terhadap jarak paling tinggi, bot dapat meraih poin secara lebih efisien. Keunggulan ini memberikan nilai kompetitif yang kuat selama pertandingan, karena memungkinkan bot untuk meraih skor setinggi mungkin dalam waktu yang terbatas. Berdasarkan pertimbangan tersebut, strategi Greedy by Highest Density terbukti menjadi solusi yang tepat dan efisien untuk bot kami dalam game Diamonds berdurasi 60 detik. Strategi ini meningkatkan peluang bot untuk meraih kemenangan dengan cara yang strategis dan terukur dalam mengumpulkan poin.

BAB IV

IMPLEMENTASI DAN PENGUJIAN

4.1 Implementasi Algoritma Greedy

4.1.1 Pseudocode

```
#Fungsi ini adalah fungsi yang menghasilkan integer, dimana fungsi akan menerima 2 buah parameter bertipe position

def calculate_distance(a: Position, b: Position) -> int:
    return ((a.x - b.x)**2 + (a.y - b.y)**2)

#Fungsi direction_correcter ini adalah fungsi yang menghasilkan 2 integer, dimana fungsi akan menerima 2 buah parameter bertipe integer.
def direction_correcter(deltax, deltay):
    if deltax == 0 and deltay == 0:
        delta_x, delta_y = random.choice(
            [(1, 0), (0, 1), (-1, 0), (0, -1)])
    return delta_x, delta_y
    return deltax, deltay

#fungsi find_nearest_diamond fungsi ini adalah fungsi yang akan mengasilkkan sebuah position, yang dimana fungsi akan menerima 2 buah parameter bertipe position dan list yang berisi diamond.
def find_nearest_diamond(current: Position, diamonds: List[GameObject]) -> Optional[Position]:
    if not diamonds:
        return None
    nearest_diamond = min(
        diamonds, key=lambda diamond: calculate_distance(current, diamond.position))
    return nearest_diamond.position

# Fungsi find_nearest_teleporter_pair fungsi ini adalah fungsi yang menghasilkan sebuah tuple yang berisi 2 position, dimana fungsi ini akan menerima 2 buah parameter dengan tipe position dan list yang berisi teleporter.
def find_nearest_teleporter_pair(current: Position, teleporters: List[GameObject]) -> Optional[Tuple[Position, Position]]:
    nearest_teleporter_pair = None
```

```

min_distance = float('inf')

teleporter_pairs = {}
for teleporter in teleporters:
    pair_id = teleporter.properties.pair_id
    if pair_id not in teleporter_pairs:
        teleporter_pairs[pair_id] = []
    teleporter_pairs[pair_id].append(teleporter)

for pair_id, pair in teleporter_pairs.items():
    if len(pair) == 2:
        teleporter, paired_teleporter = pair
        distance = calculate_distance(current, teleporter.position)
        if distance < min_distance:
            min_distance = distance
            nearest_teleporter_pair = (
                teleporter.position, paired_teleporter.position)

return nearest_teleporter_pair

```

#fungsi get_direction_pribadi fungsi ini adalah fungsi yang mengoreksi direction yang dihasilkan agar tidak error, dimana fungsi ini akan menerima 4 buah parameter bertipe integer dan sebuah list yang berisi position dari teleporter.

```

def get_direction_pribadi(current_x, current_y, dest_x, dest_y, avoid_teleporters=[]):
    delta_x = clamp(dest_x - current_x, -1, 1)
    delta_y = clamp(dest_y - current_y, -1, 1)

    if (current_x + delta_x, current_y + delta_y) in avoid_teleporters and (dest_x, dest_y) not in avoid_teleporters:
        alternative_moves = [(1, 0), (0, 1), (-1, 0), (0, -1)]
        alternative_moves.remove((delta_x, delta_y))
        for move in alternative_moves:
            if (current_x + move[0], current_y + move[1]) not in avoid_teleporters:
                return move
        return (delta_x, delta_y)
    else:
        if delta_x != 0:

```

```

        delta_y = 0
    return (delta_x, delta_y)

#fungsi is_on_path_or_close fungsi ini adalah fungsi yang menghasilkan sebuah integer, dimana
fungsi ini menerima 3 buah parameter bertipe position,dan literal.
def is_on_path_or_close(diamond_position, bot_position, base_position, threshold):
    on_path = (diamond_position.x == bot_position.x == base_position.x) or (
        diamond_position.y == bot_position.y == base_position.y)
    close_to_path = calculate_distance(diamond_position, bot_position) <= threshold or
calculate_distance(
    diamond_position, base_position) <= threshold
    return on_path or close_to_path

#fungsi next_move fungsi ini adalah fungsi yang akan menghasilkan 2 integer, dimana fungsi ini akan
menerima 3 buah parameter bertipe self, gameObject, dan board.
def next_move(self, board_bot: GameObject, board: Board):
    props = board_bot.properties
    base = props.base
    time_left = props.milliseconds_left / 1000
    bot_position = board_bot.position

    diamond_game_objects = []
    teleport_game_objects = []
    diamond_button_game_objects = []
    for obj in board.game_objects:
        if obj.type == 'DiamondGameObject':
            diamond_game_objects.append(obj)
        elif obj.type == 'TeleportGameObject':
            teleport_game_objects.append(obj)
        elif obj.type == 'DiamondButtonGameObject':
            diamond_button_game_objects.append(obj)

    diamonds_with_2_points = [
        diamond for diamond in diamond_game_objects if diamond.properties.points == 2]
    diamonds_with_1_point = [
        diamond for diamond in diamond_game_objects if diamond.properties.points == 1]

```

```

nearest_diamond_with_2_points = find_nearest_diamond(
    bot_position, diamonds_with_2_points)
nearest_diamond_with_1_point = find_nearest_diamond(
    bot_position, diamonds_with_1_point)

nearest_teleporter_pair = find_nearest_teleporter_pair(
    bot_position, teleport_game_objects)
time_to_reach_base = math.ceil(math.sqrt(calculate_distance(
    board_bot.position, base)))
if nearest_teleporter_pair:
    teleporter1, teleporter2 = nearest_teleporter_pair
    nearest_teleporter = teleporter1 if calculate_distance(
        bot_position, teleporter1) <= calculate_distance(bot_position, teleporter2) else teleporter2
    paired_teleporter = teleporter2 if nearest_teleporter == teleporter1 else teleporter1

if props.diamonds >= props.inventory_size - 1:
    distance_to_base = calculate_distance(bot_position, base)
    distance_to_base_via_teleport = calculate_distance(
        paired_teleporter, base) + calculate_distance(bot_position, nearest_teleporter)

    if distance_to_base_via_teleport < distance_to_base:
        self.goal_position = nearest_teleporter
    else:
        path_diamonds_with_1_points = [diamond for diamond in diamonds_with_1_point if
is_on_path_or_close(
        diamond.position, bot_position, base, threshold=4) and diamond.properties.points +
props.diamonds <= props.inventory_size]
        if path_diamonds_with_1_points:
            nearest_path_diamond_with_1_points = find_nearest_diamond(
                bot_position, path_diamonds_with_1_points)
            self.goal_position = nearest_path_diamond_with_1_points
        else:
            self.goal_position = base
elif (props.diamonds >= 2 and time_left < time_to_reach_base + 3):
    distance_to_base = calculate_distance(bot_position, base)
    distance_to_base_via_teleport = calculate_distance(
        paired_teleporter, base) + calculate_distance(bot_position, nearest_teleporter)
    if distance_to_base_via_teleport < distance_to_base:

```

```

        self.goal_position = nearest_teleporter
    else:
        self.goal_position = base
    elif diamonds_with_2_points:
        nearest_diamond_with_2_points = find_nearest_diamond(
            bot_position, diamonds_with_2_points)
        self.goal_position = nearest_diamond_with_2_points
    else:
        self.goal_position = nearest_diamond_with_1_point
    if self.goal_position:
        if bot_position == nearest_teleporter and self.goal_position == nearest_teleporter:
            self.goal_position = base
            delta_x, delta_y = get_direction_pribadi(
                bot_position.x, bot_position.y, self.goal_position.x, self.goal_position.y)
        else:
            teleporter_positions = [
                teleporter.position for teleporter in teleport_game_objects]
            delta_x, delta_y = get_direction_pribadi(
                bot_position.x, bot_position.y, self.goal_position.x, self.goal_position.y,
                avoid_teleporters=teleporter_positions)

    else:
        delta = self.directions[self.current_direction]
        delta_x = delta[0]
        delta_y = delta[1]
        if random.random() > 0.6:
            self.current_direction = (
                self.current_direction + 1) % len(self.directions)

    if delta_x == 0 and delta_y == 0:
        delta_x, delta_y = direction_correcter(delta_x, delta_y)

    return delta_x, delta_y

```

4.2 Struktur Data yang Digunakan

Bahasa pemrograman yang kami gunakan untuk implementasi bot ini adalah bahasa pemrograman python. Struktur data dari program di definisikan dalam class pada file [models.py](#).

Class yang digunakan dalam implementasi dan pengembangan bot terdapat di dalam folder game.

Class yang ada di dalam [models.py](#) adalah sebagai berikut.

- Class bot adalah class yang berisi informasi tentang bot terkait nama, email, dan juga id dari bot tersebut
- Class position merepresentasikan posisi dari objek yang terdiri dari koordinat x dan y
- Class base adalah kelas turunan dari position yang berisi informasi tentang koordinat x dan y untuk base dari sebuah bot
- Class properties menjelaskan berbagai properti dari objek yang ada dalam permainan sebagai berikut
 - Points yang merupakan properti milik diamond yang berisi informasi tentang jumlah poin dari setiap jenis diamond, diamond biru memiliki poin 1 dan diamond merah memiliki poin 2,
 - Pair_id merupakan properti milik teleporter yang isinya id untuk teleporter agar dapat dilakukan pairing,
 - Diamonds merupakan informasi dari jumlah diamond yang dibawa oleh bot,
 - Score adalah informasi dari jumlah poin dari diamond yang berhasil dibawa ke bot ke base,
 - Name adalah informasi mengenai nama dari objek tersebut,
 - Inventory_size merupakan properti milik bot yang berisi jumlah poin maksimal yang dapat dibawa oleh bot,
 - Can_tackle berisikan informasi apakah sebuah objek dapat ditabrak atau tidak,
 - Milisecond_left berisikan informasi mengenai sisa waktu yang dimiliki oleh bot dalam permainan,
 - Time_joined berisikan informasi mengenai waktu masuk dari objek ke dalam permainan,
 - Base berisi informasi mengenai koordinat dari base sebuah bot.
- Class GameObject adalah sebuah kelas yang berisikan informasi mengenai objek seperti id, position, type, dan properties. Id bertipe integer, position bertipe class position, type bertipe string, dan properties bertipe class properties
- Kelas Board adalah kelas yang menyimpan data terkait papan permainan, seperti id, lebar (width), tinggi (height), fitur-fitur (features), minimum_delay_between_moves, dan game_objects. Atribut minimum_delay_between_moves menentukan jeda minimum antar pergerakan bot, sedangkan game_objects memuat informasi mengenai objek-objek yang ada di dalam papan permainan. Kelas ini juga memiliki fungsi untuk memeriksa apakah pergerakan sebuah bot tergolong valid atau tidak.

Program juga memerlukan file [util.py](#) agar dapat berjalan. Fungsi-fungsi yang ada di dalam file ini adalah sebagai berikut.

- Fungsi clamp menerima tiga parameter: n, smallest, dan largest, lalu mengembalikan nilai maksimum antara smallest dan nilai minimum dari n atau largest.
- Fungsi get_direction membutuhkan koordinat x dan y dari sebuah bot, serta koordinat x dan y dari objek lain di papan. Fungsi ini akan mengembalikan delta_x dan delta_y yang menunjukkan arah gerakan bot terhadap objek tersebut.
- Fungsi position_equals menerima dua parameter, yaitu position a dan position b, untuk mengecek apakah kedua posisi tersebut bernilai sama atau tidak.

Program ini juga membutuhkan beberapa file logic yang berperan sebagai versi alternatif dari bot dengan strategi greedy. File-file tersebut berisi fungsi next_move yang menentukan pergerakan bot berdasarkan strategi greedy yang digunakan. File logic ini juga melakukan import terhadap fungsi get_direction dari util.py untuk membantu menentukan arah gerakan. Selain itu, terdapat file main.py yang berfungsi sebagai penghubung utama antara semua komponen dalam bot starter pack.

4.3 Pengujian Program

Pada bagian ini akan dilakukan pengujian beberapa fitur utama dari bot yang di pilih, yaitu *greedy bot by highest density*.

4.3.1 Skenario Pengujian

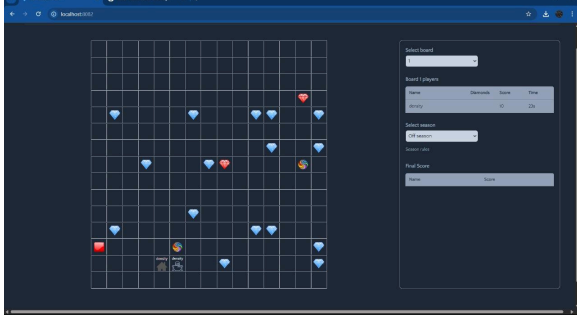
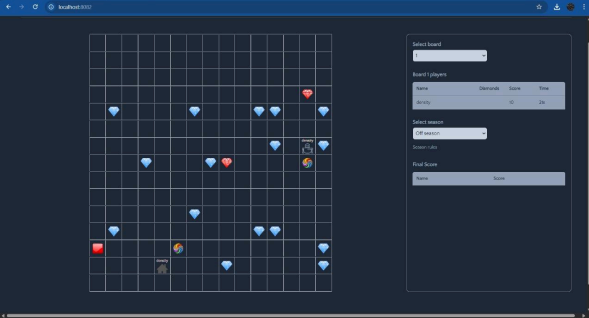
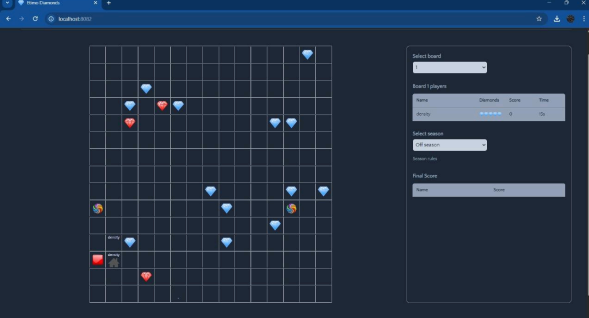
Pada bagian ini dilakukan pengujian terhadap perilaku bot utama yang menggunakan strategi *greedy by highest density*, guna memastikan seluruh komponen logika telah berjalan sesuai ekspektasi. Pengujian ini dilakukan pada beberapa skenario permainan untuk memverifikasi keakuratan pengambilan keputusan oleh bot dalam kondisi berbeda-beda. Berikut ini adalah rincian skenario pengujian yang dilakukan

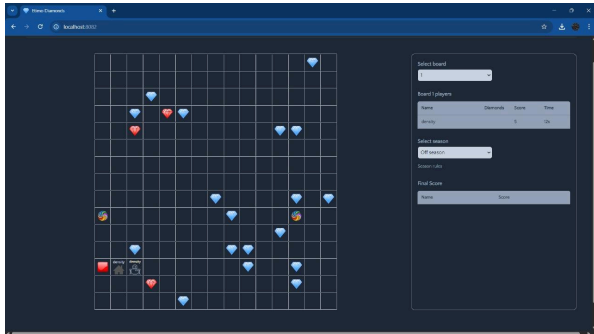
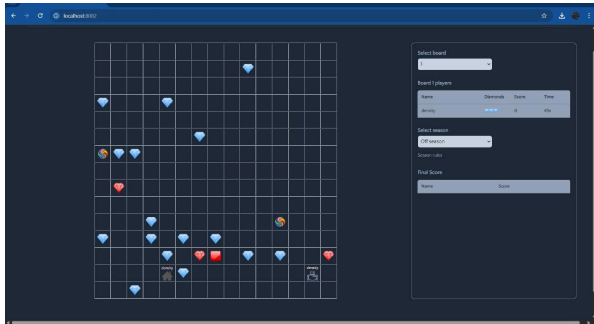
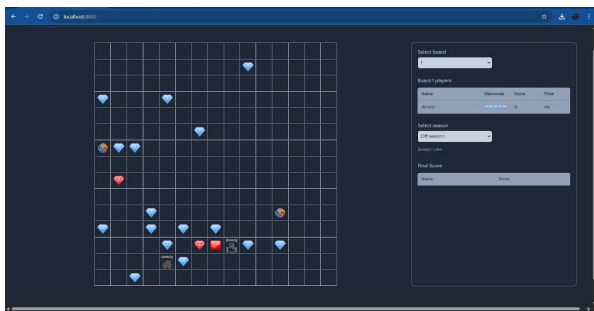
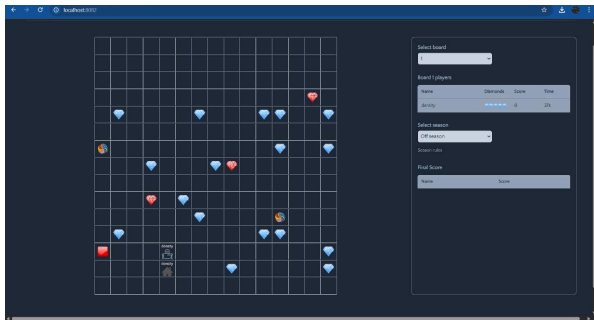
No	Skenario pengujian	Tujuan
1	Pemilihan diamond berdasarkan densitas	Menguji apakah strategi <i>greedy by highest density</i> diterapkan dengan benar
2	Penggunaan teleporter	Memastikan bot memanfaatkan teleporter jika jarak melalui teleporter lebih pendek
3	Pemilihan red button	Menguji logika pemilihan red button saat diamond pada board minim dan tidak efisien untuk dikejar

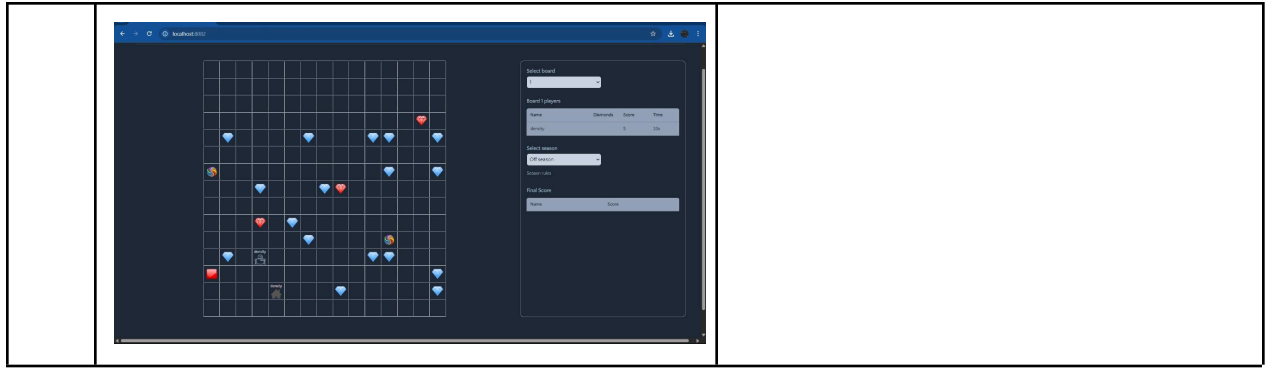
4	Keputusan pulang ke base	Memastikan bot pulang tepat waktu ke base dan mengambil diamond di jalan jika memungkinkan
---	--------------------------	--

4.3.2 Hasil Pengujian dan Analisis

Berikut ini adalah hasil dari pengujian yang telah dilakukan

No	Gambar	Penjelasan
1	<p>Kondisi awal</p>  <p>Kondisi akhir</p> 	<p>Gambar di samping menunjukkan bot sebelum dan sesudah melakukan teleportasi menuju area diamond. Teleportasi dipilih karena jarak ke lokasi diamond lebih cepat ditempuh melalui teleporter dibandingkan jika hanya berjalan biasa. Dari gambar tersebut, dapat disimpulkan bahwa fitur teleportasi telah berfungsi sebagaimana mestinya.</p>
2	<p>Kondisi awal</p>  <p>Kondisi akhir</p>	<p>Pada gambar kondisi awal, terlihat bahwa terdapat diamond biru yang posisinya lebih dekat dengan bot. Namun, bot memilih untuk mengambil diamond merah yang berjarak dua kotak karena bot bertipe high density, yang memprioritaskan diamond dengan rasio nilai poin terhadap jarak yang lebih tinggi.</p>

		
3	<p>Kondisi awal</p>  <p>Kondisi akhir</p> 	<p>Gambar di samping menunjukkan bahwa bot memilih untuk menuju red button. Keputusan ini diambil karena slot diamond pada inventory bot sudah penuh. Dari kedua gambar tersebut, dapat disimpulkan bahwa implementasi logika pemilihan red button telah berjalan dengan baik sesuai kondisi yang dihadapi.</p>
4	<p>Kondisi awal</p>  <p>Kondisi akhir</p>	<p>Gambar di samping memperlihatkan bahwa bot langsung kembali ke base setelah inventory-nya penuh. Dalam kondisi ini, bot tidak lagi mencari diamond tambahan di sekitarnya, melainkan segera pulang untuk menyetorkan diamond yang telah dikumpulkan. Hal ini menunjukkan bahwa implementasi logika pemulangan bot saat inventory penuh telah berjalan dengan semestinya.</p>



BAB V

KESIMPULAN DAN SARAN

5.1 Kesimpulan

Dari tugas besar Strategi Algoritma ini, kami telah melakukan beberapa eksperimen dengan berbagai macam bot untuk permainan Diamonds menggunakan strategi greedy. Setelah melalui serangkaian uji coba dan perbandingan antar strategi, kami akhirnya memilih bot dengan strategi kepadatan terbesar (highest density) sebagai opsi final. Pemilihan ini didasarkan pada hasil pertandingan antar bot dengan berbagai pendekatan strategi.

Hasil pertandingan menunjukkan bahwa bot dengan strategi kepadatan terbesar mampu mengumpulkan diamond dengan rata-rata paling tinggi dibandingkan dengan bot yang menggunakan strategi Closest base. Berdasarkan pengamatan selama pertandingan, bot ini dapat bekerja cukup optimal karena mempertimbangkan kepadatan diamond terbesar, memanfaatkan teleporter untuk jalur tercepat menuju diamond, dan menjaga pergerakan yang efektif ke base.

Eksperimen ini dilakukan pada board berukuran 15x15 dengan waktu permainan selama 1 menit. Berdasarkan hasil dan pengamatan, dapat disimpulkan bahwa bot dengan strategi kepadatan terbesar telah menunjukkan performa yang cukup optimal, mengingat fitur-fitur yang digunakan, rata-rata skor akhir setiap bot, serta konfigurasi papan permainan.

5.2 Saran

Saran untuk proses pengembangan bot dalam tugas besar ini adalah sebagai berikut

1. Penulisan kode sebaiknya dilakukan dengan lebih terstruktur agar mempermudah proses debugging.
2. Menghindari menyelesaikan tugas besar secara terburu-buru menjelang tenggat waktu agar bot yang dibuat bisa dikembangkan dengan lebih optimal dan matang.

LAMPIRAN

Repository [Github](#)

DAFTAR PUSTAKA

- [1] Diamond Game Engine. Diakses 30 Mei 2025
<https://github.com/haziqam/tubes1-IF2211-game-engine/releases/tag/v1.1.0>
- [2] Bot Starter Pack. Diakses 30 Mei 2025
<https://github.com/haziqam/tubes1-IF2211-game-engine/releases/tag/v1.1.0>
- [3] R. Munir, “Algoritma Greedy (Bagian 1),” *informatika.stei.itb.ac.id*, [Online]. Tersedia pada:
[https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2020-2021/Algoritma-Greedy-\(2021\)-Bag1.pdf](https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2020-2021/Algoritma-Greedy-(2021)-Bag1.pdf). Diakses 31 Mei 2025
- [4] R. Munir, “Algoritma Greedy (Bagian 2),” *informatika.stei.itb.ac.id*, [Online]. Tersedia pada:
[https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2020-2021/Algoritma-Greedy-\(2021\)-Bag2.pdf](https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2020-2021/Algoritma-Greedy-(2021)-Bag2.pdf). Diakses 01-Juni-2025
- [5] R. Munir, “Algoritma Greedy (Bagian 3),” *informatika.stei.itb.ac.id*, [Online]. Tersedia pada:
[https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2021-2022/Algoritma-Greedy-\(2022\)-Bag3.pdf](https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2021-2022/Algoritma-Greedy-(2022)-Bag3.pdf). Diakses: 01 Juni 2025.